Register file optimisation

Henrik Sundkøien Sørvik

Master of Science in Electronics

Submission date:	June 2017
Supervisor:	Kjetil Svarstad, IES
Co-supervisor:	Ronan Barzic, Nordic Semiconductor ASA
	Asghar Havashki, Nordic Semiconductor ASA

Norwegian University of Science and Technology
Department of Electronic Systems

Abstract

The increasing demand for low-power solutions and small area usage makes it necessary to explore power efficient solutions for critical parts of a system as the data storage in the register file. The register file is an array of registers capable of storing data. This thesis explores two different approaches to optimise the register file.

The first approach exploits the regular structure of a register file. By making a generator that makes relative placement directives, it is possible to place the cells of the register file in a regular structure. This is done using a placement tool that uses these directives to place the register file cells, instead of letting the tool do the placement optimisation.

The second approach looks at the architecture of the memory cells of the register file. The selection logic is one-hot encoded and two latch based designs are implemented to reduce both power consumption and area usage. Both latch based designs utilises the fact that a flip-flop can be modelled as a pair of a master latch and a slave latch. By sharing either the slave latches or the master latches, the number of latches needed for the register file is reduced. A Shared Master-latch design and a Shared Slave-latch design are implemented. All the designs are validated after implementation using existing testbenches designed for the system that the register file is a part of. The designs are synthesized and the power consumption are estimated in order to evaluate the optimisation compared to the original, flip-flop based register file.

The one-hot encoding of the selection signal proved to reduce the activity of the register file with 38.67 percent, thus reducing the dynamic power consumption with 38.41 percent. The latch based designs further improved the power consumption in the register file, with the Shared Master-latch design most efficient with an improvement of 53.42 percent on the original design and 24.36 percent on the one-hot encoded design. The Shared Slave-latch design deemed an improvement of 46.75 percent on the original design and 13.55 percent on the one-hot encoded design.

In terms of area usage showed the results that the Shared Slave-latch design was most efficient. Mostly because latches are smaller than flip-flops and that the Shared Master-latch design added extra latches to delay some of the logic. In the end did the Shared Slave-latch design uses 13.93 percent less area than the original design, and the Shared Master-latch design uses 9.97 percent less. On the other hand, did the area usage increase by 6.66 percent in the design with the one-hot encoded selection logic.

Sammendrag

I en verden hvor markedet vil ha raskere og mindre løsninger for innvevde systemer er det viktig å undersøke metoder for å forminske effektforbruket og størrelsen til integrerte kretser. Minnet til en integrert krets er en del av den kritiske stien og er ofte implementert som en registerfil En registerfil er en matrise bestående av prosessorregistre. Registerfilen er dermed en viktig del av en integrert krets som datalagringsenhet, og denne oppgaven fokuserer på metoder å optimalisere slike registerfiler.

Den første metoden som er utforsket i denne oppgaven er å utnytte den regulære strukturen til en registerfil. Ettersom en registerfil består av en rekke registre, er teorien at det er mulig å minske størrelsen og effektforbruket til registerfilen ved å eksplisitt plassere disse registrene. Normalt plasseres disse av verktøyet som brukes når registerfilen fysisk plasseres på en integrert krets, men ved å gi enkelte direktiver til verktøyet kan registrene plasseres relativt til hverandre, i stedet for å bli plassert der verktøyet tror det passer best. Et program skrevet i Python er lagd for å generere disse direktivene som inneholder informasjon om hvordan disse registrene skal plasseres relativt til hverandre.

Den andre metoden for å optimalisere registerfilen ser på selve arkitekturen til registerfilen. Valglogikken, logikken som styrer hvilken data som skal leses eller skrives fra eller til registerfilen, blir gjort om til «one-hot» kodet logikk. I tillegg blir to design basert på «latcher» utviklet for å redusere både effektforbruk og størrelsen på registerfilen. Begge «latch» designene utnytter det faktum at en «flip-flop» kan bli sett på som et par bestående av en «master-latch» og en «slave-latch». Ved å dele «master-slave» paret i to og dele en av de for hele registerfilen, vil antall «latcher» reduseres betraktelig. Et design som deler på «master-latchene» og et design som deler på «slave-latchene» er utviklet i forbindelse med denne oppgaven, hvor begge designene er videreutviklet fra den samme «one-hot» kodete valglogikken.

Alle designene er validert etter implementering ved hjelp av eksisterende testbenker designet for systemet som registerfilen er en del av. De er så syntetisert og effektforbruket er estimert for å kunne evaluere optimaliseringen gitt av de ulike løsningene og for å kunne sammenligne de med hverandre og den opprinnelige, «flip-flop» baserte registerfilen.

Å kode valglogikken «one-hot» viste seg å redusere aktiviteten i registerfilen og dermed redusere det dynamiske effektforbruket. Designene som var basert på «latcher» forbedret registerfilen ytterligere med tanke på både størrelse og effektforbruk, ettersom «latcher» er både mindre enn «flip-floper» og har et mindre effektforbruk.

Preface

This thesis is the final work of a Master of Science degree in electronics with a specialisation in digital circuit design. The work is submitted to the department of Electronic Systems. The assignment was proposed by Nordic Semiconductor in late 2016. The work officially started on the 12th of January 2017 and was submitted on the 19th of June the same year.

The intention of the thesis is to explore ways of implementing the register file in one of Nordic Semiconductors existing systems to see if there are better alternatives to the existing register file architecture.

I would like to thank professor Kjetil Svarstad from NTNU, and Asghar Havaskhi and Ronan Barzic from Nordic Semiconductor for guidance, support and weekly meetings. I also wish to express my gratitude towards Nordic Semiconductor for letting me use a workspace, relevant tools and a development environment to work on the register file. A special mention to Liguori Aniello for the collaboration on the generator for relative placement directives.

Henrik Sundkøien Sørvik
Trondheim, 19th June 2017

Contents

List of Figures

List of Tables

Acronyms

DC	Design Compiler. 19, 43, 44, 48, 50, 55, 65, 66, 72
ICC	IC Compiler. 7, 19, 40, 43, 44, 54–56, 67, 72, 77
QoR	Quality of results. 5, 6, 9
RTL	Register Transfer Language. 3, 24, 31, 62, 66, 67
SRAM	Static Random Access Memory. xi, 15, 16, 24

Chapter 1

Introduction

Current register files are generally implemented using standard flip-flop element taken from standard cell libraries. This may not be an optimal solution in term of performance, area and power. This thesis will propose alternate solutions to the register file architecture.

Different architectures are proposed in the Background theory chapter, a Multi-bit flip-flop design, a Shared Master-latch design, a pulsed-latch design, a Shared Slave-latch design and a Multi-port SRAM design.

The Shared Master-latch design and the Shared Slave-latch design are implemented. They are implemented because of the complexity of the designs and availability of latch cells in the standard cell libraries. This was not the case for Multi-bit flip-flops cells, thus was this design approach not implemented. The Multi-port SRAM design and the pulsed-latch design were deemed to complex to implement during the course of the work of this thesis.

The original register file is modified to two separate files, and with the selection logic changed to one-hot encoded logic. This proved some advantages compared to the original design, especially in terms of switching activity, so this is also presented in the thesis as a optmisation scheme.

The architectures are to be run through verification and synthesis. The verification asserts that the functionality is correct and the system is working properly. Synthesis are done to generate results on power consumption, area usage and timing, parameters to compare the different designs to each other. Waveforms generated from verification are further used to run power estimation on the designs, using the SpyGlass Power Estimate tool.

Another take on register file optimisation is researched in this thesis. Register files have a highly regular structure. In this case the register file consists of 32 registers of 32 flip-flops each. Arranging them in columns could improve the system in terms of power consumption, area usage and timing. To accomplish this certain directives are made to the Route & Place tool, to force the tool to group the register file cells in a regular structure. A script to generate these directives are to be made in Python.

This script produces the directives and a graphical presentation of the placement of the cells. When the register file are placed with a Route & PLace tool, are the directives fed to the tool. This makes the tool follow these directives to generate the system with the register cells placed in a regular structure. The tool can then be used to analyse the area usage, power consumption and timing to compare to the original register file structure.

1.1 Background and Motivation

The increasing demand for low-power solutions and small area usage makes it necessary to explore power efficient solutions for critical parts of a system as the data storage in the register file. This is a field much explored and there exist numerous literature on register file optimization. Chapter 2 present different alternatives to optimize register file architecture. Various approaches as relative placement of register file cells and memory cell architecture optimization are presented, with provided literature. Different approaches on the architectural level are discussed, based on improving the regular flip-flop based architectures, latch based architectures and RAM based architectures.

1.2 Goals

The goals of this thesis is to select some of the optimization techniques presented in Chapter 2 and explore those techniques. The different approaches will be functionally verified and thoroughly tested.

The first part is done to ensure that the explored design techniques are functionally equivalent to the original register, which the thesis is trying to improve.

After this, power, area and timing analysis will be applied to get important parameters to compare with the original. In the end are the results compared and evaluated, to see if an alternate approach to the register file design are better or worse than the regular flip-flop based architecture currently in use.

1.3 Research Method

The research method in this thesis is based on the same principles for the exploration of alternate register file architectures as in the experiment with relative placement of flip-flops in the register file.

The principle is to start with a clean, working system that passes all its verification. Then it is possible to swap the register file with a alternate register file, and then, by getting the same results from the verification, can the new register file be seen as functionally equivalent to the original. Then it is possible to run the same analysis on the original design as on the new, that being Place & Route, synthesis or power estimation, and compare the results given by the analysis.

This principle is followed by the relative placement experiment as well, as the Route & Place analysis is run with and without the relative placement directives to have comparable results.

The implementation and architecture of the different architectures and optimisation schemes are presented in Chapter 5 and in Chapter 4. The verification and analysis are further described in Chapter 6.

1.4 Thesis Structure

This thesis is divided into following chapters:

- Introduction - Present the goals and motivation behind this project.
- Background Theory - Includes theory important to understand the work that is described in this thesis and research on different optmisation approaches to register file design.
- Relevant tools - Presents relevant tools for the research and implementation of the thesis.
- Architecture - Describes the researched architectures of the thesis They are described at block level of the design.
- Implementation - This chapter describes how the architectures in Chapter 4 are implemented using RTL code.
- Experiments and Results - Describes how the alternate designs are functionally verified, synthesized, run placement analysis and estimated power consumption. The results are presented by each design and optimisation scheme.

- Evaluation and Discussion - Evaluates the design based on the results presented in the previous chapter, and discusses different topics that came along with the implementation, together with comparing the different optimisation schemes.
- Conclusion - Presents a conclusion on the project given the results presented and discusses possible future work on this topic.

Chapter 2

Background Theory

The reduction of power dissipation in modern microchips is one of the most important challenges. With increased die size and advanced process technologies, whole systems with an ever-growing number of transistors are integrated on a single chip. A demand for portable consumer products with independent power supply increase the demand for development of low-power design techniques [Debopam Ghosh and Mukherjee [2013]].

Area is an important aspect as well, as it is desirable to make the chip as small as possible. This is to make the product more portable, to be able to have as many transistors as possible, and the area usage often corresponds to the power consumption.

Data stores are an important power critical parts of a microchip system, and is normally implemented as a synthesizable register file described as part of the design on the register transfer level. A register file is essentially an array of registers with common addressing logic.

This chapter will present different techniques to optimise register files, both by exploring alternate architectures and by looking at the placement of the register file cells on the chip.

2.1 Relative placement optimization

Relative placement is the method of controlling the placement of cells during the place and route design stage. As the name implies, it works by telling the placement tool how a group of cells are supposed to be placed relatively to each other. The method is described in Synopsis SolvNet [rel [2017]]. The article states that relative placement usually is applied to datapaths and registers, thus beneficial for improving a register file. Relative placement can be used to explore Quality of results (QoR) benefits such as shorter wire lengths, reduced congestion, better timing, skew control, fewer vias, better yield and lower dynamic and leakage power [rel [2017]].

Relative placement constraints created for a system implicitly generates a matrix structure of the instances and annotate the placement of the instances. The annotated netlist that are made as a result are used for physical optmisation, where the tool preserves the structure and places it as a single entity or group, as shown in Figure 2.1.

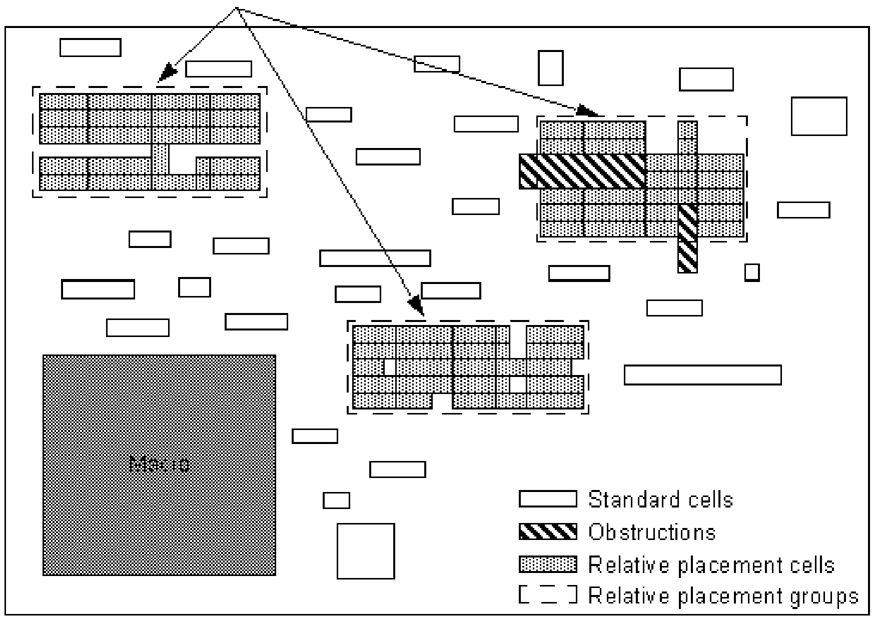


Figure 2.1: Relative Placement in a Floorplan
rel [2017]

2.1.1 Benefits of Relative Placement

Along with being technology-independent and having the ability to improve routability, relative placement provides the following benefits [rel [2017]]:

- Reduces the placement search space in critical areas of the design, which improves the predictability of the QoR and congestion.

- Maintains relative placement during placement, optimisation, clock tree synthesis, and routing.
- Provides a method for maintaining structured placement for legacy or intellectual property (IP) designs.
- Handles flat and hierarchical designs.
- Allows sizing of relative placement cells while maintaining relative placement.

2.1.2 Creating Relative Placement Groups

A relative placement group is an association of cells, other relative placement groups, and blockages. The relative placement group is defined by the number of columns and rows it consist of. A relative placement group is created issuing the command *create_rp_group* to the IC Compiler. The name of the group is specified by the *-name* option. The number of rows and columns of the relative placement group is by default one, but can be specified by the options *-rows* and *-columns.*

The example in Listing 2.1 shows how a relative placement group named *RP1*, with six columns and rows, is made with the ICC.

Listing 2.1: Example of creating a Relative Placement Group

```
icc_shell> create_rp_group −name RP1 −columns 6 −rows 6
```

Figure 2.2 shows a relative placement group. It consist of six rows and columns. Worth to note is that the column count begins from the left, and the row count begins in the bottom. I.e. is the cell with position 0,0 placed in the bottom left corner of the group. The width of a column is equal to the widest cell in the column, and similarly is the height of a row the same as the tallest cell in the row. In the example in Figure 2.2 are not all the positions in the structure in use.

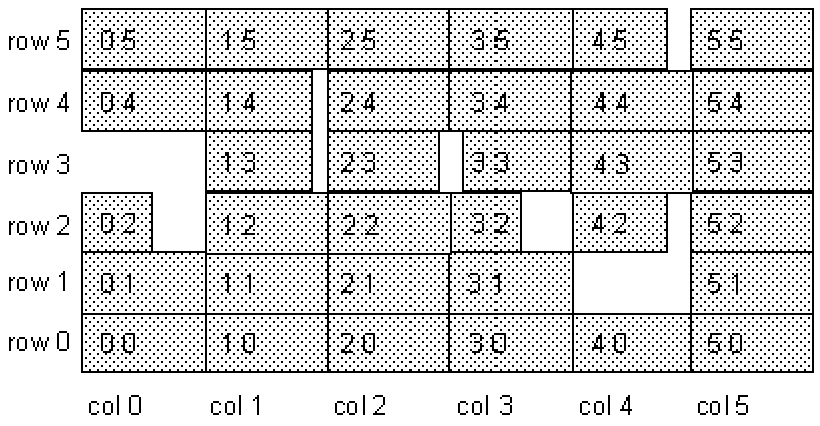


Figure 2.2: Relative Placement Column and Row Positions
rel [2017]

2.2 Architectural optimisation

One way to improve the existing register files is to change the architecture of the register file itself. Small changes in area, performance or power consumption in one single memory element will drastically change the traits of the register files as it could be of a large size. Different techniques exploit different traits of the register file. Some techniques improve the registers of the register array that is the register file, while other techniques exploits the traits of the whole register array.

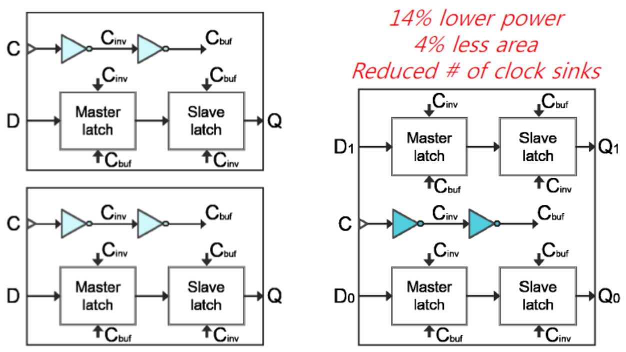
2.2.1 Improved flip-flop architectures

An approach to optimise register files is to improve the already existing flip-flop architectures. A flip-flop is an edge-sensitive circuit that has two stable states that can be used to store state information. The most common register consist of flip-flops, where each flip-flops stores one bit. A register file based on flip-flops contains as many flip-flops as its size.

The edge-sensitive characteristic requires extra logic for clocking, but increases the reliability of the sequential circuitry.

2.2.1.1 Multi-bit flip-flop cells

Gourav Kapoor and Gupta [2014] discuss the idea of multi-bit flip-flop cells. The clock is in a basic flip-flop memory structure fed into all the flip-flops. By reducing the clock network, for example by removing the number of clock buffers, the overall QoR of the design will improve. The attribute of a multi-bit flip-flop memory architecture reduces the number of clock buffers, thus reducing the overall QoR.



(a) Two 1-bit flip-flops (b) 2-bit flip-flop

Figure 2.3: 2 bit flip-flop example
Gourav Kapoor and Gupta [2014]

Figure 2.3 shows how two one-bit master-slave flip-flops can be merged into one two-bit master-slave flip-flop. It can be seen in the figure that the number of combinational logic gates required for the clocks are reduced from four to two. This deems a reduction of 14 percent of power consumption, and 4 percent less area.

The shared clock network between all the single-bit flip-flops of a multi-bit flip-flops changes the structure drastically as all the single-bit elements are physically placed nearby. This trait resolves some challenges with physical design implementation and could reduce the area usage of the register file.

2.2.2 Latch-based architectures

Latches require naturally less area and power usage than flip-flops because of the level-triggered behavior, rather than being edge-triggered. This is because they do not require the gating logic that makes the flip-flops edge-triggered. Figure 2.5 and 2.4 shows an D latch with enable and a master-slave positive-edge-triggered D flip-flop, respectively. As it shows, the D latch needs fewer logic elements to be realised than a D flip-flop. The latch in Figure 2.5 have additional enable logic, which are not necessarily present in a D latch, thus making it possible to have memory elements with even less logic gates.

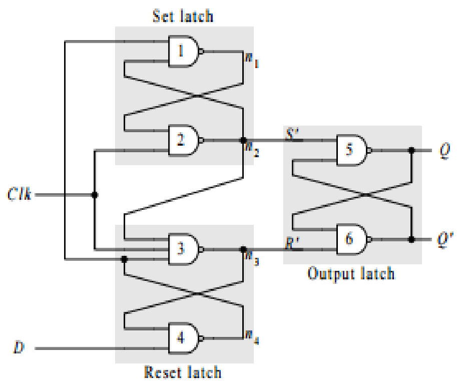


Figure 2.4: Master-slave positive-edge-triggered D flip-flop
Yildiz [2017]

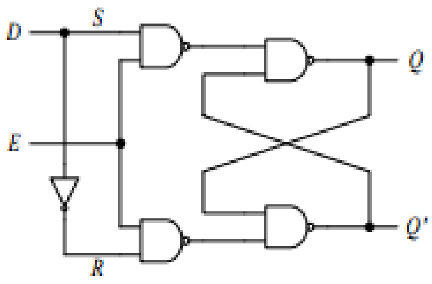


Figure 2.5: D latch with enable
Yildiz [2017]

Timing is important in ASIC design and especially in the design of the memory. It is not possible to replace the flip-flops in a register file with latches without any extra means of control. It would be highly unreliable or would require a lot of additional control logic which would contribute to unnecessary power consumption and area usage, and it could be slower. There exists different solutions to the timing problems when dealing with latches which will be discussed in the following subsections.

2.2.2.1 Master Latch Sharing

This solution is described in M. Wróblewski and Nossek [2003]. This paper introduces a method of reducing area and power consumption of a synthesisable register file by using a single master-latch shared by a number of slaves. A common storage element is the edge-triggered master-slave D flip-flop, which consist of a master-latch and a slave-latch as shown in Figure 2.6.

To store a value in the flip-flop, the master-latch freezes the value with the triggering edge of the clock signal and the slave-latch becomes transparent with the value visible at the output of the flip-flop. At the other edge, the slave-latch stores the value while the master-latch is transparent.

For data stores, w flip-flops are grouped to word-level registers, where w means the number of bits of the data signal. Each bit of the data signal is connected to

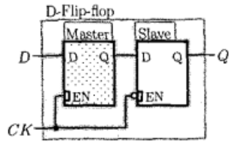


Figure 2.6: Master-slave D-flip-flop

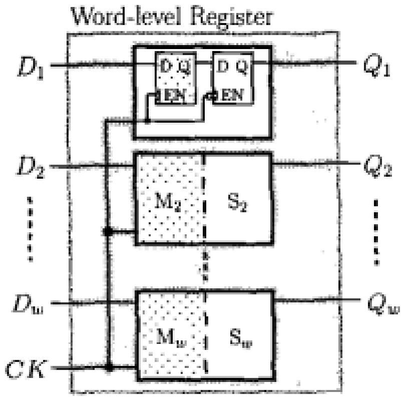


Figure 2.7: Word-level register
M. Wróblewski and Nossek [2003]

one master-slave D-flip-flop of the word-level register, as can be seen in Figure 2.6. Register files, in which r data values can be stored, consist of r word-level registers, as shown in Figure 2.7. All of the data inputs of all the registers are connected to the data bus, leading to a high bus capacitance.

Two approaches are commonly used to update data stored in registers. The first approach is to use a feedback loop. The feedback loop is connected from the output to the input of the flip-flop, via a multiplexer. If there is a new value to be stored, then it is routed via the multiplexer to the input. If not, the output is connected to the input and thus rewriting the stored information every clock cycle. This requires extra logic to control the multiplexer.

The other approach is to use clock gating. This requires much less power consumption than the feedback loop and thus usually preferred. This method was applied to the register file shown in Figure 2.8. Logic, and possible sequential cells represented as *CG*, are added to the clock path. The same enable logic as in the prior approach described are used to decide whether or not to clock to a given register. Thus data is written only when it changes and clock signals of not selected registers are disabled.

A consequence of the clock gating approach is that every master is always transparent, except when new data is written to its slave. Every transition on the data bus is repeated by all master latches, but usually it is only one that needs to pass the new state to its slave. This requires unnecessary power consumption which the method of shared master circuits are trying to get rid of.

It is possible to store into many registers simultaneously, but all the registers

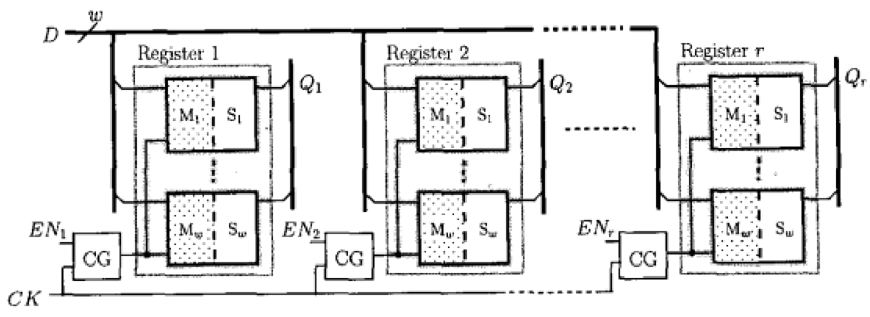


Figure 2.8: Conventional register file with flip-flops
M. Wróblewski and Nossek [2003]

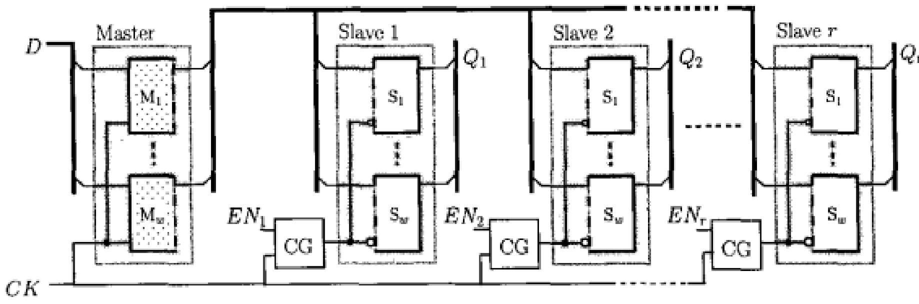


Figure 2.9: Modified register file with shared master latches
M. Wróblewski and Nossek [2003]

that has clock signal enabled receive the same value. Under this assumption, it is possible to replace all r master latches that are connected to a specific bit-line of the data bus by one single master latch. The corresponding slave latches share this master latch, this architecture is shown in Figure 2.9. This architecture limits the register file to maximum store one value per clock cycle. It is however possible to enable multiple stores by adding a new set of master latches, a general rule is that each input data port to a Shared Master-latch register file needs to be latched, i.e. needs a set of master latches.

The Shared Master-latch architecture has the following advantages compared to the regular master-slave D-flip-flop. The capacitive load is reduced, since there is only one or few master latches connected to the data bus. Transitions seen on the data bus cause only the internal capacitances of a single latch to be recharged, not everyone. This reduces the power consumption. Lastly, the number of latches are reduced, in a register file with n x m bits, from $2 * n * m$ to $m + n * m$. Fewer latches deems less area usage and power consumption in the register file.

The method presented in M. Wróblewski and Nossek [2003] presents results on saving in power consumption of up to 50 percent. There are, however, some timing issues that needs to be resolved and the number of simultaneously stores are limited by the number of master latches.

2.2.2.2 Slave Latch Sharing

Kime et al. [2007] describes a Shared Slave-latch based approach on the register file. In opposite of the shared master latch architecture are the slave latches shared and thus reducing the number of slave latches. That the slave latches are shared, means that the output ports of the register file are latched. The case is the opposite on the Shared Master-latch approach, where the input ports are latches.

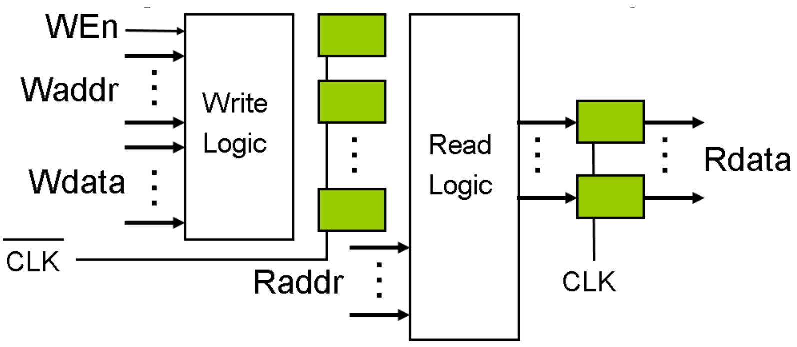


Figure 2.10: Shared slave latch register file
Kime et al. [2007]

Figure 2.10 is a simple representation of a Shared Slave-latch register file. The master latches are the green blocks between the write logic and the read logic, and the shared slave latches are the green blocks after the read logic. The master latches are clocked by the inverted clock, $\overline{\text{CLK}}$ and the slave latches by CLK. That the slaves and the masters are clock gated by different phases of the clock, ensures that both latch groups are never transparent at the same time, ensuring reliable data storage. The benefit of this approach are much of the same as the shared master latch approach, the number of latches are reduced from a normal master-slave design. In a normal master-slave register file with $n * m$-bit, $2 * n * m$ latches are needed. With the shared latch approach, the number of slave latches is reduced from n^2 to n, making the total number of latches $m + n * m$, as the same amount of master latches are required.

2.2.2.3 Pulsed-Latch Circuits

Shin and Paik [2011] describes register files based on pulsed-latch architecture. A pulsed-latch architecture tries to exploit both the advantages of latches and flip-flops, namely the performance and size of latch based solutions and the timing and the reliability of flip-flops. This offers a higher performance and lower power consumption in ASIC design. Shin and Paik [2011] identifies a design methodology and tools for pulsed-latch ASICs to explore this register file architecture.

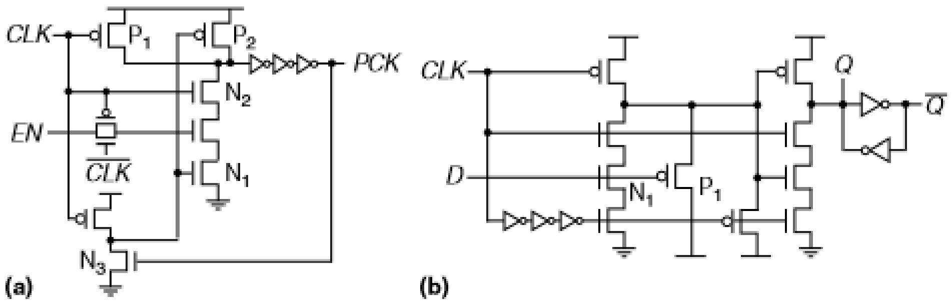


Figure 2.11: a) Pulser and b) Pulsed flip-flop

In a pulsed-latch circuit, the clock is driving multiple pulse generators, called pulsers, which are dispersed in a placement region. Figure 2.11 a) shows a pulser, which in 45 nm technology consumes 5 times the amount of power as a regular latch. This makes it important to keep the number of pulsers as low as possible. A pulser can be integrated in a latch, a pulsed flip-flop, to avoid distortion of the pulse shape. Figure 2.11 b) is such a pulsed flip-flop. The pulse is not explicitly generated, but rather emulated by the inverter chain, where the delay is equivalent with the pulse width. To gain the advantage of both shared pulser (pulsed-latch) and no disortion of pulse shape (pulsed flip-flop), it is possible to integrate more than one latch with a single pulser. This yields a pulsed register, which is possible to use to make a register file based on pulsed latches.

A pulsed latch is like a faster flip-flop with a longer hold time. It is possible to change a traditional ASIC design with flip-flops after synthesis to a pulsed-latch version, by simply replacing the flip-flops with latches. Additional pulsers has to be inserted and properly placed, and the correct number of pulsers, to guarantee the integrity of the pulse shape.

2.2.3 Other architectures

This subsection will cover architectures which does not categorize as latch based or based on flip-flops.

2.2.3.1 Multi-port SRAM Architecture

Multiport Static Random Access Memory (SRAM) with support for several simultaneous write and read operations is a common building block in design of register files in many processing units, graphics processing units and digital signal processors [Hsiao and Wu [2014]]. As the processing units become more complex, the number of read and write ports required in the register file increases. For example does the register file of the Intel Itanium microprocessor use 12-read and 10-write operations. Hsiao and Wu [2014] uses the ARM memory generator

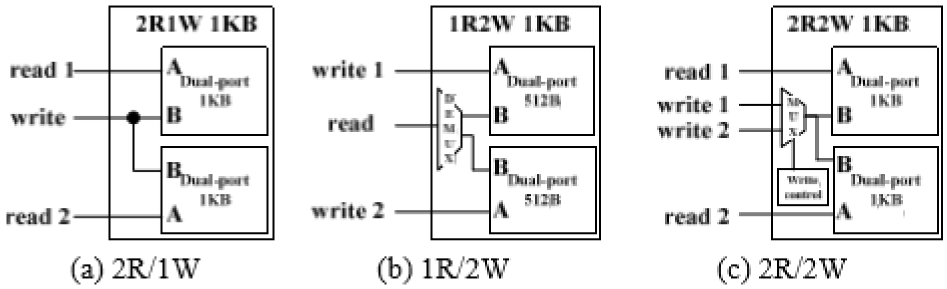


Figure 2.12: Designs of multiport 1KB SRAM of (a) 2R/1W, (b) 1R/2W, and (c) 2R/2W, based on dual-port SRAM from ARM memory generator
Hsiao and Wu [2014]

to generate the SRAM memory cells, which is constricted to Dual-port memory cells. To enable register files with additional read or write ports, constructs as shown in Figure 2.12 as used. Figure 2.13 shows a single bit of SRAM memory.

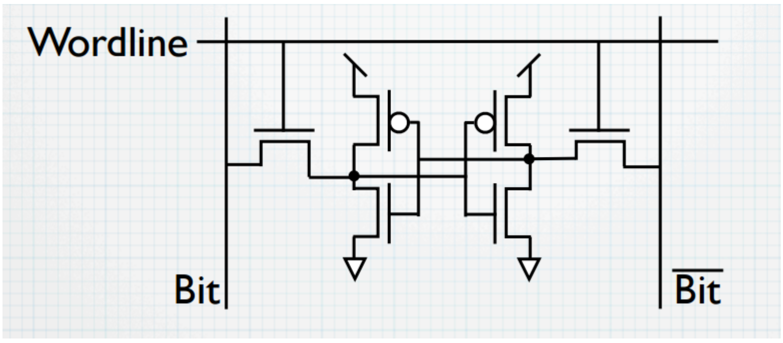


Figure 2.13: SRAM bitcell
John Wawrzynek and [TA]

On-chip memories use SRAM memories built upon large arrays of the bitcells in Figure 2.13. As seen are the SRAM designed at Transistor-Level, a lower level in the design hierarchy than the Register-Transfer level.

Hsiao and Wu [2014] states that there is not efficient to design multiport memory with many read and write ports from the limited dual-port memory available from ARM, proposes the following multiport SRAM cell circuits.

Figure 2.14 shows various one read and one write mulitport SRAM cell circuits, where the number of transistors varies from six to nine. Table 2.1 shows a comparison of the circuits from Figure 2.14. The table shows the trade-off between area, delay, power and robustness. The article further move on with the design in Figure 2.14 (f) because of the overall characteristics.

1R/1W SRAM cells	**layout area (um^2)**	**critical adelay (ns)**	**total power (nW)**	**leakage power (nW)**	*hold SNM (V)*	*read SNM (V)*	*write SNM (V)*
6T	5.18	0.05	13.30	8.15	0.33	0.11	0.25
7T	5.37	0.08	17.45	9.07	0.33	0.32	0.25
8T	5.54	0.08	18.75	9.11	0.33	0.32	0.44
9T	5.82	0.09	17.40	7.25	0.33	0.31	0.44
Proposed	5.45	0.09	16.10	7.42	0.32	0.31	0.38

Table 2.1: Comparison of Various 1R/1W SRAM cell circuits
Hsiao and Wu [2014]

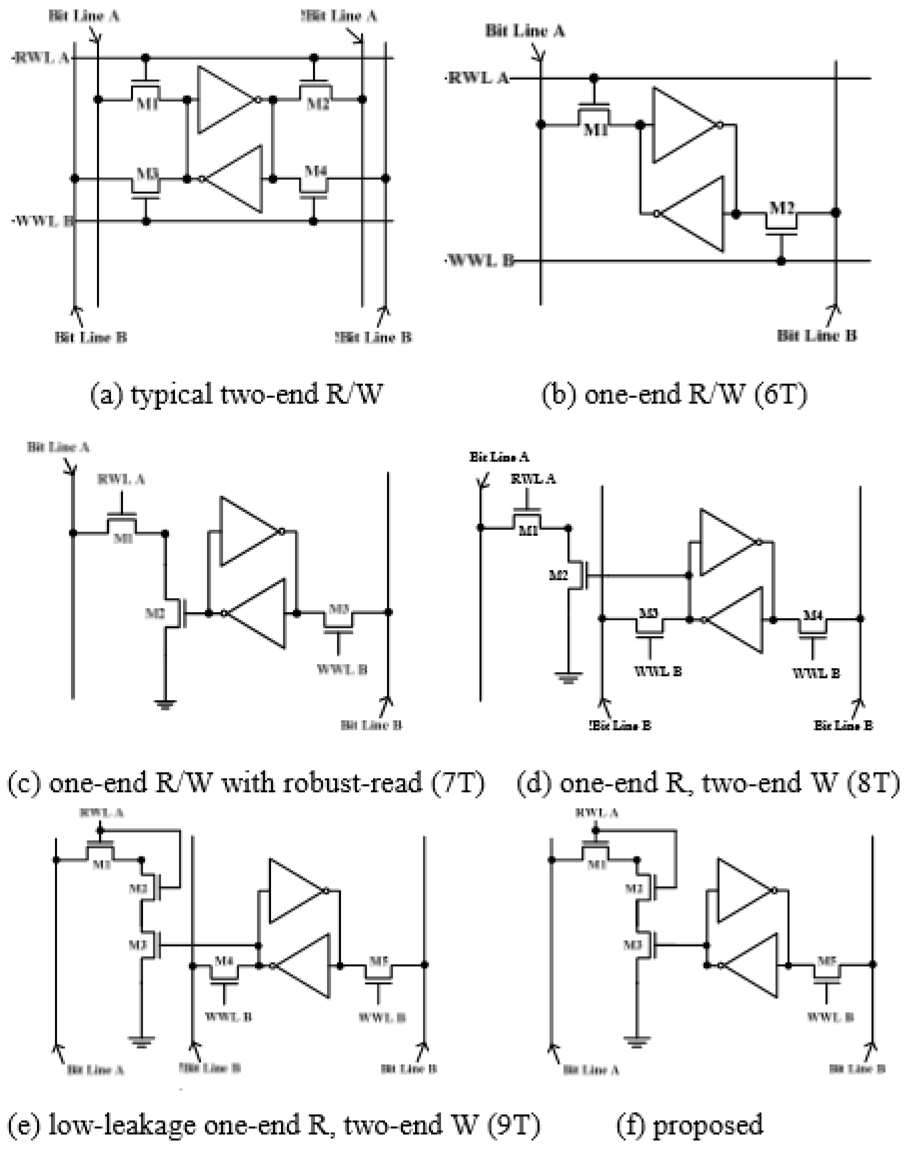


Figure 2.14: Various 1R/1W multiport SRAM cell circuits
Hsiao and Wu [2014]

Chapter 3

Relevant tools

This chapter focuses on the tools that were used during the coarse of this thesis. The tools were used to run synthesis, route & place analysis and power estimation.

3.0.1 Design Compiler

The Design Compiler (DC) is a tool to run synthesis on register-transfer level designs. The DC is provided by Synopsys and is described in Synopsys [2017a]. During synthesis the DC optimizes the design to provide small, fast logical representations. The synthesis produces technology-dependent, gate-level designs from the provided HDL descriptions. DC includes algorithms for optimisation for timing, area, power and test, and provides a solution to ensure that results correlate to layout, eliminating costly iteration between synthesis and physical implementation.

3.0.2 IC Compiler

The IC Compiler (ICC) is a Route & Place system provided by Synopsys. It is a tool for chip-level physical implementation and is described in Synopsys [2017b]. It includes flat and hierchical design planning, placement, clock tree synthesis, routing and optimisation. For this thesis the relevant capabilities are the placement possibility, to test the effect of the relative placement scheme. How relative placement is done with the ICC is described in Section 2.1. The ICC contains the capability of running analysis on area usage, timing and power consumption on the physical implementaion.

3.0.3 SpyGlass Power Estimate

The SpyGlass Power Estimation is a tool used to estimate the power consumption of a design while still at the register-transfer level [Synopsys [2017e]]. The tool uses an area-based synthesis engine to map the register-transfer level design onto a gate netlist. This synthesis engine uses most of the cells available in the defined target libraries. This provides a reliable starting point for power estimation. As well as the provided cells, SpyGlass adds virtual cells to model the clock tree and the high-fanout nets. The produced netlist allows SyGlass to calculate the contribution of static and dynamic power, to the total power. The SpyGlass Power Estimate tool can also use simulation and parasitics data to help with the calculations.

3.0.3.1 Leakage Power

Leakage Power is primarily sourced from unwanted current in the transistors channels when they are switched off. This is what is often referred to as static power. The leakage power of a certain cell is defined in the target library's Liberty file, either as a fixed value or as a function of the power pins state and the gate inputs. The total leakage power is then the sum of the leakage power of all the cell present in the design. The total leakage power are reported by SpyGlass in categories such as combinational, sequential and memories. More on how leakage power are working are described in Synopsys [2017d]

3.0.3.2 Internal Power

Internal power is the power dissipated whitin the boundary of a cell when a transition occurs [Synopsys [2017c]]. The internal power is the first component to dynamic power. It is defined in the Liberty files provided from the target library. Specifically, the Liberty file contains energy consumption data for each transition as a function of the input transition (slew) and the output load. Activity information coming from simulation data is used by SpyGlass to estimate how often a cell is toggling, and can from this derive power consumption figures. This is reported by SpyGlass in similar categories as the leakage power.

3.0.3.3 Switching Power

The Switching Power is the amount of energy consumed per unit of time in the nets which interconnect the gates. More on how it is calculated is described in Synopsys [2017f]. It is proportional to the capacitance of the net, and to the square value of the voltage. The capacitance is computed by adding the contribution of the cell pin found in the Liberty file, to the contribution from the wire itself. The wire load are either given as input to SpyGlass or modeled with the Wire Load Model. The switching power is also proportional to the toggling activity of the net which is derived from simulation data.

Chapter 4

Architecture

This chapter will focus on the architecture of the different solutions for realizing the register file. The original design will be presented, as well as the one-hot encoded design, and both the latch based design, the Shared Master-latch design and and the Shared Slave-latch design. First the design choices will be discussed.

4.1 Design choices

This chapter presents the architectures of the solutions chosen to be explored in this thesis. Those solutions are both latch based, the Shared Master-latch design and the Shared Slave-latch design. In addition a modified version of the original, flip-flop based register file is made, with the only edition being that the selection logic of the register file is changed to be one-hot encoded. The latch based designs were chosen as the theory in Chapter 2 proposed solid gains in terms of area usage and power consumption, and the the complexity were not too great. Since both alternate solutions are latch based, either by latching the input or the output, and are basing the register file on latches, makes it easier to compare to each outher.

The one-hot encoded design are made as a mean to split up the hierarchy of the register file, and move the architecture of the register file to its own module, making it easier to implement new solution afterwards. The one-hot encoding was added to save switching activity, reducing the power consumption of the register file. Both the latch based designs contains the same one-hot encoded selection logic as well.

As of the designs not chosen from Chapter 2 there are different reasons. The multi-bit flip-flop register file would add variety in the thesis in terms of using another approach than latches, but the libraries containing the multi-bit flip-flops cells were not implemented, making it impossible to synthesise a design with multi-bit flip-flops.

The multiport SRAM approach were based on a design approach done at a lower level of abstraction, namely on the transistor level, while this thesis is working on the register-transfer level. Therefor this optimisation scheme were not implemented. The last architectural optimisation scheme presented in Chapter 2 was the Pulsed latch circuit. This as well did not contain the optimisation in the RTL code itself, but rather inserting certain pulsers after synthesis. This proved to complex of a task for this thesis, and outside the focus on RTL optimisation.

4.2 Standard Flip-flop design

The standard flip-flop design is the base of this project and it already existed in the environment of the project. The main goal of this thesis is to investigate alternate solutions to the design presented in this section.

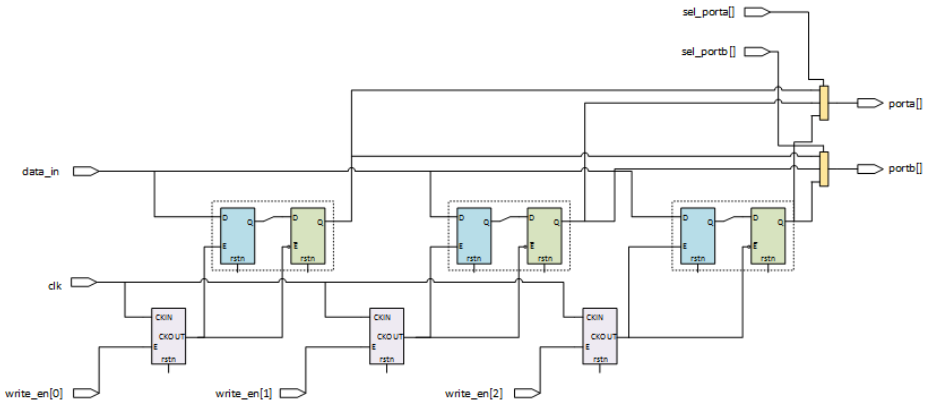


Figure 4.1: Standard Flip-Flop based register file topology

Figure 4.1 is an example of a topology of a standard flip-flop based register file. It is a two read, one write 3x1-bit register file, as opposed to the system which has a register file with two writes and two reads, and 32x32-bits. This simplification is done for the simplicity of the figure, which still shows the characteristic of the design.

The 3x1-bit register file has one data input port. It is connected to all the flip-flops of the register file. The flip-flops are enabled by a write enable signal and gated by the system clock. Thus are the master latch of a flip-flop only transparent when the value of the write enable signal corresponds to said flip-flop at the positive edge of the clock.

Two multiplexers controlled by each output ports selection signal directs which registers that are connected to the output ports at all time. In this simple figure does one register only consist of one flip-flop. Each flip-flop is connected to *rstn*, a reset signal which when invoked will set all registers in the register file to zero.

The standard flip-flop register file of 32x32 bits has two 32 input data, *rd* and *rd2*. They are multiplexed to choose which one that is written to the register file. Otherwise are the architecture similar to Figure 4.1, except that there are 32 rows of registers, and the registers consist of 32 flip-flops each.

4.3 Flip-flop design with one-hot encoded selection signals

The architecture of the register file is slightly altered in this architecture to enable one-hot encoded selection logic. This change does not affect the functionality and architecture of the register file itself, only on the setup of the module, and in the selection logic. Figure 4.2 shows how the register file module is built in this design. The four selection signals, two for the write ports and two for the read ports, are changed from 5-bit signals to 32-bit one-hot encoded signals. The register file itself is moved to its own module, with similar signals and functionality as the prior register file setup.

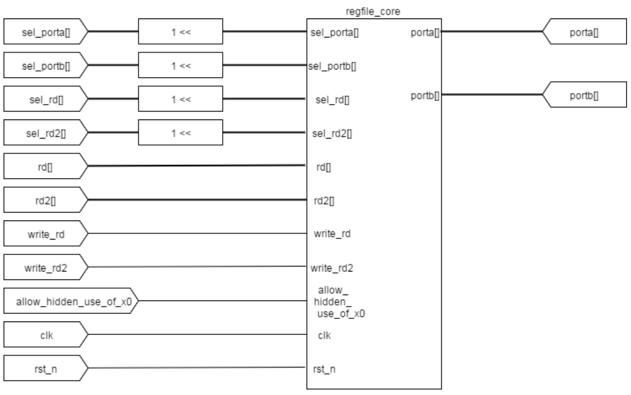


Figure 4.2: Architecture of the register file in the one-hot encoded design

4.4 Shared Master-latch architecture

In this architecture the flip-flops is changed to latches. It takes use of the principle of the master-slave flip-flops by having one master latch for all the slaves in a register. This is more thoroughly described in Section 2.2.2.1.

Figure 4.3 shows the architecture of a two read, one write 3x1-bit register file. The design for the system are of course 32x32-bits, but Figure 4.3 is to show the functionality of a shared master latch register file. The data is first read to the master latch, which is transparent when the clock is low, therefor gated by an inverted clock. The master latch is also enabled by a write enable signal, to only store values when the input data is valid. The data propagates further from the master latch to the slave latches. The slave latches are transparent when the clock is high, and the correct selection logic is valid. The latches are transparent when the corresponding selection bit in the one-hot selection signal, *sel_data[]* in Figure 4.3, are high, as well as the delayed write enable signal are high. The selection signal are also a delayed version of the selection signal from the input.

The selection signals are delayed with latches that are clock-gated by the inverted clock, to make sure that the selection logic for the slave latches are moved in time to be valid for the slave latches as well as the master latches. The output ports are multiplexed by two selection signals, enabling it to connect to the correct slave latch.

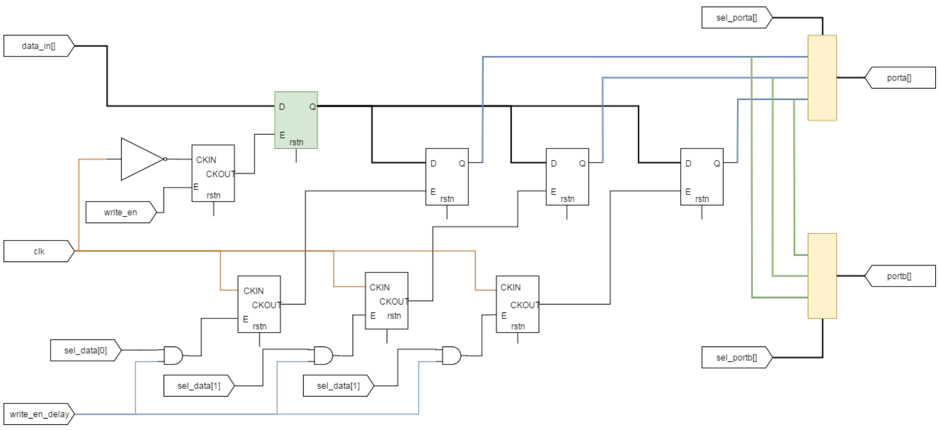


Figure 4.3: Shared Master-latch architecture

The theory in Section 2.2.2.1 states that a shared master latch architecture will produce $n + m^2$ latches, thus making a system with a 32*32-bit register file needing $32 + 32^2 = 1056$ latches. The system does, however, have two input ports. This makes it necessary to have two master latches, one for each input port. With this addition in mind, the equation becomes $2 * m + n^2$, and the total number of latches in design becomes then $2 * 32 + 32^2 = 1088$.

Except for the extra input port in the system, the architecture for the Shared Master-latch design is similar to the system described with Figure 4.3. The number of master-latches and slave-latchess are greater, but the principles are the same.

Figure 4.4 shows a single read and write operation for the shared master-latch design. The data-in port *rd* gets the value 2001000 while the clock is low, an added delay to ensure that the system handles physical delay. As seen in the figure exists a delayed version of the write enable signal, *write_rd*, and the selection signal, *sel_rd* both with the suffix *_tmp*.

First, is the data stored in *regfile_temp*, the master latch for the read port *rd*. This master-latch is transparent when the clock is low and *write_rd* is high, as it is when rd gets it value. The value is passed onto the slave latches when the clock gets high, as it is transparent only when the clock is high. Since the *sel_rd_tmp* only is enabling register x2, the value is only passed to that register. The output port *porta* immediately gets the value from register x2 as the outputs of the slave registers are multiplexed to the output ports, and the selection signal controlling the output *porta*, *sel_porta*, is set to register x2.

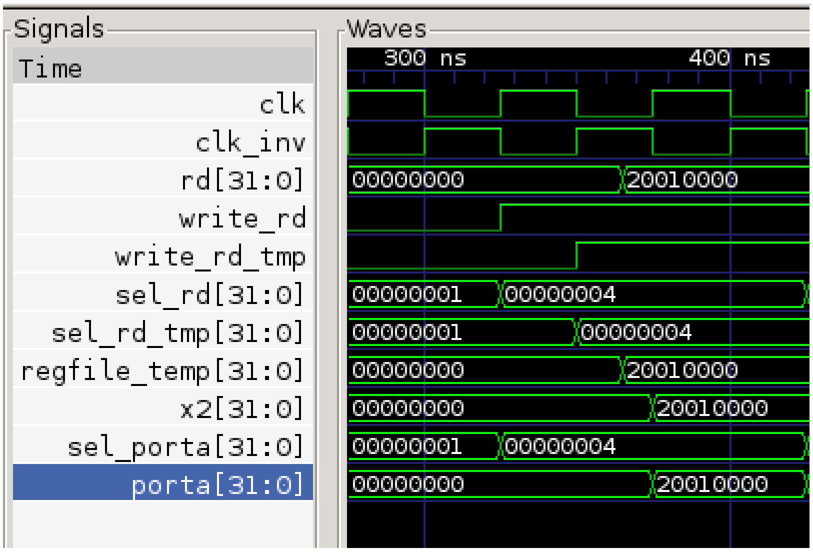


Figure 4.4: Write and Read operation for Shared Master-latch architecture

4.5 The Shared Slave-latch architecture

The Shared Slave-latch architecture is, as the Shared Master-latch architecture, based on latches. The mayor difference between the designs are that there are latches on the output on the Shared Slave-latch implementation, and on the input of the Shared Master-latch implementation. The theory of a Shared Slave-latch register is described in Section 2.2.2.2. This section will on the other hand, present how the Shared Slave-latch register file implemented for the system work.

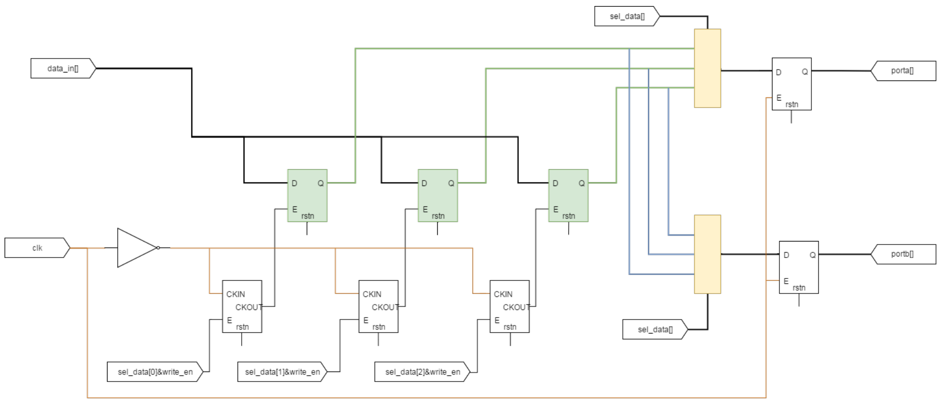


Figure 4.5: Shared Slave-latch architecture

Figure 4.5 shows an example register file based on the Shared Slave-latch approach. This architecture is a 3x1-bit register file, with two write ports and one read port. The theory states that a Shared Slave-latch register file needs $n + n * m$, giving $1 + 3 * 1 = 4$ latches in total. As seen in the figure, there are five latches. This is because the architecture has two read ports, thus giving two sets of slave latches to keep the value for each output port.

The register file works by first reading the data into the master latches. The master latches are only transparent when the clock is low, and are all enabled by write logic. The write logic consist of a mutual write enable signal, *write_en*, and a designated bit of the data select signal, *sel_data[]*. The data select signal is one-hot encoded which asserts that only one master latch is transparent at the time. The output of the master latches are multiplexed by two selection signals, one for each read port. The data is then propagating to the slave latches, which are read to the output ports.

As for the Shared Slave-latch register file implemented for the system, are the basics the same as described in the last paragraph, except that the number of

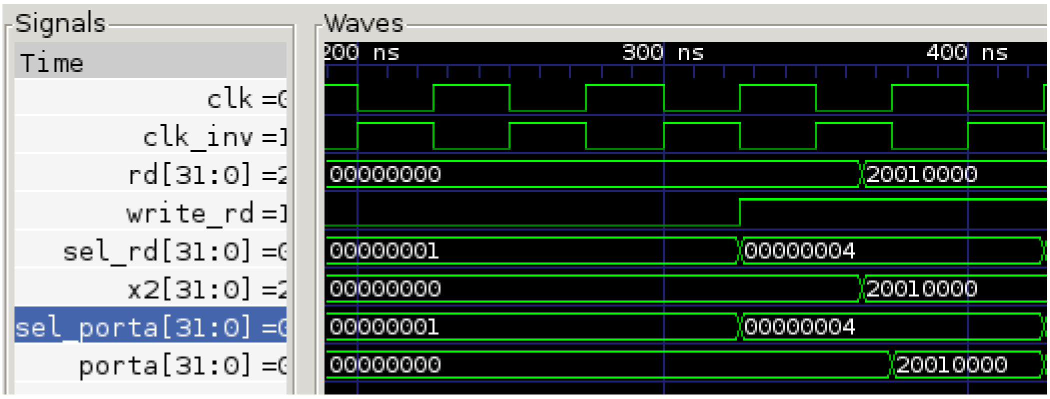


Figure 4.6: Write and Read operation for Shared Slave-latch architecture

bits is greater, and that there are two input ports instead of the one in Figure 4.5. The extra input port does not affect the architecture in terms of the number of latches, only the selection logic is a bit different. First of all, the slave latches are now transparent when the corresponding bit in the selection signal to either ports are high. A multiplexer chooses between the input from the ports based on the selection signals.

Figure 4.6 shows a single write and read operation of the Shared-Slave latch implementation in the system environment. Data appears on the input port *rd*. A delay are added to the data to simulate possible delay in the physical data. Since both the write enable signal, *write_rd*, and the clock is low, the value is immediately passed to the master registers since the master latches are transparent. The value is passed to register x2, since the selection signal is enabling that register. When the clock goes high, the master latches are blocking, the slave registers are transparent and the value from x2 passes to the output port *porta*. This is done as the selection signal for *porta* is pointing at x2, selecting the output of the x2 latch-based register.

Chapter 5

Implementation

In this chapter will the implementation of the project be presented and discussed. It focuses on the actual RTL implementation in Verilog, where Chapter 4 looks at the architectures that are implemented. The architectures implemented are the original design with modified selection logic, one-hot encoded, a shared master-latch design and a shared slave-latch design. The original design implementation is presented as well. Furthermore is the script made to generate the relative placement directives presented in this chapter. The script is implemented in python and generates relative placement directives and a graphical representation on how the cells are to be placed.

5.1 The Original register file design

The original register file is implemented in Verilog. It is implemented as a module with two data inputs and two data outputs, with additional selection logic as inputs. The data inputs and outputs are 32 bits wide. The actual register file is declared as a 32x32 bit signal, named *regfile.*

```
generate
   for (i = 0; i < 32 ; i = i + 1) begin : loop
   always @(posedge clk or negedge rst_n) begin
      if (~rst_n)
      regfile[i] <= 32'h0;
      else
        if(((sel_rd != 0) || allow_hidden_use_of_x0) && (sel_rd == i
            [RF_PORTRD_MSB:0]) & write_rd) begin
      regfile[i] <= rd;
    end
        else if (((sel_rd2 != 0) || allow_hidden_use_of_x0) && (
            sel_rd2 == i[RF_PORTRD_MSB:0]) && write_rd2) begin
      regfile[i] <= rd2;
    end
   end
  end
endgenerate
```

Listing 5.1: Write process for the original register file

The process of writing to the register file is implemented as in Listing 5.2. The generate-for statement makes it possible to iterate over all the registers of the register file. Inside the generate for is each register assigned a value. The reset mechanism are first implemented, as an asynchronous reset toggled by the negative edge *rst_n*, which sets every register to zero if active. An if-else statements controls which input to read from, and whether to write at all. This if statement is a part of the positive clock-edged part of the process. The selection logic in the if-else are two selection signals and two write enables, one for each input. The selection signals checks whether the current position in the register file is to be written to. The *allow_hidden_use_of_x0* is a signal that enables the use of the 0th register under certain, special circumstances.

```
  always @(sel_porta or regfile[sel_porta] or
      allow_hidden_use_of_x0) begin
    if((sel_porta != 0) || allow_hidden_use_of_x0) begin
     porta = regfile[sel_porta];
   end
   else begin
     porta = 0;
   end
end

  always @(sel_portb or regfile[sel_portb] or
      allow_hidden_use_of_x0) begin
    if((sel_portb != 0) || allow_hidden_use_of_x0) begin
     portb = regfile[sel_portb];
   end
   else begin
     portb = 0;
   end
end
```

Listing 5.2: Read process for the original register file

Listing 5.2 shows the two processes controlling the reading from the register file. Each process controls one output port. The output port is read to from the regfile, from the position of the value to the selection signal. Unless the selection signal is zero and *allow_hidden_use_of_x0* are low, which set the output port to zero.

5.2 Standard Flip-flop design with one-hot encoded selection logic

This approach splits the register file into two modules. The selection logic in the original register file module, and the actual register file is moved to a module called the register file core. The main purpose of the selection logic module is to transform the selection signals from regular selection logic, i.e. only values, to one-hot encoded signals. That is done as in Listing 5.3.

```
input  [RF_PORTA_MSB:0]  sel_porta;
input [RF_PORTB_MSB:0]   sel_portb;
input [RF_PORTRD_MSB:0]  sel_rd;
input [RF_PORTRD_MSB:0]  sel_rd2;

wire [DATA_MSB:0]          sel_porta_oh;
wire [DATA_MSB:0]          sel_portb_oh;
wire [DATA_MSB:0]          sel_rd_oh;
wire [DATA_MSB:0]          sel_rd2_oh;

assign sel_porta_oh = (sel_porta == 0) ? 1 : (1'b1 << sel_porta);
assign sel_portb_oh = (sel_portb == 0) ? 1 : (1'b1 << sel_portb);
assign sel_rd_oh   = (sel_rd == 0) ? 1 : (1'b1 << sel_rd);
assign sel_rd2_oh = (sel_rd2 == 0) ? 1 : (1'b1 << sel_rd2);
```

Listing 5.3: Selection logic transformation

The values in the five-bit signals are translated to 32-bit one-hot encoded signals. One-hot encoded signals are only zeroes, except for the position in which corresponds with the value. For example, 110 in binary (6) is 01000000 one-hot encoded, where only the 6th bit is high. This lowers the switching activity of a transition from one value to another, as only two bits changes value.

The write process of the register file is simplified in this design. This is because the selection signals are now one-hot encoded, which makes it possible to change the if-conditions from

```
if(((sel_rd != 0) || allow_hidden_use_of_x0) && (sel_rd == i[
    RF_PORTRD_MSB:0]) & write_rd) begin
```

and

```
else if (((sel_rd2 != 0) || allow_hidden_use_of_x0) && (sel_rd2 == i
    [RF_PORTRD_MSB:0]) && write_rd2) begin
```

to

```
if((sel_rd[i] & write_rd & sel_rd !=0) & (sel_rd[0] != 1 |
    allow_hidden_use_of_x0)) begin
```

and

```
else if((sel_rd2[i] & write_rd2 & sel_rd2 != 0) & (sel_rd2[0] != 1 |
    allow_hidden_use_of_x0)) begin
```

Otherwise is the write process the same as in the original design. The read process, however, is quite different. Because the selection signals of the outputs are one-hot encoded, it is not possible to use *regfile[sel_porta]* or *regfile[sel_portb]*. *Porta* and *portb* are the output ports. Listing 5.4 shows how data from the register file is read to the output port *portb*. Note that the code listing is shortened for visual purposes.

```
assign portb = (sel_portb[0] != 1) || allow_hidden_use_of_x0 ?
                        ({32{sel_porta[0]}}& regfile[0]) |
                        ({32{sel_portb[1]}}& regfile[1]) |
                        ({32{sel_portb[2]}}& regfile[2]) |
                        ..
                        ({32{sel_portb[30]}}& regfile[30]) |
                        ({32{sel_portb[31]}}& regfile[31]) : 0 ;
```

Listing 5.4: Read process of the One-hot encoded design

5.3 Shared Master-latch design

This architecture scratches the standard flip-flop design and use latches instead of flip-flops. The shared master latch register file theory are described in Section 2.2.2.1. The design is a continuation of the one-hot encoded flip-flop design as the selection logic is still one-hot encoded. The architecture of the register file is changed however. Since this design is based on latches are the processes not clocked. To implement the master latch, two processes reads in data to two temporary registers,*regfile_temp* and *regfile_temp2* as shown in Listing 5.5. The processes are clock-gated and enabled by the two write enable signals.

```
always @(*) begin
   if (write_rd&clk_inv) begin
      regfile_temp <=rd;
   end
end // always @ begin

always @(*) begin
   if(write_rd2&clk_inv) begin
      regfile_temp2 = rd2;
   end
end // always @ begin
```

Listing 5.5: Master latch implementation

The slave latches are implemented with a process and a generate-for. The generate-for are present to iterate through all the slave latch registers, *regfile[0]* to *regfile[31]*. The process can be seen in Listing 5.6.

```
generate
    for (i=0; i<32; i=i+1) begin : loop
        always @(*) begin
            if(~rst_n) begin
                regfile[i] = 32'h0;
            end //if
            else begin
                if( (sel_rd_tmp[i]&write_rd_tmp&clk) & (sel_rd_tmp
                    [0]!= 1 | allow_hidden_use_of_x0) ) begin
                    regfile[i] = regfile_temp;
                end //if
                else if( (sel_rd2_tmp[i]&write_rd2_tmp&clk) & (
                    sel_rd2_tmp[0]!=1 | allow_hidden_use_of_x0)     )
                    begin
                    regfile[i] = regfile_temp2;
                end //elseif
            end //else
        end //always
    end //for
endgenerate
```

Listing 5.6: Slave latch implementation

Note that the slave latches are clock gated with *clock_inv*, the system clock inverted. This is to ensure that the master latch and the slave latches are not transparent at the same time, ensuring reliable data storage. Which slave that is written to is selected by the one-hot encoded signals *sel_rd_tmp* and *sel_rd2_-tmp*, both signals enabled by write enable signals. The selection signals and the write enable signals are delayed, using latches described by the process in Listing 5.7. This is done to ensure that the signals has the correct value as the data is moved from the master latches to the slave latches.

```
always @(*) begin
    if(clk_inv) begin
        sel_rd_tmp <= sel_rd;
        sel_rd2_tmp <= sel_rd2;
        write_rd_tmp <= write_rd;
        write_rd2_tmp <= write_rd2;
    end
end
```

Listing 5.7: Delay of selection signals

The reading process of the shared master latch architecture is implemented similarly as the one-hot encoded flip-flop design.

5.4 Shared Slave-latch design

The Shared Slave-latch design is closely related to the Shared Master-latch design, and the implementations are similar in many ways. Both implementations are based on splitting up the master-slave latches, and sharing one of them. The case with the Shared Slave-latch architecture is that the slaves are shared, and thus putting the latches on the output, rather than on the input as in the Shared Master-latch design. The Shared Slave-latch register file theory are described in Section 2.2.2.2. As in the Shared Master-latch design it is still based on the one-hot encoded selection logic implemented for the One-hot encoded design. The input data are read directly into the master latches, using a similar generate-for process as the slave latch implementation of the Shared Master-latch approach. This process can be seen in Listing 5.8.

```
  generate
       for (i=0; i<32; i=i+1) begin : loop
    always @(*) begin
       if(~rst_n) begin
          regfile[i] = 32'h0;
       end //if
       else begin
          if( (sel_rd[i]&write_rd&clk_inv) & (allow_hidden_use_of_x0
              | !sel_rd[0] )) begin
                 regfile[i] = rd;

          end //if
          else if( (sel_rd2[i]&write_rd2&clk_inv) & (
              allow_hidden_use_of_x0 | !sel_rd2[0])) begin
                 regfile[i] = rd2;

          end //elseif
       end //else
          end //always
       end //for
 endgenerate
```

Listing 5.8: Master latches implementation

The read process are changed from the Shared Master-latch design and the One-hot encoded design. It is here that the slave latches are implemented. The output ports are thus implemented in a non-clocked process, and gated by the clock. This process can be seen in Listing 5.9 where the latches on the output port *porta* are implemented.

```
always @(*) begin
        if(clk) begin
        porta <= (sel_porta[0] != 1) | allow_hidden_use_of_x0 ?
({32{sel_porta[0]}}& regfile[0]) |
({32{sel_porta[1]}}& regfile[1]) |
({32{sel_porta[2]}}& regfile[2]) |
...
({32{sel_porta[30]}}& regfile[30]) |
({32{sel_porta[31]}}& regfile[31]) : 0;
end
```

Listing 5.9: Putting latches on the output port *porta*

5.5 Pyplace - Relative placement script

Relative placement is described in Section 2.1. A script, Pyplace, is a part of the thesis made to generate the directives described in Section 2.1. The purpose of Pyplace is to shorten the process of generating the relative placement directives, with the relative placement groups and relative positions of cells, for the Route & Place analysis. The script is constructed of different class types, for different relative place groups. The standard Cell construct consist of a single cell that contains information on height, width, name, horizontal position and vertical position. A cell also contains member functions on relative placement, which enables placing a cell relative to another cell by modifying the positional variables. Furthermore it is a Bit class which is made of two objects of the Cell type, representing a flip-flop and a multiplexer. 32 objects of the Bit type are made to the next tier class, the Register class. The first bit is set to the origin, x-position zero and y-position zero.

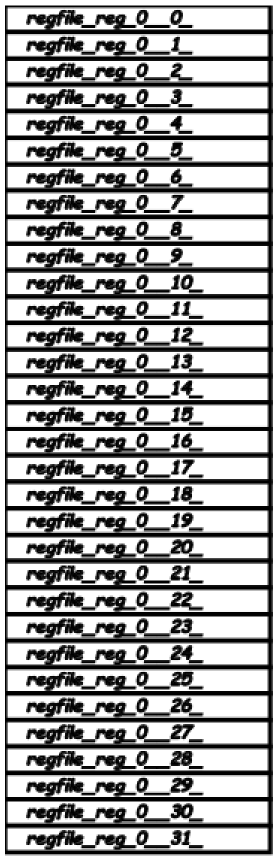


Figure 5.1: Relative placed flip-flops graphically presented

The other 31 bit objects are placed beneath the first bit. Two member functions are implemented for the Register class, printBits and writeRpGroup. Both functions prints the placement of the flip-flops of the bits, printBits prints it as a svg picture and writeRpGroup writes to the relative placement directive tcl file. Figure 5.1 shows the picture generated from printBits and Listing 5.11 shows the directives created by writeRpGroup, altough both are constricted to only show the first register of the register file, not the whole register file.

The multiplexers from the bits in the Register class are dropped from both the generated picture and the relative placement directives. This is done because the

ICC needs the exact reference name of the cells to be grouped together. Those reference name follows a certain pattern for the flip-flops, regfile_reg_x___x_, but the connecting multiplexers (and other cells) are named randomly and thus making it hard to connect them to the flip-flops.

Listing 5.11 shows the relative placement for a single register of the register file. To create the relative placement group for the whole register file, a simple for loop which iterates through all the registers is made. The command to create the relative placement group is made outside the for-loop. The process of creating the relative placement directives can be seen in Listing 5.10.

```
f = open('test.tcl','w')
f.write("create_rp_group rp1 -design pil -columns 32 -rows 32 \n")
rp_group_str = "add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/"
temp = Register(0,0)
temp.printBits()
for i in range(0,32):
        temp = Register(i,i)
        temp.writeRpGroup(f,rp_group_str)
        f.write("\n")
f.close()
```

Listing 5.10: Relative placement directive creation

```
create_rp_group rp1 -design pil -columns 32 -rows 32
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___0_ -column 0 -row 0
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___1_ -column 0 -row 1
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___2_ -column 0 -row 2
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___3_ -column 0 -row 3
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___4_ -column 0 -row 4
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___5_ -column 0 -row 5
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___6_ -column 0 -row 6
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___7_ -column 0 -row 7
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___8_ -column 0 -row 8
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___9_ -column 0 -row 9
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___10_ -column 0 -row 10
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___11_ -column 0 -row 11
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___12_ -column 0 -row 12
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___13_ -column 0 -row 13
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___14_ -column 0 -row 14
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___15_ -column 0 -row 15
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___16_ -column 0 -row 16
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___17_ -column 0 -row 17
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___18_ -column 0 -row 18
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___19_ -column 0 -row 19
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___20_ -column 0 -row 20
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___21_ -column 0 -row 21
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___22_ -column 0 -row 22
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___23_ -column 0 -row 23
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___24_ -column 0 -row 24
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___25_ -column 0 -row 25
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___26_ -column 0 -row 26
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___27_ -column 0 -row 27
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___28_ -column 0 -row 28
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___29_ -column 0 -row 29
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___30_ -column 0 -row 30
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___31_ -column 0 -row 31
```

Listing 5.11: Relative placement directives

Chapter 6

Experiments and Results

In this chapter will the experimental setup and results from the thesis be presented. The chapter is divided into Experimental Plan in Section 6.1, Experimental Setup in Section 6.2, Verification Results in Section 6.3, SpyGlass Power Estimation in Section 6.4, Synthesis Area Results in Section 6.5, Synthesis Timing Results in Section 6.6 and lastly results from the Placement analysis in Section 6.7. The experimental plan describes the plan for the experiments in the thesis. The experimental setup discuss the setup required to do the experiments. The last sections presents the results from the research done in this thesis, from Verification, Synthesis, Power Estimation and Place & Route analysis.

6.1 Experimental Plan

To test whether a solution is better or worse than a different solutions, a set of objective, comparable data is needed. For a microchip system as used in this thesis this could be the systems power consumption, area usage or timing. Verification are important to assert the validity of a new architecture. It is no new functionality of the alternate register file architectures, they have to be functionally equivalent to the original. By running the same testbenches on the alternate designs as the original, this is assured given that the same results from the verification are present. This section will focus on how the different systems are verified and how the power, area and timing data is collected, through synthesis, power estimation and place & route tools described in Chapter 3.

6.1.1 Verification

Verification are an important mean to ensure that a systems functionality are working according to its specifications. For the system the register file are implemented on, exists a number of testes and testbenches to accomplish this. The verification are run on the whole CPU. This ensures that the changes in the register file does not affect the rest of the system badly, and the register file are tested in the environment it is designed for. The testes contains verification for the register file level as well. This throughout verification makes sure the alternate design techniques works as the original register file, and makes it valid to compare them to the original design approach.

Initially are a python script named Runtest.py used. Runtest.py are a Python script that are used to launch simulations using different simulators. In this project was the Icarus Verilog simulator used. The Runtest.py initially launches a quick simulation that runs testbenches that tests the functionality of the system. With regard to the register file, stimuli are generated, reading and writing data to and from the register file. The data read from the register file is asserted with known values to check that register file works correctly.

The simulation generates a waveform that can be read to look at the behavior of each signal at each hierarchical level of the design. This waveform is a important mean to debug issues with new architectures by comparing the signal behavior with the known, working original design. The generated waveform is later used in the SpyGlass Power Estimate to calculate the dynamic power of the system, as it contains information about the switching activity of the system.

After an architecture is initially verified is further testing applied. The quick simulation launched from Runtest.py is initially used as the test-runtime is small and it gives an initial indication that system is working. However, a set of torture testes are applied to the system CPU to test the system under more difficult circumstances. The torture testes runs in both the icarus and the pysim environment, and runs a checker script to assert that the data written and read to the register file during the torture simulation on both the simulation environments are correct. The torture testes are issued with the Runtest.py simulation launcher as the initial quick simulation was. The stress level are possible to set before a run, and can be set to 10, 50 or 100, depending on the level of stress issued to the system. To summarise are the torture testes a large set of stress testes pushing the system to its limits, asserting the data afterwards. To make absolutely sure that the system works perfectly, a torture test run with a stress level of 100 is issued, and all the testes should pass.

Too further strengthen the validity of the verification are a delay added to the input data. The input data goes through calculation and different operations before it propagates to the register file. In a physical world, this data does not always arrive perfectly at the rising edge of the system clock. To simulate this physical delay, an delay of 75 percent of the clock period are added to both the data inputs in the module a hierarchical level above the register file. All the testes are again run with this added delay, both the initial quick simulation and the torture testes, and as the original testes does this have to pass to ensure that the alternate architectures works as the original design.

6.1.2 Synthesis

Synthesis are done to translate a design at register-transfer level to a design implementation in terms of logic gates. At this abstraction level, it iss possible to run analysis on power consumption, area usage and timing. In this thesis is the tool DC used, which are described further in Chapter 3. DC is in this thesis used to extract the area usage information of the register file of the alternate architectural optmisations. Timing analysis is also run, using an existing set of timing constraints given for the original system environment. The synthesis of the design generates a netlist for the complete system. This netlist is later used to run both Power Estimation with SpyGlass Power Estimate and in the Place & Route analysis.

6.1.3 Power Estimation

The power consumption is an important measurement for the success of an architectural optimisation scheme. The DC produces a power report after synthesis, but SpyGlass Power Estimate are used instead to estimate the power consumption. This is because the SpyGlass Power Estimate is specialised to estimate the power consumption in a system such as the system the register file are implemented on. The SpyGlass Power Estimate uses the netlist generated by synthesis and a waveform generated from a simulation launched by the Runtest.py launcher. The inclusion of the waveform makes the analysis of the dynamic power consumption more precise as the switching and toggling of signals and nets are important to measure the dynamic power. This is not a part of the power analysis with the DC, which makes the SpyGlass Power Estimate a better option to calculate the power consumption.

6.1.4 Place & Route

Section 3.0.2 describes the tool used for Place & Route. This is done on the relative placement part of the thesis. The ICC produces reports on power consumption, area usage and timing. A schematic is produced, and this shows graphically how the relative placement directives affects the system. By keeping the graphical representation in mind, the reports can be analyzed and compared to check the success of the relative placement. The ICC uses the netlist produced by the synthesis as a base for analysis.

6.2 Experimental Setup

This section describes the necessary setup to run the verification, synthesis and Place & Route described in Section 6.1. The verification are set up for the Original design. Since the changes applied to the register file are to split it into two modules, these new files has to be added to the file list in the simulation setup. The testbench are using variables and signals directly from the register file, and the path to this signals has to be changed. Otherwise are the setup easy as there are possible to swap between the different register file architectures, running om the same environment. The changes applied when adding the delay to the testes were added in the module one step above the register file in the hierarchy, and thus applicable to all the different architectures.

Much of the same were the case for the setup for the synthesis. An existing environment to run the system with DC was used, and only small changes were applied to run the new register file architectures. The register file core had to be added in the file list, otherwise were the parameters and setup the same as given by the environment for the original system and register file.

The place & route analysis did require more setup. First, to enable relative placement, the relative placement directives had to be sourced from the tcl script that sets up the ICC. Commands to the ICC to force the optmisation to not touch the relative placement groups had to be added as well. Lastly, since the ICC uses the netlist generated by synthesis, synthesis analysis has to be done first, and the ICC setup script has to point to the correct netlist.

The SpyGlass Power Estimate tool uses the netlist from synthesis as well. The waveform from simulation has to be added to the SpyGlass directory, and setup in terms of start and end time has to be set up before the power estimation analysis are issued. Like in the other tools had the file list to be edited to include all the relevant modules used by the system. Otherwise it were possible to use the already existing SpyGlass environment set up for the system.

6.3 Verification Results

Here is the results from the verification part of the thesis presented. Verification is done on the whole core of the system and is done for all the different architectures implemented in this project. The verification plan are described in Section 6.1.1. The three modified register files were all run through the different testes as described in the verification plan. Firstly did the One-hot encoded design pass all the testes, torture testes and testes with added delay without any issues. This is because the One-hot encoded design does not affect the architecture of the register file itself, only the surrounding selection logic, making it relatively easy to make it functionally equivalent as the original design.

The Shared Slave-latch approach was the second design implemented and did deem more troublesome to pass the testes. The initial test did work right away, but the torture testes deemed to difficult for the design. Changes were thus made to make it pass, more specifically did the process of the master latches have to be split into two separate processes.

Then did the design have to pass the testes with delayed input data, which it did not initially. To cope with the delay were the delayed selection signals added to the design, as the delayed data showed the need for this. The design now passed the delayed testes, in addition to the earlier mentioned testes, and as far as the verification goes, is considered functionally equivalent as the original design. As the Shared Slave-latch design were the last implemented and closely related to the Shared Master-latch design, did the design process go streamlined and it passed all the testes right away.

6.4 SpyGlass Power Estimation Results

This section presents the results from the power estimation of the different architectural approaches explored in this thesis. These results are generated from running the different designs with the SpyGlass Power Estimate. The SpyGlass Power Estimate are described in Section 3.0.3, were the calculation of the different sources of power consumption are described. The power results are presented in tables, which splits the power sources into leakage power, internal power and switching power, and a column that shows the total power consumption by each source. These results are a vital part of evaluating and comparing the architectures. The power estimation results are from the original register file architecture, as well as the alternate architectures, the One-hot encoded design, the Shared Master-latch design and the Shared Slave-latch design. The reports shows the reports from the whole system, and the power consumption of just the register file.

	Leakage	**Internal**	**Switching**	**Total**	
Total Power	2.86E-02	105.495	68.833	174.510	μW
Combinational Power	1.37E-02	22.364	55.938	78.316	μW
Sequential Power	5.03E-02	60.065	1.507	61.586	μW
Black Box Power	0	0	0	0	μW
Memory Power	0	0	0	0	μW
IO PAD Power	0	0	0	0	μW
Clock Power	5.15E-04	22.220	11.388	34.608	μW

Table 6.1: Power results for the Original design

The power consumption estimates for the Original design are shown in Table 6.1.

	Leakage	**Internal**	**Switching**	**Total**	
Total Power	2.83E-02	99.615	66.444	164.087	μW
Combinational Power	1.40E-02	16.921	51.682	68.616	μW
Sequential Power	1.37E-02	59.492	1.417	60.922	μW
Black Box Power	0	0	0	0	μW
Memory Power	0	0	0	0	μW
IO PAD Power	0	0	0	0	μW
Clock Power	5.15E-04	22.203	11.345	34.548	μW

Table 6.2: Power results for the One-hot encoded design

Table 6.2 are the power estimated for the One-hot encoded design.

	Leakage	**Internal**	**Switching**	**Total**	
Total Power	2.54E-02	97.592	62.371	159.989	μW
Combinational Power	1.36E-02	15.820	48.691	64.524	μW
Sequential Power	1.13E-02	61.924	1.452	63.387	μW
Black Box Power	0	0	0	0	μW
Memory Power	0	0	0	0	μW
IO PAD Power	0	0	0	0	μW
Clock Power	4.84E-04	19.848	12.229	32.078	μW

Table 6.3: Power results for the Shared Master-latch design

The Shared Master-latch design has power consumption characteristics as present in Table 6.3.

	Leakage	**Internal**	**Switching**	**Total**	
Total Power	2.54E-02	99.061	64.368	163.454	μW
Combinational Power	1.38E-02	18.197	50.968	69.179	μW
Sequential Power	1.11E-02	61.041	1.516	62.567	μW
Black Box Power	0	0	0	0	μW
Memory Power	0	0	0	0	μW
IO PAD Power	0	0	0	0	μW
Clock Power	4.83E-04	19.823	11.884	31.708	μW

Table 6.4: Power results for the Shared Slave-latch design

The power estimation results for the Shared Slave-latch design can be found in Table 6.4.

	Leakage	**Internal**	**Switching**	**Total**	
Original	7.49E-03	16.206	9.283	25.497	μW
One-hot	7.17E-03	10.123	5.573	15.703	μW
Shared Master	4.29E-03	8.152	3.721	11.877	μW
Shared Slave	4.32E-03	9.619	4.828	13.576	μW

Table 6.5: Power Consumption in register file of the different designs

Table 6.5 shows how the differences in the different architectures directly affects the register file.

6.5 Synthesis Area Results

This section presents the area usage reports for each design of the architectural optimisation exploration. The reports are generated by synthesis run by DC and shows the number of cells used in the different register files, as well as the area used by these cells.

	Number of cells	**Area**
Combinational logic	5670	8375.7
Flip-flops	1024	6881.28
Clock gating	32	170.24
Total	6726	15427.22

Table 6.6: Area usage of the register file of the Original design

Table 6.6 shows the area usage of the register file of the Original design. The data is divided into combinational logic, flip-flops and clock-gating logic, and the total area usage and the total number of cells of the register file.

	Number of cells	**Area**
Combinational logic	6403	9402.6
Flip-flops	1024	6881.28
Clock gating	32	170.24
Total	7459	16454.12

Table 6.7: Area usage of the register file of the One-hot encoded design

This table, Table 6.7, shows the area usage and the number of cells of the One-hot encoded design. The data is divided similarly as the Original design as the architecture is similar, except for additional combinational logic requird for the one-hot encoding.

	Number of cells	**Area**
Combinational logic	5429	9346.96
Master latches	64	250.88
Slave latches	1024	4014.08
Latches for delay logic	66	277.2
Total	6583	13889.12

Table 6.8: Area usage of the register file of the Shared Master-latch design

Table 6.8 are the area results from the Shared Master-latch design. It shows both the number of cells and the area usage of the register file. Here are the data for the master latches and the slave latches shown, as well as the data in the Original design and the One-hot encoded design, except for the clock gating logic as this does not appear in the Shared Master-latch design. The latches used to delay some of the selection logic are present as well in the table.

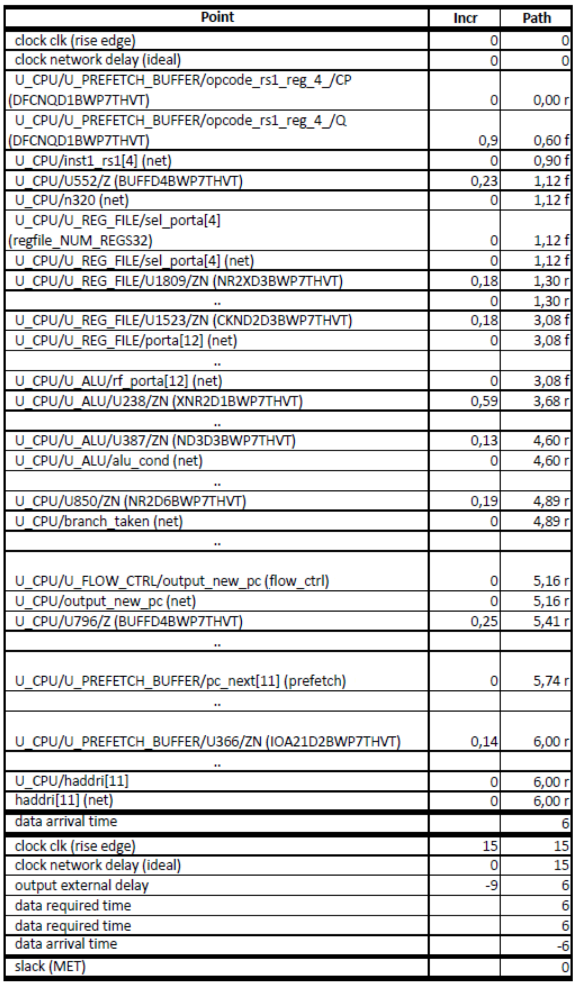
	Number of cells	**Area**
Combinational logic	5362	9012.8
Master latches	1024	4014.08
Slave latches	64	250.88
Total	6450	13277.76

Table 6.9: Area usage of the register file of the Shared Slave-latch design

The area results from the Shared-Slave latch design are shown in Figure 6.9. These results contain the same information as the Shared Master-latch results, except for the delay logic as this are not present in the Shared Slave-latch design.

6.6 Synthesis Timing Results

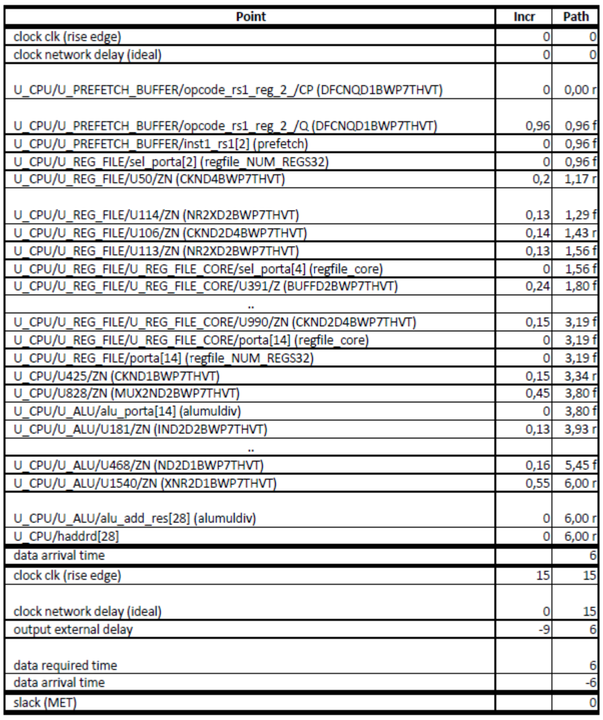
This subsection presents the Timing reports generated by the DC. The timing analysis are done post-synthesis and the reports presented in this section are chosen to include timing reports that traverse through the register file for accurate measurements on how the alternate register files affects the timing of the system. Figure 6.1 shows a timing report for the Original design. The path starts in the



Point	Incr	Path
clock clk (rise edge)	0	0
clock network delay (ideal)	0	0
U_CPU/U_PREFETCH_BUFFER/opcode_rs1_reg_4_/CP (DFCNQD1BWP7THVT)	0	0,00 r
U_CPU/U_PREFETCH_BUFFER/opcode_rs1_reg_4_/Q (DFCNQD1BWP7THVT)	0,9	0,60 f
U_CPU/inst1_rs1[4] (net)	0	0,90 f
U_CPU/U552/Z (BUFFD4BWP7THVT)	0,23	1,12 f
U_CPU/n320 (net)	0	1,12 f
U_CPU/U_REG_FILE/sel_porta[4] (regfile_NUM_REGS32)	0	1,12 f
U_CPU/U_REG_FILE/sel_porta[4] (net)	0	1,12 f
U_CPU/U_REG_FILE/U1809/ZN (NR2XD3BWP7THVT)	0,18	1,30 r
..	0	1,30 r
U_CPU/U_REG_FILE/U1523/ZN (CKND2D3BWP7THVT)	0,18	3,08 f
U_CPU/U_REG_FILE/porta[12] (net)	0	3,08 f
..		
U_CPU/U_ALU/rf_porta[12] (net)	0	3,08 f
U_CPU/U_ALU/U238/ZN (XNR2D1BWP7THVT)	0,59	3,68 r
..		
U_CPU/U_ALU/U387/ZN (ND3D3BWP7THVT)	0,13	4,60 r
U_CPU/U_ALU/alu_cond (net)	0	4,60 r
..		
U_CPU/U850/ZN (NR2D6BWP7THVT)	0,19	4,89 r
U_CPU/branch_taken (net)	0	4,89 r
..		
U_CPU/U_FLOW_CTRL/output_new_pc (flow_ctrl)	0	5,16 r
U_CPU/output_new_pc (net)	0	5,16 r
U_CPU/U796/Z (BUFFD4BWP7THVT)	0,25	5,41 r
..		
U_CPU/U_PREFETCH_BUFFER/pc_next[11] (prefetch)	0	5,74 r
..		
U_CPU/U_PREFETCH_BUFFER/U366/ZN (IOA21D2BWP7THVT)	0,14	6,00 r
..		
U_CPU/haddri[11]	0	6,00 r
haddri[11] (net)	0	6,00 r
data arrival time		6
clock clk (rise edge)	15	15
clock network delay (ideal)	0	15
output external delay	-9	6
data required time		6
data required time		6
data arrival time		-6
slack (MET)		0

Figure 6.1: Timing report for the Original design

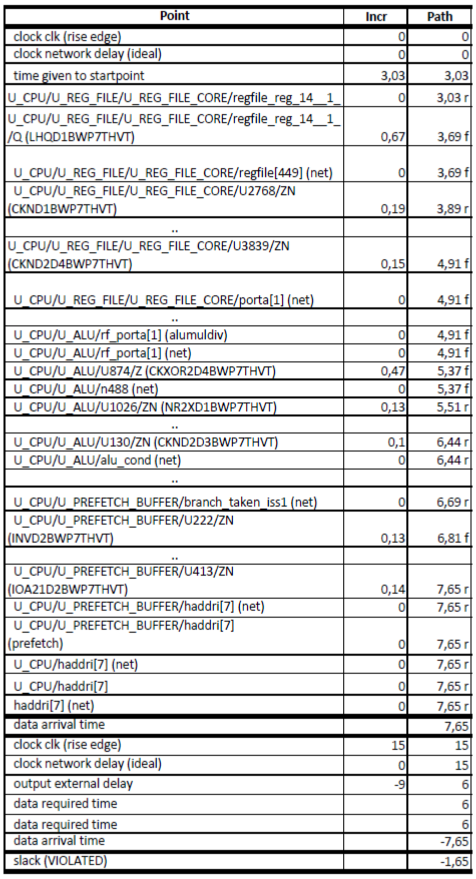
prefetch buffer and ends in the *haddri*, an output of the CPU. The path is going through the register file.



Point	Incr	Path
clock clk (rise edge)	0	0
clock network delay (ideal)	0	0
U_CPU/U_PREFETCH_BUFFER/opcode_rs1_reg_2_/CP (DFCNQD1BWP7THVT)	0	0,00 r
U_CPU/U_PREFETCH_BUFFER/opcode_rs1_reg_2_/Q (DFCNQD1BWP7THVT)	0,96	0,96 f
U_CPU/U_PREFETCH_BUFFER/inst1_rs1[2] (prefetch)	0	0,96 f
U_CPU/U_REG_FILE/sel_porta[2] (regfile_NUM_REGS32)	0	0,96 f
U_CPU/U_REG_FILE/U50/ZN (CKND4BWP7THVT)	0,2	1,17 r
U_CPU/U_REG_FILE/U114/ZN (NR2XD2BWP7THVT)	0,13	1,29 f
U_CPU/U_REG_FILE/U106/ZN (CKND2D4BWP7THVT)	0,14	1,43 r
U_CPU/U_REG_FILE/U113/ZN (NR2XD2BWP7THVT)	0,13	1,56 f
U_CPU/U_REG_FILE/U_REG_FILE_CORE/sel_porta[4] (regfile_core)	0	1,56 f
U_CPU/U_REG_FILE/U_REG_FILE_CORE/U391/Z (BUFFD2BWP7THVT)	0,24	1,80 f
..		
U_CPU/U_REG_FILE/U_REG_FILE_CORE/U990/ZN (CKND2D4BWP7THVT)	0,15	3,19 f
U_CPU/U_REG_FILE/U_REG_FILE_CORE/porta[14] (regfile_core)	0	3,19 f
U_CPU/U_REG_FILE/porta[14] (regfile_NUM_REGS32)	0	3,19 f
U_CPU/U425/ZN (CKND1BWP7THVT)	0,15	3,34 r
U_CPU/U828/ZN (MUX2ND2BWP7THVT)	0,45	3,80 f
U_CPU/U_ALU/alu_porta[14] (alumuldiv)	0	3,80 f
U_CPU/U_ALU/U181/ZN (IND2D2BWP7THVT)	0,13	3,93 r
..		
U_CPU/U_ALU/U468/ZN (ND2D1BWP7THVT)	0,16	5,45 f
U_CPU/U_ALU/U1540/ZN (XNR2D1BWP7THVT)	0,55	6,00 r
U_CPU/U_ALU/alu_add_res[28] (alumuldiv)	0	6,00 r
U_CPU/haddrd[28]	0	6,00 r
data arrival time		6
clock clk (rise edge)	15	15
clock network delay (ideal)	0	15
output external delay	-9	6
data required time		6
data arrival time		-6
slack (MET)		0

Figure 6.2: Timing report for the One-hot encoded design

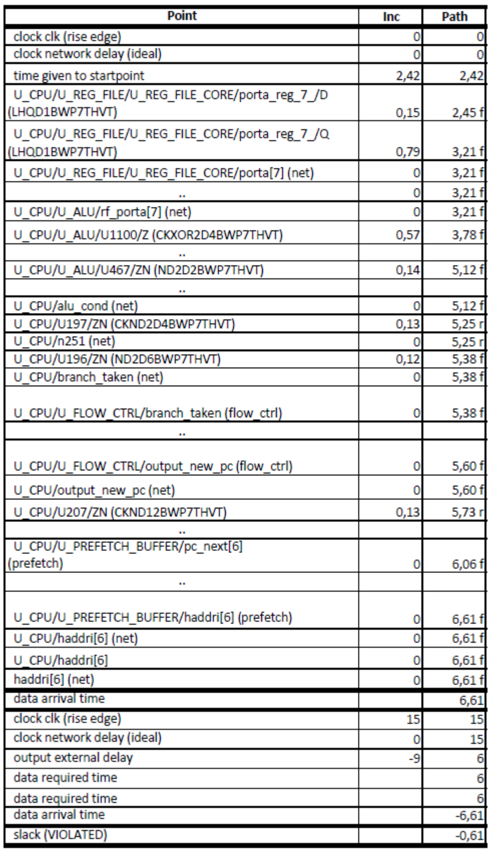
The Figure 6.2 shows a timing report for the One-hot encoded design. Here does the path starts in the register file core and end up in *haddrd*, which is also an output of the CPU.



Point	Incr	Path
clock clk (rise edge)	0	0
clock network delay (ideal)	0	0
time given to startpoint	3,03	3,03
U_CPU/U_REG_FILE/U_REG_FILE_CORE/regfile_reg_14__1_	0	3,03 r
U_CPU/U_REG_FILE/U_REG_FILE_CORE/regfile_reg_14__1_ /Q (LHQD1BWP7THVT)	0,67	3,69 f
U_CPU/U_REG_FILE/U_REG_FILE_CORE/regfile[449] (net)	0	3,69 f
U_CPU/U_REG_FILE/U_REG_FILE_CORE/U2768/ZN (CKND1BWP7THVT)	0,19	3,89 r
..		
U_CPU/U_REG_FILE/U_REG_FILE_CORE/U3839/ZN (CKND2D4BWP7THVT)	0,15	4,91 f
U_CPU/U_REG_FILE/U_REG_FILE_CORE/porta[1] (net)	0	4,91 f
..		
U_CPU/U_ALU/rf_porta[1] (alumuldiv)	0	4,91 f
U_CPU/U_ALU/rf_porta[1] (net)	0	4,91 f
U_CPU/U_ALU/U874/Z (CKXOR2D4BWP7THVT)	0,47	5,37 f
U_CPU/U_ALU/n488 (net)	0	5,37 f
U_CPU/U_ALU/U1026/ZN (NR2XD1BWP7THVT)	0,13	5,51 r
..		
U_CPU/U_ALU/U130/ZN (CKND2D3BWP7THVT)	0,1	6,44 r
U_CPU/U_ALU/alu_cond (net)	0	6,44 r
..		
U_CPU/U_PREFETCH_BUFFER/branch_taken_iss1 (net)	0	6,69 r
U_CPU/U_PREFETCH_BUFFER/U222/ZN (INVD2BWP7THVT)	0,13	6,81 f
..		
U_CPU/U_PREFETCH_BUFFER/U413/ZN (IOA21D2BWP7THVT)	0,14	7,65 r
U_CPU/U_PREFETCH_BUFFER/haddri[7] (net)	0	7,65 r
U_CPU/U_PREFETCH_BUFFER/haddri[7] (prefetch)	0	7,65 r
U_CPU/haddri[7] (net)	0	7,65 r
U_CPU/haddri[7]	0	7,65 r
haddri[7] (net)	0	7,65 r
data arrival time		7,65
clock clk (rise edge)	15	15
clock network delay (ideal)	0	15
output external delay	-9	6
data required time		6
data required time		6
data arrival time		-7,65
slack (VIOLATED)		-1,65

Figure 6.3: Timing report for the Shared Master-latch design

The timing report for the Shared Master-latch design are presented in Figure 6.3. The path starts in the register file core and ends in *haddri*, as the Original design.



Point	Inc	Path
clock clk (rise edge)	0	0
clock network delay (ideal)	0	0
time given to startpoint	2,42	2,42
U_CPU/U_REG_FILE/U_REG_FILE_CORE/porta_reg_7_/D (LHQD1BWP7THVT)	0,15	2,45 f
U_CPU/U_REG_FILE/U_REG_FILE_CORE/porta_reg_7_/Q (LHQD1BWP7THVT)	0,79	3,21 f
U_CPU/U_REG_FILE/U_REG_FILE_CORE/porta[7] (net)	0	3,21 f
..	0	3,21 f
U_CPU/U_ALU/rf_porta[7] (net)	0	3,21 f
U_CPU/U_ALU/U1100/Z (CKXOR2D4BWP7THVT)	0,57	3,78 f
..		
U_CPU/U_ALU/U467/ZN (ND2D2BWP7THVT)	0,14	5,12 f
..		
U_CPU/alu_cond (net)	0	5,12 f
U_CPU/U197/ZN (CKND2D4BWP7THVT)	0,13	5,25 r
U_CPU/n251 (net)	0	5,25 r
U_CPU/U196/ZN (ND2D6BWP7THVT)	0,12	5,38 f
U_CPU/branch_taken (net)	0	5,38 f
U_CPU/U_FLOW_CTRL/branch_taken (flow_ctrl)	0	5,38 f
..		
U_CPU/U_FLOW_CTRL/output_new_pc (flow_ctrl)	0	5,60 f
U_CPU/output_new_pc (net)	0	5,60 f
U_CPU/U207/ZN (CKND12BWP7THVT)	0,13	5,73 r
..		
U_CPU/U_PREFETCH_BUFFER/pc_next[6] (prefetch)	0	6,06 f
..		
U_CPU/U_PREFETCH_BUFFER/haddri[6] (prefetch)	0	6,61 f
U_CPU/haddri[6] (net)	0	6,61 f
U_CPU/haddri[6]	0	6,61 f
haddri[6] (net)	0	6,61 f
data arrival time		6,61
clock clk (rise edge)	15	15
clock network delay (ideal)	0	15
output external delay	-9	6
data required time		6
data required time		6
data arrival time		-6,61
slack (VIOLATED)		-0,61

Figure 6.4: Timing report for the Shared Slave-latch design

The timing report for the Shared Slave-latch design, Figure 6.4, does as the Shared Master-latch design start the path of the timing report in the register file core and ends it in *haddri*.

6.7 Placement Results

In this section are the results from the Place & Route analysis of the design presented. Place & Route is run with the ICC tool described in Section 3.0.2. The analysis are run on the One-hot encoded design with and without relative placement of the flip-flops of the registers in the register file.

6.7.1 Power consumption results

The ICC generates reports on the power consumption of the system after the Place & Route analysis. These are presented in this subsection, the power consumption of the One-hot encoded design with and without relative placement of the flip-flops in the registers of the register file.

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	%
io_pad	0	0	0	0	0%
memory	0	0	0	0	0%
black_box	0	0	0	0	0%
clock_network	3.1232E-02	3.9148E-02	567.9432	7.0380E-02	27.97%
register	5.6596E-02	1.8740E-03	5.6666E+03	5.8476E-02	23.24%
sequential	1.3549E-02	2.5370E-05	5.7393E+03	1.3580E-02	5.40%
combinational	2.5852E-02	8.3330E-02	3.5139E+04	0.1092	43.40%
Total	0.1272 mW	0.1244 mW	4.7113E+04 pW	0.2517 mW	

Table 6.10: Power Consumption in design without relative placement of flip-flops

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	%
io_pad	0	0	0	0	0%
memory	0	0	0	0	0%
black_box	0	0	0	0	0%
clock_network	2.8901E-02	5.0366E-02	445.6206	7.9268E-02	30.82%
register	5.6330E-02	2.2060E-03	5.6668E+03	5.8541E-02	22.76%
sequential	1.3532E-02	5.3476E-05	5.7393E+03	1.3591E-02	5.28%
combinational	2.4689E-02	8.1054E-02	3.6053E+04	0.1058	41.13%
Total	0.1235 mW	0.1337 mW	4.7904E+04 pW	0.2572 mW	

Table 6.11: Power Consumption in design with relative placement of flip-flops

6.7.2 Area usage reports

This subsection presents the area usage reports generated by the ICC. They differ a bit from the area reports generated by the DC, with for instance net length and separate buffer and inverter count and area. Area reports for both the system with and without the relative placement scheme is present. Note that the area reports are for the whole system, and not for only the register file.

Hierarchical Cell Count	228
Hierarchical Port Count	7119
Leaf Cell Count	28282
Buf/Inv Cell Count	4422
Buf Cell Count	479
Inv Cell Count	3943
CT Buf/Inv Cell Count	116
Combinational Cell Count	24840
Sequential Cell Count	3442
Macro Count	0
Combinational area	53734.820
Noncombinational area	22705.199
Buf/Inv area	7696.360
Total Buffer Area	1498.560
Total Inverter Area	6197.800
Macro/Black Box area	0
Net area	0
Net XLength	309860.281
Net YLength	298719.438
Cell area	76440.019
Design area	76440.019
Net Length	608579.750

Table 6.12: Area usage in design without relative placement of flip-flops

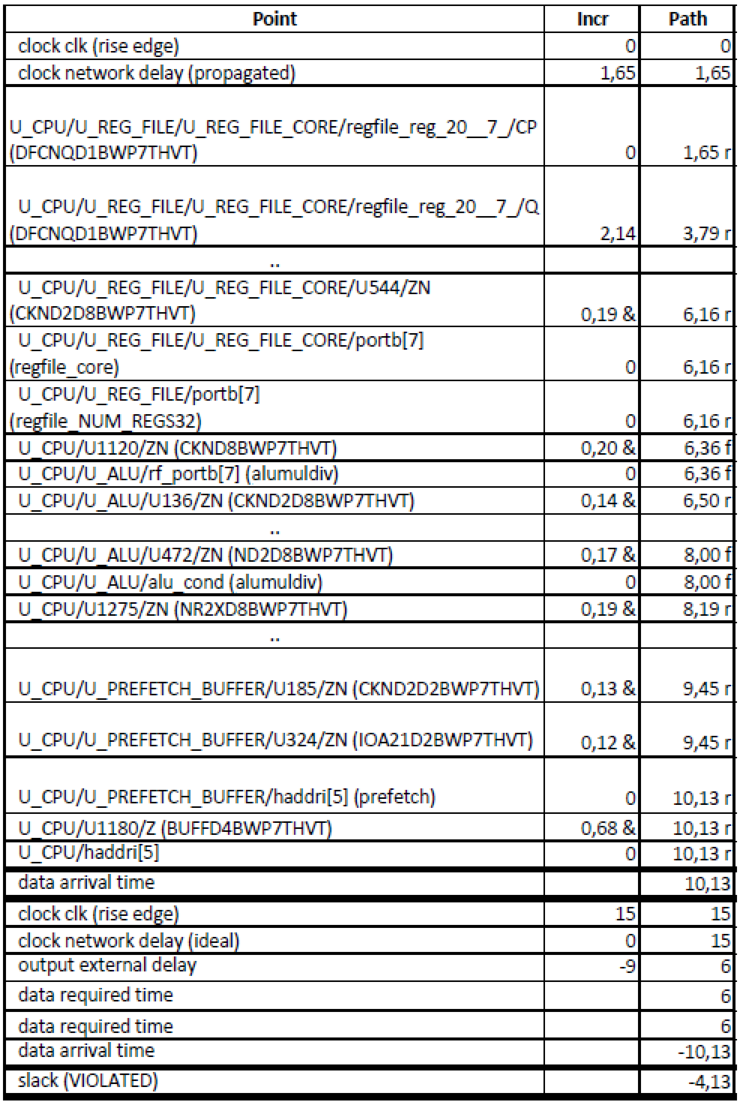
Hierarchical Cell Count	228
Hierarchical Port Count	6993
Leaf Cell Count	28396
Buf/Inv Cell Count	4705
Buf Cell Count	548
Inv Cell Count	4157
CT Buf/Inv Cell Count	110
Combinational Cell Count	24954
Sequential Cell Count	3442
Macro Count	0

Combinational area	53804.620
Noncombinational area	22657.039
Buf/Inv area	8450.680
Total Buffer Area	1807.400
Total Inverter Area	6643.280
Macro/Black Box area	0
Net area	0
Net XLength	299700.938
Net YLength	429033.438
Cell area	76461.659
Design area	76461.659
Net Length	728734.375

Table 6.13: Area usage in design with relative placement of flip-flops

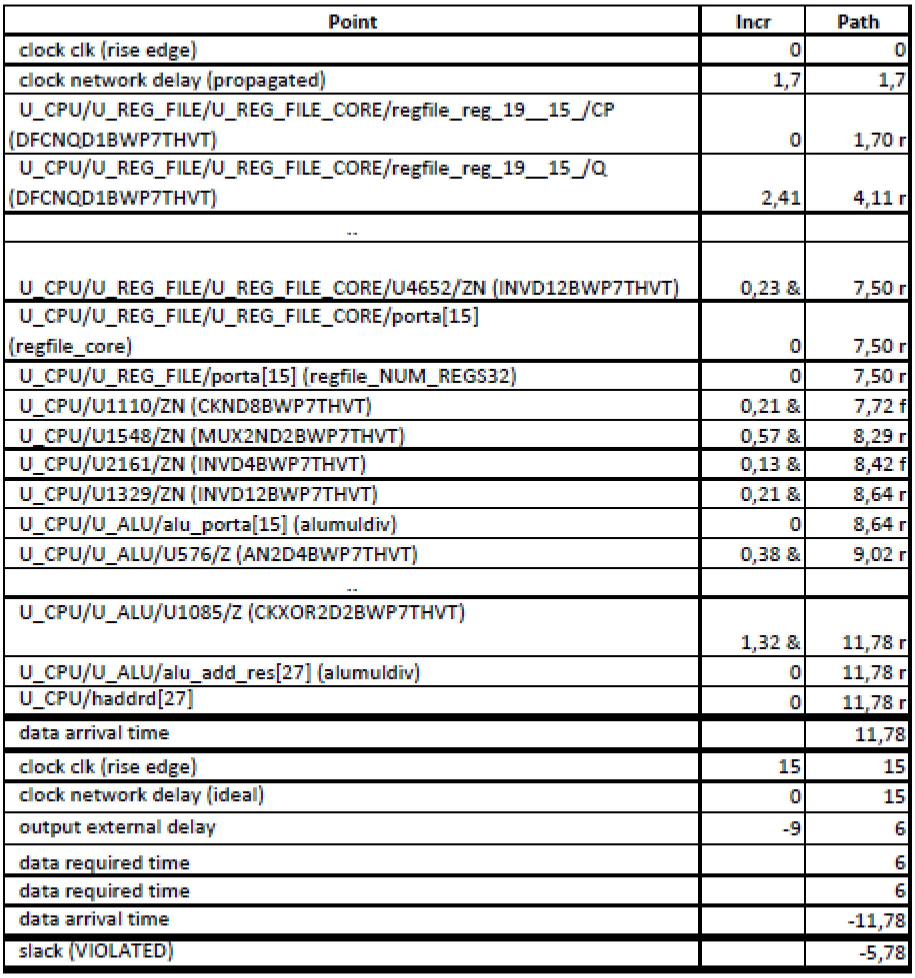
6.7.3 Timing Results

This subsection presents the timing reports generated by the ICC. The timing analysis are done after placement and the reports presented in this section are chosen to include timing reports that traverse through the register file for accurate measurements on how the relative placement scheme affects the timing of the system.



Point	Incr	Path
clock clk (rise edge)	0	0
clock network delay (propagated)	1,65	1,65
U_CPU/U_REG_FILE/U_REG_FILE_CORE/regfile_reg_20__7_/CP (DFCNQD1BWP7THVT)	0	1,65 r
U_CPU/U_REG_FILE/U_REG_FILE_CORE/regfile_reg_20__7_/Q (DFCNQD1BWP7THVT)	2,14	3,79 r
..		
U_CPU/U_REG_FILE/U_REG_FILE_CORE/U544/ZN (CKND2D8BWP7THVT)	0,19 &	6,16 r
U_CPU/U_REG_FILE/U_REG_FILE_CORE/portb[7] (regfile_core)	0	6,16 r
U_CPU/U_REG_FILE/portb[7] (regfile_NUM_REGS32)	0	6,16 r
U_CPU/U1120/ZN (CKND8BWP7THVT)	0,20 &	6,36 f
U_CPU/U_ALU/rf_portb[7] (alumuldiv)	0	6,36 f
U_CPU/U_ALU/U136/ZN (CKND2D8BWP7THVT)	0,14 &	6,50 r
..		
U_CPU/U_ALU/U472/ZN (ND2D8BWP7THVT)	0,17 &	8,00 f
U_CPU/U_ALU/alu_cond (alumuldiv)	0	8,00 f
U_CPU/U1275/ZN (NR2XD8BWP7THVT)	0,19 &	8,19 r
..		
U_CPU/U_PREFETCH_BUFFER/U185/ZN (CKND2D2BWP7THVT)	0,13 &	9,45 r
U_CPU/U_PREFETCH_BUFFER/U324/ZN (IOA21D2BWP7THVT)	0,12 &	9,45 r
U_CPU/U_PREFETCH_BUFFER/haddri[5] (prefetch)	0	10,13 r
U_CPU/U1180/Z (BUFFD4BWP7THVT)	0,68 &	10,13 r
U_CPU/haddri[5]	0	10,13 r
data arrival time		10,13
clock clk (rise edge)	15	15
clock network delay (ideal)	0	15
output external delay	-9	6
data required time		6
data required time		6
data arrival time		-10,13
slack (VIOLATED)		-4,13

Figure 6.5: Timing report for the design without relative placement of flip-flops



Point	Incr	Path
clock clk (rise edge)	0	0
clock network delay (propagated)	1,7	1,7
U_CPU/U_REG_FILE/U_REG_FILE_CORE/regfile_reg_19__15_/CP (DFCNQD1BWP7THVT)	0	1,70 r
U_CPU/U_REG_FILE/U_REG_FILE_CORE/regfile_reg_19__15_/Q (DFCNQD1BWP7THVT)	2,41	4,11 r
--		
U_CPU/U_REG_FILE/U_REG_FILE_CORE/U4652/ZN (INVD12BWP7THVT)	0,23 &	7,50 r
U_CPU/U_REG_FILE/U_REG_FILE_CORE/porta[15] (regfile_core)	0	7,50 r
U_CPU/U_REG_FILE/porta[15] (regfile_NUM_REGS32)	0	7,50 r
U_CPU/U1110/ZN (CKND8BWP7THVT)	0,21 &	7,72 f
U_CPU/U1548/ZN (MUX2ND2BWP7THVT)	0,57 &	8,29 r
U_CPU/U2161/ZN (INVD4BWP7THVT)	0,13 &	8,42 f
U_CPU/U1329/ZN (INVD12BWP7THVT)	0,21 &	8,64 r
U_CPU/U_ALU/alu_porta[15] (alumuldiv)	0	8,64 r
U_CPU/U_ALU/U576/Z (AN2D4BWP7THVT)	0,38 &	9,02 r
--		
U_CPU/U_ALU/U1085/Z (CKXOR2D2BWP7THVT)	1,32 &	11,78 r
U_CPU/U_ALU/alu_add_res[27] (alumuldiv)	0	11,78 r
U_CPU/haddrd[27]	0	11,78 r
data arrival time		11,78
clock clk (rise edge)	15	15
clock network delay (ideal)	0	15
output external delay	-9	6
data required time		6
data required time		6
data arrival time		-11,78
slack (VIOLATED)		-5,78

Figure 6.6: Timing report for the design with relative placement of flip-flops

6.7.4 Graphical placement results

This section presents the graphical representation of the placement of the system. The cells of the register file are highlighted, mainly the flip-flops as they are relative placed. Both the placement with and without the relative placement directives is present in this section.

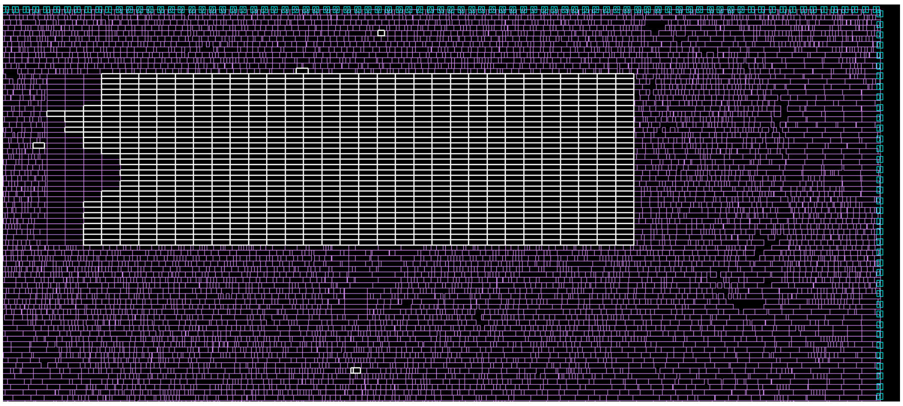


Figure 6.7: Placement of flip-flop cells with relative placement directives

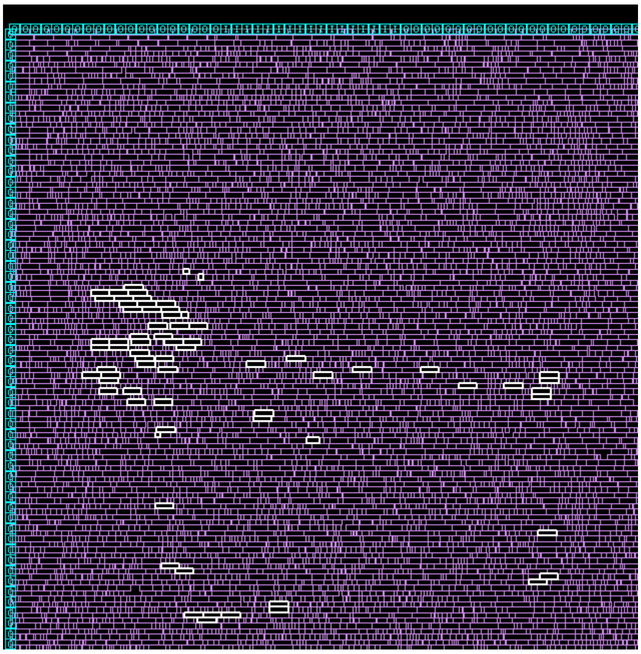


Figure 6.8: Placement of flip-flops cells without relative placement directives

Chapter 7

Evaluation and Discussion

This chapter wraps up chapter 6 and evaluate and discuss the provided results from the research work of the thesis. The evaluation section discuss the result of each design, and evaluates the success of the different optimisation schemes. The discussion continues in the discussion section, where the designs are compared and which is most successful are discussed. The Original design are the discussion basis in the evaluation and discussion of the results as this are the starting point of the optimisation, but the One-hot is used to compare with the latch based designs as they are using the same selection logic.

7.1 Evaluation

The evaluation section will discuss the results for the given parts of the thesis. Whether assumptions are proven right or wrong, and the success of each architecture or method of register file optimisation, will be discussed. The section is divided into the three different register file architectures, and the relative placement experiment.

7.1.1 The one-hot encoded design

This design was originally meant to make later design changes more streamlined, and not an optimisation of the register file itself. Some improvement were expected as the switching activity when using one-hot encoding are lower, and this were proven to be the case as can be seen in the subsection about power consumption. The latch based designs are using the same one-hot encoded selection logic as this design, so the evaluation of the one-hot encoded design are thus important as well.

The architecture of the One-hot encoded design is not very different from the Original design only that the selection signals are one-hot encoded instead of regular signals. This requires additional combinational logic to encode the one-hot encoded signals, and changes in how the RTL code is written could have affected the synthesized design in terms of inferred combinational logic cells.

7.1.1.1 Area usage

Table 6.7 shows the area results of the register file to the One-hot encoded design. Compared to the results from the Original design, Table 6.6, are the total number of cells in the One-hot encoded design 7459 and 6726 in the Original design. This increase in number of cells are shown to be in the combinational logic, namely from 5670 to 6403 cells and an increase of 1026,9 in area usage. Otherwise are the numbers the same.

The increase of 733 combinational cells is related to the encoding of the selection signals. 200 cells, mostly NAND and NOR gates, are used to one-hot encode the four selection signals. The rest of the increase could stem from extra logic required since the selection signals are 27 bits wider than in the original register file and some could be because the RTL code is not written entirely similar.

In the end were an increase in combinational logic for the One-hot encoded design expected as the architecture and functionality are the same, and still based on registers consisting of flip-flops. The only changes are then extra logic thus increasing the number of cells and area usage.

7.1.1.2 Power consumption

Table 6.2 shows the results from the power estimation of the One-hot encoded design. Compared to the total power consumption in the Orignal design, Table 6.1, is the total power consumption reduced from 174.51 μW to 164.087 μW, a 5.97 percent reduction. The register file is the only architectural change in the design, so the results from the power analysis of the register file are better for comparison. Table 6.5 shows the power consumption in the register file of the different designs. Here is the improvement more clear from the Original design to the One-hot encoded design, more specifically a drop from 25.497 μW to 15.703 μW, a 38.41 percent reduction. The leakage power are negligible, so most of the improvement are made in the Internal and Switching power. To recap how the Internal power and the Switching power is calculated, described in Section 3.0.3, the Internal Power is dependent on the transition whitin the boundary of a cell when a transition occurs, and the Switching power is proportional to the toggling activity of nets.

The architecture itself are based on the same cells as the Original, with registers consisting of flip-flops. The reduction of power consumption is then a result of the improvement on the selection logic. The traits of one-hot encoded signals are that switching the value requires less toggling as only two bits are changing its values when changing the value of the signal. By having the selection signals one-hot encoded, toggling activity in the register file are reduced. The Internal power and the Switching power are thus reduced in the register file, which leads to the decrease in power consumption in the design compared to the original.

The activity analysis from SpyGlass Power Estimate shows that the average activity in the register file of the one-hot encoded design are 38.67 percent less than in Original design, corresponding nicely with the 38.41 percent drop of power consumption as it is mostly dynamic power.

7.1.1.3 Timing

The timing reports for the One-hot encoded signal can be seen in Figure 6.2, and the design meets its timing constraints. This is the case for the Original design as well, which is presented in Figure 6.1. The timing report is showing the data propagation from U_CPU/U_PREFETCH_BUFFER/opcode_rs1_reg_2_ to U_CPU/haddrd[28], a path that accesses the register file. This access ensures that the timing constraints are met when accessing the register file. Since the one-hot encoding is a simple, combinational operation, it is expected that the One-hot encoded design meet its timing constraints as the Original design do not have a negative slack and have the same sequential logic.

7.1.2 The Shared Master-latch design

This design splits up the traditional master-slave setup for a flip-flop and uses an approach that shares the master latches, reducing the total number of latches. There is a unique cell inferred for flip-flops, so the number of register cells is not reduced. A flip-flop cell require a larger part of the area per cell, than a latch cell. Using latches deems more difficult in terms of timing and data validity as latches are not following the clock edge as flip-flops does. This requires careful usage of clock-gating and enable signals, to ensure that data is not read when a latch is transparent, and that data is captured at the correct time. As a consequence of this, extra latches are added to the design to delay the signals that enables the writing to the register file, so that when data is written it are done correctly. This can be seen in the area results shown in Table 6.8, as the area used in the latches for delay logic part of the table.

7.1.2.1 Area usage

The flip-flop based designs got 1024 register cells, flip-flops, and the Shared Master-latch design got 1088 register cells, 1024 slave latches and two sets of master latches of 32 latches each, which corresponds well with the theory presented in Chapter 2. It is however, added extra latches to delay the selection logic to ensure the validity of the writing of data from the master latches to the slave latches. There are a total of 66 latches used for the delay logic, making the total of extra register elements 130, compared to the flip-flop based designs.

A flip-flop cells occupy area of 6.72, while a latch occupy 3.92 units of area. So even though there is an increase of 130 register elements, the total area usage of the sequential cells goes down from 6881.28 to 4542.16, only 66 percent of the area usage for sequential logic as in the original register file. There are also a reduced number of combinational logic cells required to drive the Shared Master-latch register file, compared to the One-hot encoded design, although the area usage are about the same.

In total do the Shared Master use 6583 cells and an area of 13889.12 as seen in Table 6.8. This deems an improvement on the One-hot encoded design of 876 less cells and 15.58 percent less area usage. As discussed stems most of the improvement from the reduced size of the latch cells compared to the flip-flop cells.

7.1.2.2 Power consumption

The results of the power analysis of the Shared Master-latch design are presented in Figure 6.3. The total power consumption the design is 159.989 μW and the total power consumption of the register file is 11.877 μW. The improvement on the Original design is great, but the design is based on the One-hot encoded design and thus uses the same one-hot encoded selection logic. This means that it is more natural to look at the improvement compared to the One-hot encoded design rather than the Original design. The improvement in terms of total power is a drop of 2.5 percent and a drop of 24.36 percent for the total power of the register file. M. Wróblewski and Nossek [2003] states that the savings on power consumption could be up to 50 percent and with 24.36 percent is this implementation not as efficient. This implementation of the Shared Master-latch register file has two input ports, and thus two sets of master latches, which could contribute to the difference in power consumption savings.

The improvement is in the register file, and is present in all the different sources of power consumption. The leakage power is mainly given by the cells present in the design, and the latch cells have a lesser leakage power than flip-flop cells.

The power activity analysis gives that the activity in the register file are about the same as in the One-hot encoded design, but in the register file core, where the actual register file architecture are implemented, are the activity 28.4 percent lower. The usage of latches decrease the power consumption as well, as the dynamic power consumption of a latch is less than a flip-flop. A combination of the low activity on the registers and the improvement because of the latches are the total power consumption of the Shared Master-latch design the lowest in total of the approaches researched in this thesis.

7.1.2.3 Timing

The timing reports for the Shared Master-latch design are shown in Figure 6.3. The timing report shows the path from U_CPU/U_REG_FILE/U_REG_-FILE_CORE/regfile_reg_14___1_ to U_CPU/haddri[7]. The report shows that the timing constraint are not met. Looking at the figure it is obviously that the timing violation stems from the traits of the latches that the register file are based upon. While the Original design and the One-hot encoded starts the timing calculation at 0, the Shared Master-latch design are given a penalty of 3.03 to the start point, giving little chance of meeting the timing constraints. It is unclear exactly why the latch based designs are given a different time at start point than the flip-flop based designs. That the latches are level-triggered while the flip-flops are edge-triggered is certainly a part of the problem, as the timing constraints are set up for the Original design. Without special considerations are the DC looking at the Shared Master-latch design as a regular flip-flop based register file and does not take the nature of latches into account, giving the penalty to account for the level-edge nature of the latches, to wait for the clock to be on rising edge before further timing analysis. Without the extra time given at start, the slack would have been positive.

7.1.3 The Shared Slave-latch design

The Shared Slave-latch design are, as the Shared Master-latch design, based on latches instead of flip-flops in the register file architecture. The basics are the same except that the extra latches are moved from the input to the output, i.e. there are the slave latches that are shared, not the master latches. This gives the same amount of latches used for the register file architecture, but there are fewer latches in total, since the Shared Slave-latch design does not have to add the delay to the selection logic. This design is as well one-hot encoded, produced from the One-hot encoded design, and it is thus more reasonable to compare the results to that design. The nature of the CPU, that the selected register to the output ports does change more frequently than the data written to the register file, increases the activity of the Shared Slave-latch design as well as increase the power consumption, which will be described further in following sections.

7.1.3.1 Area usage

Table 6.9 shows the area results for the register file architecture used in the Shared Slave-latch approach. The total number of cells is reduced from the Shared Master-latch design by 133 cells. 66 of those are from the removal of the latches for the delay logic. The remaining decrease of 67 occurs in the combinational logic. This could stem from, or partly stem from, that the shared master latches are gated by both the clock and write enable signals in the Shared Master-latch design, while the shared slave latches in the Shared Slave-latch design are only enabled by the clock, decreasing the number of cell required for the enable logic. Otherwise, some of the decrements could be present because of how the RTL are written and how it is interpreted by the DC.

Compared to the One-hot encoded design is the Shared Slave-latch design 20 percent more efficient in area usage. The improvement is largely because of the replacement of flip-flops with latches, namely an improvement in area usage from 6881.28 to 4264.96. There are used less cells for the combinational logic, but the decrease in used area is not great in that perspective. In total is the Shared Slave-latch approach the most efficient architecture in terms of area usage. Mostly because it has the same effect on the area of the latches as the Shared Master-latch design, and it does not require the extra delay logic.

7.1.3.2 Power consumption

The results of the power analysis of the Shared Master-latch design are presented in Table 6.4. The total power consumption are slightly better than in the One-hot encoded design, but worse than the Shared Master-latch design.

The reason that the Shared Slave-latch design uses more power than the Shared Master-latch design even though it is latch-based and uses less latches is that the activity in the register file is higher. This increases the dynamic power more than the reduction in number of latches does. The activity is increased as the slave latches on the output are only gated by the clock, as opposed to the master latches in the Shared Master-latch design which are gated by the clock and a write enable signal for each master latch setup. In addition may the system more frequently change which register it reads to the output than it writes to the input. All in all are the total power consumption for the system 163.454 μW and the total power consumption of the register file are 13.576 μW. There are still a decrease of 13.54 percent compared to the One-hot encoded design, a significant reduction of power even though it does not beat the Shared Master-latch design.

7.1.3.3 Timing

The timing reports for the Shared Master-latch design are shown in Figure 6.4. The timing report shows the path from U_CPU/U_REG_FILE/U_REG_-FILE_CORE/porta_reg_7_/D to U_CPU/haddri[6]. The report shows that the timing constraint are not met with a negative slack of -0.61. Looking at the figure it is obviously the timing violation stems from the traits of the latches that the register file are based upon. As the Shared Master-latch design is the time given to start point greater than zero, 2.42 in this case. Otherwise does the system prove efficient, as the slack would indeed have been positive without the penalty on the start point. This is probably because of the traits of the latches, which was also discussed in Subsection 7.1.2.3.

7.1.4 Relative Placement of the One-hot encoded design

This approach differ from the rest as it is no changes in the RTL of the design. The relative placement directives are fed to the Place & Route tool which places the specified cells as given in the directives.

The theory is that placing the flip-flops of the registers in the register file close by each other may be an improvement to letting the ICC place and optimise the design.

Originally were the scheme to place adjacent cells, logic like multiplexers, relative to the flip-flops as well. However, deemed the naming scheme of the combinational logic cells to be too difficult to include in the script that generates the relative placement directives. In the end, did the relative placement approach deem no real advantages compared to letting the ICC place all the cells.

The results presented for the relative placement experiment are a bit different than for the alternate architectures. This is because no waveform were available, and thus not making it possible to run the SpyGlass Power Estimate. The ICC does generate reports on power consumption, area usage and timing, but as another tool are used, the data is not comparable to the results from the architectural exploration, but are instead compared with a regular placement analysis of the One-hot encoded design.

Figure 6.7 shows the placement using the relative placement directives when running the Place & Route analysis. The flip-flop cells are clearly placed in a columns closely by each other. Figure 6.8 shows how the cells are placed without the relative placement directives. Note that fewer cells are highlighted, as they are not related without the relative placement.

7.1.4.1 Area usage

The area report for the regular design is shown in Table 6.12 and the area report for the design run with the relative placement directives is shown in Table 6.13. As of the total cell area, is the differences negligible. This is natural as it is about the same amount of cells, except that some extra cells are inferred for the relative placement design. Some extra combinational cells are added, as well as buffers and inverters. It is apparent from Figure 6.7 that by placing the flip-flops together, the paths to previously adjacent logic are extended which leads to the extra cells and the total net length are increased with 19.74 percent. The area report are missing information of net area, and this is because the wire load are not specified in the technology library provided.

7.1.4.2 Power consumption

The results for both the regular One-hot encoded design and the one with relative placement are, as was the case with area usage, pretty similar. Table 6.10 shows that the total power without the use of relative placement are 251.7 μW, while Table 6.11 shows that the total power consumption using relative placement are 257.2 μW. The leakage power are the same, so it is the dynamic power that are affected by the placement of the flip-flops. The relative placement scheme deems positive results in terms of the cell internal power usage. This source of power consumption stems from the toggling of the cells internal value. The switching power on the other hand, power used on switching of nets, are showing negative results. This stems from the increase in length of the nets, increasing the net switching penalty. In the end are the negative effects on the switching power greater than the positive effects on the internal cell power, making the total power consumption higher in the design with relative placement that without. The difference are, however, negligible, only about 2 percent higher.

7.1.4.3 Timing

Neither approach meets the timing constraints. The constraints are inferred from the tool and it is not important whether they fail or pass, but by comparing similar paths they can be compared to evaluate how the changes affects the designs ability to meet its timing constraints. Figure 6.6 shows that the design with relative placement of flip-flops do have a negative slack of -5.78, while the original design, Figure 6.5 have a negative slack of -4.13. Both timing reports shows the path from one flip-flop in the register file to the one instance of the output *haddr*. The time it takes the relative placed design to go from the flip-flop, to the output port of the register file are 7.50, while it is 6.16 in the original design. The decrease in the slack is caused by the increased length of the nets of the system as well as the decreased locality of the connected cells.

7.2 Discussion

This section discusses and compares the different optimisation techniques explored in the thesis. Parameters such as area usage, power consumption and timing will be considered to evaluate the success of each design. There are trade-offs, as seen previously in this chapter, to the different optimization approaches, between the different evaluation criterias. Section 7.1 where the results are evaluated will be the basis of this discussion.

7.2.1 Comparison of the architectural optimisations

The different optimisation techniques will in this section be compared based on how the optimisation approaches has affected the systems power consumption, timing and area usage. Different trade-offs will be discussed, and whether or not an alternate approach is worth replacing the existing register file architecture.

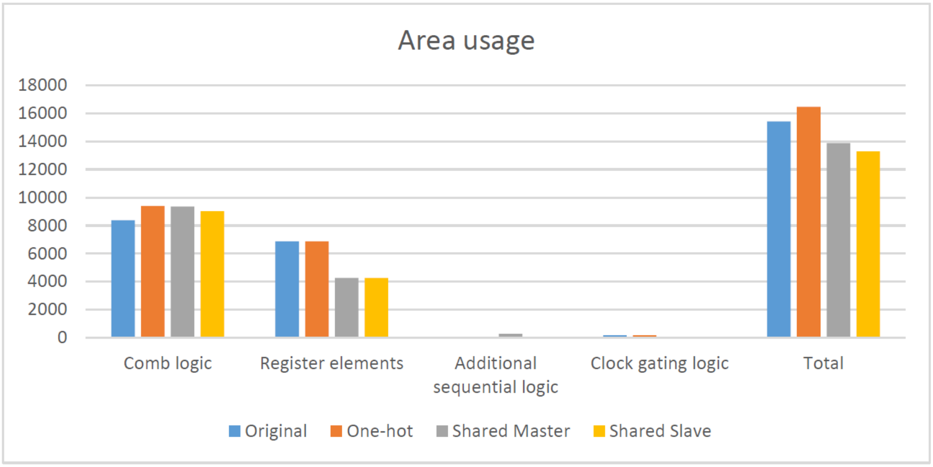


Figure 7.1: Comparison of area usage of the different design approaches

Figure 7.1 shows the differences in the various designs in terms of area usage. The data presented is the area used for each register file. It shows that the logic for one-hot encoding the signals affects the combinational logic in the three custom made designs, as the graph shows that the area used for the combinational logic are greater in these than in the original. The Shared Slave-latch design is slightly better than the other two custom made designs. The column Register elements shows the area used for the register file itself, and clearly displays the gain of using latches instead of flip-flops with respect to the area usage. In total, since it does not have any extra cells as for the clock gating or delaying logic, are the Shared Slave-latch design the most efficient on area usage.

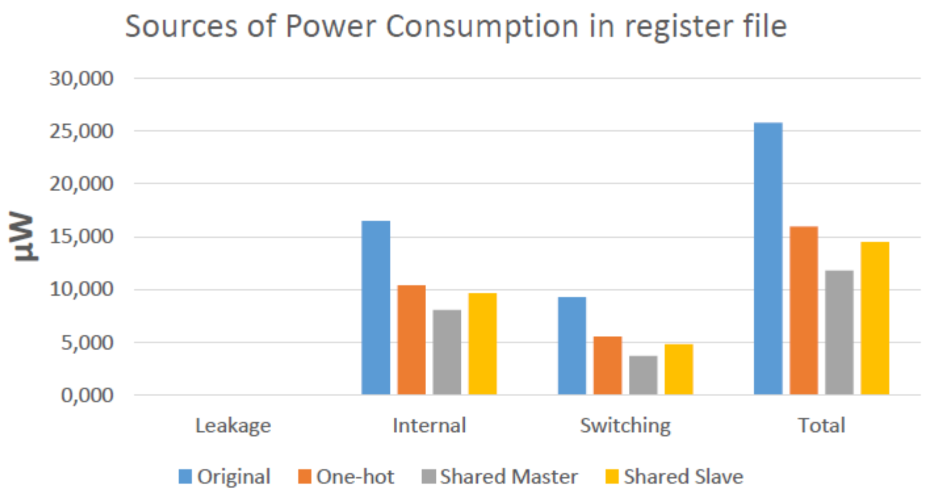


Figure 7.2: Comparison of power consumption of the different design approaches

The power consumption in the designs are showed in Figure 7.2. The graph clearly shows how the static power, the leakage power, are negligible compared to the dynamic power. The original design measures up poorly compared to the alternate designs. The one-hot encoded logic clearly is the cause of the difference, as the three alternate designs all uses one-hot encoded selection logic. The latch designs proves to be more efficient than the one-hot encoded design, although not so much for the Shared Slave-latch design. The dynamic power are consisting of the switching power and the internal power. The switching power is given by the switching activity of nets, and the capacative contribution from cell pins and the internal power is calculated from the power dissipated by cells when transitions occurs. In the end are the dynamic power given by the switching activity and the cells. The activity are about the same in the one-hot encoded design as in the Shared Master-slave latch design so the improvement stems from the switch from flip-flops to latches. Even though the Shared Slave-latch design also consist of latches, actually fewer than the Shared Master-latch design, the power consumption is greater. This is because the switching activity are greater in the Shared Slave-latch design, which is discussed further in Section 7.1.

The slack of the different designs are shown in Table 7.1. This shows that the there are no timing violations in the flip-flop based architectures, but the latch based architectures have different degrees of negative slack i.e. timing violations.

	Original	**One-hot encoded**	**Shared Master-latch**	**Shared Slave-latch**
Start point	U_-PREFETCH_-BUFFER-/opcode_-rs1_reg_-4_	U_-PREFETCH_-BUFFER-/opcode_-rs1_reg_-2_	U_REG_-FILE_-CORE/reg-file_reg_-14_1_	U_REG_-FILE_-CORE/-porta_reg_-7_/D
End Point	haddri[11]	haddrd[28]	haddri[7]	haddri[6]
Slack	0	0	-1.65	-0.61

Table 7.1: Comparison of timing report of the different design approaches

As discussed in Section 7.1 may the negative slack stem from the fact that the systems constraints are set up for a system with flip-flops, making it apparent that the time penalties given in the timing reports of the latch based design may not be valid. Even though the slack may be invalid, the timing are important to make sure the system works properly when made into a physical chip. This means that the latch based designs will wot not work properly, or not provide reliable data storage as data can be lost if the timing is off when storing or loading data to the register file.

7.2.2 Comparison of the relative placement of flip-flop design

It is hard to compare the design with relative placement of flip-flops with the architecture optimisations as the results are collected with different means, i.e. from the ICC compared to from synthesis done with DC and estimation done with SpyGlass Power Estimate. There are however run Place & Route analysis on the One-hot encoded design with and without relative placement directives, as discussed in Section 7.1. The results shows that the relative placement approach does not improve the design in terms of area usage, power consumption or timing. It does on the other hand make it slightly worse, specially in terms of timing. This is because by grouping the flip-flops together, the wires connected to adjacent cells becomes longer, in fact is the net length increased from 608579.75 to 728734.375, an increase of 19.68 percent. This is what makes the slack to go down from -4.13 to -5.78, as well as some extra, inferred combinational logic. In the end was the experiment with relative placement of flip-flops unsuccessful.

7.3 Contributions

This thesis provides alternatives to the currently existing register files. As discussed in Section 7.1 and Section 7.2 were the experiment on alternate architectures successful. Each of the three alternate register file architectures had its advantages in terms of power consumption or area usage. The results presented proves that there are alternatives to the original flip-flop based register file, that would improve the efficiency of the memory of the system the experiments were done upon. As Nordic Semiconductor are a company that focuses on low-power solutions, one of the alternate, more power efficient architectures would be a good fit.

Chapter 8

Conclusion

The results presented in Chapter 6 and the following discussion and evaluation in Chapter 7 lead to the following conclusion on the experiments conducted in this thesis. First of all, the scheme of relative placement of flip-flops in the register file deemed unsuccessful. There were no improvements in the evaluation parameters, power consumption, area usage or timing. The results were rather on the negative side, with basis in an 20 percent increase in wire length.

The work on the alternate register file architectures proved more successful. The designs implemented were a design with improvements on the selection logic, one-hot encoded. This design were similar to the original flip-flop based register file except for said selection logic. Furthermore were two latch based designs made with the One-hot encoded design as a foundation, making the latch based designs using the same one-hot encoded selection logic. The latch based designs were designed after the theory on Shared Master-latch register files and Shared Slave-latch register files described in Chapter 2. The difference were that the master latches are shared in the Shared Master-latch design, thus latching the input, while the Shared Slave-latch design are latched on the output, sharing the slave latches. The optimisation on the selection logic added extra area in terms of extra cells for the encoding of the selection signals, but deemed highly effective in terms of power consumption. The power estimate showed that the activity in the register file went down 38.67 percent, reducing the dynamic power consumption with 38.41 percent. The latch based designs further improved the power consumption in the register file, with the Shared Master-latch design most efficient with an improvement of 53.42 percent on the original design and 24.36 percent on the one-hot encoded design. The Shared Slave-latch design deemed an improvement of 46.75 percent on the original design and 13.55 percent on the one-hot encoded design.

In terms of area usage showed the results that the Shared Slave-latch design was most efficient. Mostly because latches are smaller than flip-flops and that the Shared Master-latch design added extra latches to delay some of the logic. In the end did the Shared Slave-latch design uses 13.93 percent less area than the original design, and the Shared Master-latch design uses 9.97 percent less. On the other hand, did the area usage increase by 6.66 percent in the design with the one-hot encoded selection logic. Timing, the last success criteria, showed some limitations in the designs. The flip-flop based design, the original and the one-hot encoded, both passed the timing reports without negative slack. Both the latch based designs did, however, report negative slack and thus timing violations. In fact did the Shared Master-latch design report a slack of -1.65 and the Shared Slave-latch design had a slack of -0.61. That the timing constraints are met are important for the physical behavior of the design, but there are reasons to question the result of the timing reports. This is because the timing constraints are written for the original design, a clock-edged, flip-flop based design and the latch based designs are on level-edge.

All in all proved the alternate architectures to be more effective than the original flip-flop register file based on the success criteria parameters area usage and power consumption. The designs were verified by throughout testing with and without a simulated, physical delay of data and proved successful. It is a possibility to change the register file used in the system this thesis is working on to either one of the alternate design approaches. The Shared Slave-latch architecture proved most efficient in area usage and the Shared Master-latch consumed least power. With timing in mind, proved the flip-flop design with one-hot encoded selection signals most reliable. Overall did the Shared Master-latch design improve the register file the most, and would be a good replacement for the original flip-flop design.

8.1 Future Work

The structure of the thesis makes it possible to extend the work as long as there exist other, alternate design approaches for register files. There were described a few in Chapter 2 that would be possible to test out as well as the approaches explored during this thesis. The multi-bit flip-flop approach were dropped because of missing libraries that support that kind of cell, but when that is available this would be interesting to explore. The design approaches in the thesis has been based on latches, and another approach would have improved the variability of the thesis. Such as the previously mentioned multi-bit flip-flop design that improves the flip-flop approach, or doing it a completely different direction as the SRAM based design presented in Chapter 2.

The relative placement of flip-flops in the register file deemed unsuccessful as there were no improvement on power consumption, area usage or slack. Much of the disadvantages stems from an increased net length, mainly because the distance between the flip-flops and connected cells increased. If the generator that generates the relative placement directive were to add those connected cells, this would not be the case. This was not done because that the ICC requires the reference name of the cells that are to be relative placed, and only the flip-flops in the register file inferred regular reference names for the cells in the current setup.

By either changing the naming of the connected cells in the netlist or making the generator comply to the existing naming, would it be possible to place the related cells in the relative placement group. This would decrease the net length and hopefully deem more effective in terms of area usage and power consumption.

A normal procedure of the making of register files are that they are automatically generated. To make a generator for the new register file architectures would be a way of implementing the register files into different systems. The structure are the same with different sizes on the register file, and it should be possible to make a generator which produce register files of different sizes. This applies to all the alternate designs produces in this thesis.

References

Introduction to physical datapath with relative placement. *Synopsys SolvNet*, 2017.

Arka De Debopam Ghosh, Jyotirmoy Guha and Anirban Mukherjee. Power reduction in modern vlsi circuits – a review. *International Journal of Students Research in Technology & Management*, 1(1):361–369, August 2013.

Gaurav Gupta Gourav Kapoor and Nalin Gupta. Using multi-bit flip-flop custom cells to achieve better soc design efficiency. *Freescale Semiconductor*, August 2014.

Shen-Fu Hsiao and Pu-Cheng Wu. Design of low-leakage multi-port sram for register file in graphics processing unit. *IEEE*, June 2014.

John Lazzaro John Wawrzynek, Krste Asanovic and Yunsup Lee (TA). Cs250 vlsi systems design lecture 8: Memory. 2010. URL `https://inst.eecs.berkeley.edu/~cs250/fa10/lectures/lec08.pdf`.

C. R. Kime, M.J. Schulte, and M. Lipasti. Register files and memories. *Digital Engineering Laboratory, ECE 554*, page 13, February 2007.

A. Wortmann S. Simon W. Pieper M. Wróblewski, M. Mueller and J. A. Nossek. A power efficient register file architecture using master latch sharing. *Circuits and Systems, 2003. ISCAS '03.*, May 2003.

Youngsoo Shin and Seungwhun Paik. Pulsed-latch circuits: A new dimension in asic design. *IEEE*, March 2011.

Synopsys. About design compiler. *Design Compiler User Guide, version M-2016.12*, 2017a. URL `https://solvnet.synopsys.com/dow_retrieve/M-2017.06/dcolh/Default.htm#dcug/dcug1_About_Design_Compiler.htm%3FTocPath%3DDesign%2520Compiler%2520Documents%7CDesign%2520Compiler%2520User%2520Guide%252C%2520version%2520M-2016.`

12%7CDesign%2520Compiler%2520Introduction%7CAbout%2520Design%2520Compiler%7C_____0.

Synopsys. Ic compiler. 2017b. URL https://www.synopsys.com/implementation-and-signoff/physical-implementation/ic-compiler.html.

Synopsys. Internal power. 2017c. URL https://solvnet.synopsys.com/dow_retrieve/M-2017.06/spyglass/spyglass_olh/htmlhelp/index.html#page/power_est%2FInternal_Power.htm%23.

Synopsys. Leakage power. 2017d. URL https://solvnet.synopsys.com/dow_retrieve/M-2017.06/spyglass/spyglass_olh/htmlhelp/index.html#page/power_est/Leakage_Power.htm.

Synopsys. Introduction to spyglass power estimation and spyglass power reduction. 2017e. URL https://solvnet.synopsys.com/dow_retrieve/M-2017.03/spyglass/spyglass_olh/htmlhelp/index.html#page/power_est%2FIntroduction.htm%23%20og%20https%2F%2Fsolvnet.synopsys.com%2Fdow_retrieve%2FM-2017.03%2Fspyglass%2Fspyglass_olh%2Fhtmlhelp%2Findex.html%23page%2FApplicationNotes%2FHow_Power_is_Computed.htm%23wwconnect_header.

Synopsys. Switching power. 2017f. URL https://solvnet.synopsys.com/dow_retrieve/M-2017.06/spyglass/spyglass_olh/htmlhelp/index.html#page/power_est%2FSwitching_Power.htm%23.

Abdullah Yildiz. Latches and flip-flops. February 2017. URL http://cse.yeditepe.edu.tr/~ayildiz/attachments/flipflops.pdf.

Appendices

Listing 1: Register file top module used for the custom made designs

```
module regfile (/*AUTOARG*/
   // Outputs
   porta, portb,

   // Inputs
   sel_porta, sel_portb, sel_rd, sel_rd2, rd, write_rd, rd2,
   write_rd2,hreadyd,
   clk, rst_n,allow_hidden_use_of_x0
   );

   parameter NUM_REGS=32;
`include "parameters.v"

  // Selection
   input   [RF_PORTA_MSB:0]  sel_porta;
   input  [RF_PORTB_MSB:0]   sel_portb;
   input  [RF_PORTRD_MSB:0]  sel_rd;
   input  [RF_PORTRD_MSB:0]  sel_rd2;
   // Data ports
   output  [DATA_MSB:0]      porta;
   output  [DATA_MSB:0]      portb;
   input  [DATA_MSB:0]       rd;
   input                               write_rd;
   input  [DATA_MSB:0]       rd2;
   input                               write_rd2;
   input                               hreadyd;

   // only when using micro-rom code
   // we use x0 as a temporary register
   input                               allow_hidden_use_of_x0;


   input                               clk;
   input                               rst_n;

   /*AUTOINPUT*/

  /*AUTOOUTPUT*/

  /*AUTOREG*/
  // Beginning of automatic regs (for this module's undeclared outputs)

   wire [DATA_MSB:0]          sel_porta_oh;
   wire [DATA_MSB:0]          sel_portb_oh;
   wire [DATA_MSB:0]          sel_rd_oh;
   wire [DATA_MSB:0]          sel_rd2_oh;



   assign sel_porta_oh = (sel_porta == 0) ? 1 : (1'b1 << sel_porta);
   assign sel_portb_oh = (sel_portb == 0) ? 1 : (1'b1 << sel_portb);
   assign sel_rd_oh  = (sel_rd == 0) ? 1 : (1'b1 << sel_rd);
   assign sel_rd2_oh = (sel_rd2 == 0) ? 1 : (1'b1 << sel_rd2);




regfile_core U_REG_FILE_CORE(/*AUTOINST*/
                                                            //Selection
                                                            .sel_porta(sel_porta_oh),
                                                            .sel_portb(sel_portb_oh),
                                                            .sel_rd(sel_rd_oh),
                                                            .sel_rd2(sel_rd2_oh),
                                                            //Data ports
                                                            .porta(porta),
                                                            .portb(portb),
                                                            .rd(rd),
                                                            .write_rd(write_rd),
                                                            .rd2(rd2),
                                                            .write_rd2(write_rd2),
                                                            .clk(clk),
                                                            .rst_n(rst_n),
                                                            .allow_hidden_use_of_x0(
                                                                allow_hidden_use_of_x0)



);





endmodule // regfile
```

Listing 2: Register file core module of the One-hot encoded design

```
module regfile_core (/*AUTOARG*/
   porta, portb,

   // Inputs
  sel_porta,sel_portb,sel_rd,sel_rd2,rd,write_rd,rd2,write_rd2,clk,rst_n,
      allow_hidden_use_of_x0

   );

   parameter NUM_REGS=32;
`include "parameters.v"

  // Selection
   input [DATA_MSB:0]   sel_porta;
   input [DATA_MSB:0]   sel_portb;
   input [DATA_MSB:0]   sel_rd;
   input [DATA_MSB:0]   sel_rd2;

   // Data ports
   output [DATA_MSB:0]       porta;
   output [DATA_MSB:0]       portb;
   input [DATA_MSB:0]        rd;
   input                               write_rd;
   input [DATA_MSB:0]        rd2;
   input                               write_rd2;
   //input                               hreadyd;
   input                               clk;
   input                               rst_n;
   input                               allow_hidden_use_of_x0;
   /*AUTOINPUT*/

  /*AUTOOUTPUT*/

  /*AUTOREG*/
  // Beginning of automatic regs (for this module's undeclared outputs)
  wire [DATA_MSB:0] porta;
  wire [DATA_MSB:0] portb;

  // End of automatics
  genvar i;
  /*AUTOWIRE*/



  reg [DATA_MSB:0] regfile [NUM_REGS-1:0];


   assign porta = (sel_porta[0] !=  1) || allow_hidden_use_of_x0 ?
                                                ({32{sel_porta[0]}}&regfile[0]) |
                                                ({32{sel_porta[1]}}&regfile[1]) |
                                                ({32{sel_porta[2]}}&regfile[2]) |
                                                ({32{sel_porta[3]}}&regfile[3]) |
                                                ({32{sel_porta[4]}}&regfile[4]) |
                                                ({32{sel_porta[5]}}&regfile[5]) |
                                                ({32{sel_porta[6]}}&regfile[6]) |
                                                ({32{sel_porta[7]}}&regfile[7]) |
                                                ({32{sel_porta[8]}}&regfile[8]) |
                                                ({32{sel_porta[9]}}&regfile[9]) |
                                                ({32{sel_porta[10]}}&regfile
                                                  [10])|
                                              ({32{sel_porta[11]}}&regfile
                                                  [11])|
                                              ({32{sel_porta[12]}}&regfile
                                                  [12])|
                                              ({32{sel_porta[13]}}&regfile
                                                  [13])|
                                              ({32{sel_porta[14]}}&regfile
                                                  [14])|
                                              ({32{sel_porta[15]}}&regfile
                                                  [15])|
                                              ({32{sel_porta[16]}}&regfile
                                                  [16])|
                                              ({32{sel_porta[17]}}&regfile
                                                  [17])|
                                              ({32{sel_porta[18]}}&regfile
                                                  [18])|
                                              ({32{sel_porta[19]}}&regfile
                                                  [19])|
                                              ({32{sel_porta[20]}}&regfile
                                                  [20])|
                                              ({32{sel_porta[21]}}&regfile
                                                  [21])|
                                              ({32{sel_porta[22]}}&regfile
                                                  [22])|
                                              ({32{sel_porta[23]}}&regfile
                                                  [23])|
                                              ({32{sel_porta[24]}}&regfile
                                                  [24])|
                                              ({32{sel_porta[25]}}&regfile
                                                  [25])|
                                              ({32{sel_porta[26]}}&regfile
                                                  [26])|
                                              ({32{sel_porta[27]}}&regfile
                                                  [27])|
                                              ({32{sel_porta[28]}}&regfile
                                                  [28])|
                                              ({32{sel_porta[29]}}&regfile
                                                  [29])|
                                              ({32{sel_porta[30]}}&regfile
                                                  [30])|
                                              ({32{sel_porta[31]}}&regfile
                                                  [31]) : 0;

   assign portb = (sel_portb[0] != 1) || allow_hidden_use_of_x0 ?
                                              ({32{sel_porta[0]}}&regfile[0])|
                                              ({32{sel_portb[1]}}&regfile[1])|
                                              ({32{sel_portb[2]}}&regfile[2])|
                                              ({32{sel_portb[3]}}&regfile[3])|
                                              ({32{sel_portb[4]}}&regfile[4])|
                                              ({32{sel_portb[5]}}&regfile[5])|
                                              ({32{sel_portb[6]}}&regfile[6])|
                                              ({32{sel_portb[7]}}&regfile[7])|
                                              ({32{sel_portb[8]}}&regfile[8])|
                                              ({32{sel_portb[9]}}&regfile[9])|
                                              ({32{sel_portb[10]}}&regfile
                                                  [10])|
                                              ({32{sel_portb[11]}}&regfile
                                                  [11])|
                                              ({32{sel_portb[12]}}&regfile
                                                  [12])|
                                                  ({32{sel_portb[13]}}&regfile
                                                      [13])|
                                                  ({32{sel_portb[14]}}&regfile
                                                      [14])|
                                                  ({32{sel_portb[15]}}&regfile
                                                      [15])|
                                                  ({32{sel_portb[16]}}&regfile
                                                      [16])|
                                                  ({32{sel_portb[17]}}&regfile
                                                      [17])|
                                                  ({32{sel_portb[18]}}&regfile
                                                      [18])|
                                                  ({32{sel_portb[19]}}&regfile
                                                      [19])|
                                                  ({32{sel_portb[20]}}&regfile
                                                      [20])|
                                                  ({32{sel_portb[21]}}&regfile
                                                      [21])|
                                                  ({32{sel_portb[22]}}&regfile
                                                      [22])|
                                                  ({32{sel_portb[23]}}&regfile
                                                      [23])|
                                                  ({32{sel_portb[24]}}&regfile
                                                      [24])|
                                                  ({32{sel_portb[25]}}&regfile
                                                      [25])|
                                                  ({32{sel_portb[26]}}&regfile
                                                      [26])|
                                                  ({32{sel_portb[27]}}&regfile
                                                      [27])|
                                                  ({32{sel_portb[28]}}&regfile
                                                      [28])|
                                                  ({32{sel_portb[29]}}&regfile
                                                      [29])|
                                                  ({32{sel_portb[30]}}&regfile
                                                      [30])|
                                                  ({32{sel_portb[31]}}&regfile
                                                      [31]) : 0 ;


generate
   for (i = 0; i < 32 ; i = i + 1) begin : loop0
      always @(posedge clk or negedge rst_n) begin
   if (~rst_n) begin
     regfile[i] <= 32'h0;
   end
   else begin
      if( (sel_rd[i]&write_rd&sel_rd!=0) & (sel_rd[0]!=1 |
          allow_hidden_use_of_x0) ) begin
        regfile[i] <= rd;
     end
      else if( (sel_rd2[i]&write_rd2&sel_rd2!=0) & (sel_rd2[0]!=1 |
          allow_hidden_use_of_x0)  ) begin
         regfile[i] <= rd2;
     end

   end
      end
   end
endgenerate
```

```
wire  [31:0] x0 ;
wire  [31:0] x1 ;
wire  [31:0] x2 ;
wire  [31:0] x3 ;
wire  [31:0] x4 ;
wire  [31:0] x5 ;
wire  [31:0] x6 ;
wire  [31:0] x7 ;
wire  [31:0] x8 ;
wire  [31:0] x9 ;
wire  [31:0] x10;
wire  [31:0] x11;
wire  [31:0] x12;
wire  [31:0] x13;
wire  [31:0] x14;
wire  [31:0] x15;
wire  [31:0] x16;
wire  [31:0] x17;
wire  [31:0] x18;
wire  [31:0] x19;
wire  [31:0] x20;
wire  [31:0] x21;
wire  [31:0] x22;
wire  [31:0] x23;
wire  [31:0] x24;
wire  [31:0] x25;
wire  [31:0] x26;
wire  [31:0] x27;
wire  [31:0] x28;
wire  [31:0] x29;
wire  [31:0] x30;
wire  [31:0] x31;

   wire [31:0] ra ;
   wire [31:0] s0 ;
   wire [31:0] fp ;
   wire [31:0] s1 ;
   wire [31:0] s2 ;
   wire [31:0] s3 ;
   wire [31:0] s4 ;
   wire [31:0] s5 ;
   wire [31:0] s6 ;
   wire [31:0] s7;
   wire [31:0] s8;
   wire [31:0] s9;
   wire [31:0] s10;
   wire [31:0] s11;
   wire [31:0] sp;
   wire [31:0] v0;
   wire [31:0] v1;
   wire [31:0] a0;
   wire [31:0] a1;
   wire [31:0] a2;
   wire [31:0] a3;
   wire [31:0] a4;
   wire [31:0] a5;
   wire [31:0] a6;
   wire [31:0] a7;
```

```
  wire [31:0] t0;
  wire [31:0] t1;
  wire [31:0] t2;
  wire [31:0] t3;
  wire [31:0] t4;
  wire [31:0] t5;
  wire [31:0] t6;
  wire [31:0] tp;
  wire [31:0] gp;




assign     x0 =   regfile[0];
assign     x1 =   regfile[1];
assign     x2 =   regfile[2];
assign     x3 =   regfile[3];
assign     x4 =   regfile[4];
assign     x5 =   regfile[5];
assign     x6 =   regfile[6];
assign     x7 =   regfile[7];
assign     x8 =   regfile[8];
assign     x9 =   regfile[9];
assign     x10 =  regfile[10];
assign     x11 =  regfile[11];
assign     x12 =  regfile[12];
assign     x13 =  regfile[13];
assign     x14 =  regfile[14];
assign     x15 =  regfile[15];
assign     x16 =  regfile[16];
assign     x17 =  regfile[17];
assign     x18 =  regfile[18];
assign     x19 =  regfile[19];
assign     x20 =  regfile[20];
assign     x21 =  regfile[21];
assign     x22 =  regfile[22];
assign     x23 =  regfile[23];
assign     x24 =  regfile[24];
assign     x25 =  regfile[25];
assign     x26 =  regfile[26];
assign     x27 =  regfile[27];
assign     x28 =  regfile[28];
assign     x29 =  regfile[29];
assign     x30 =  regfile[30];
assign     x31 =  regfile[31];

  assign     ra =  regfile[1];
  assign     sp =  regfile[2];
  assign     gp =  regfile[3];
  assign     tp =  regfile[4];
  assign     t0 =  regfile[5];
  assign     t1 =  regfile[6];
  assign     t2 =  regfile[7];
  assign     s0 =  regfile[8];
  assign     fp =  regfile[8];
  assign     s1 =  regfile[9];
  assign     a0 =  regfile[10];
  assign    a1 =  regfile[11];
  assign    a2 =  regfile[12];
  assign    a3 =  regfile[13];
  assign    a4 =  regfile[14];
  assign    a5 =  regfile[15];
  assign    a6 =  regfile[16];
  assign    a7 =  regfile[17];
  assign    s2 =  regfile[18];
  assign    s3 =  regfile[19];
  assign    s4 =  regfile[20];
  assign    s5 =  regfile[21];
  assign    s6 =  regfile[22];
  assign    s7 =  regfile[23];
  assign    s8 =  regfile[24];
  assign    s9 =  regfile[25];
  assign    s10 =   regfile[26];
  assign    s11=  regfile[27];
  assign    t3 =  regfile[28];
  assign    t4 =  regfile[29];
  assign    t5 =  regfile[30];
  assign    t6 =  regfile[31];

  integer     jj;
  /* verilator lint_off STMTDLY */
  initial begin
    #1; // spyglass disable CheckDelayTimescale-ML
    for(jj=0;jj<32;jj=jj+1) begin
      regfile[jj] = 32'b0;
    end
  end
endmodule // nregfile_core
```

Listing 3: Register file core module of the Shared Master-latch design

```
module regfile_core (/*AUTOARG*/
   porta, portb,

   // Inputs
  sel_porta,sel_portb,sel_rd,sel_rd2,rd,write_rd,rd2,write_rd2,clk,rst_n,
      allow_hidden_use_of_x0

   );

   parameter NUM_REGS=32;
`include "parameters.v"

  // Selection
   input [DATA_MSB:0]   sel_porta;
   input [DATA_MSB:0]   sel_portb;
   input [DATA_MSB:0]   sel_rd;
   input [DATA_MSB:0]   sel_rd2;

   // Data ports
   output [DATA_MSB:0]      porta;
   output [DATA_MSB:0]      portb;
   input [DATA_MSB:0]       rd;
   input                              write_rd;
   input [DATA_MSB:0]       rd2;
   input                              write_rd2;
   //input                              hreadyd;
   input                              clk;
   input                              rst_n;
  input                               allow_hidden_use_of_x0;
   /*AUTOINPUT*/

  /*AUTOOUTPUT*/

  /*AUTOREG*/
  // Beginning of automatic regs (for this module's undeclared outputs)
  wire [DATA_MSB:0] porta;
  wire [DATA_MSB:0] portb;

  // End of automatics
  genvar i;
  /*AUTOWIRE*/

   //Q1 = sel_porta[1] & clk;

   wire  clk_inv;


   reg [DATA_MSB:0]  regfile_temp;
   reg [DATA_MSB:0]  regfile_temp2;
   reg [DATA_MSB:0]  regfile [NUM_REGS-1:0];
   reg [DATA_MSB:0]   sel_rd_tmp;
   reg [DATA_MSB:0]   sel_rd2_tmp;
   reg                        write_rd_tmp;
   reg                        write_rd2_tmp;

//always @(posedge clk_inv

assign clk_inv = ~clk;
always @(*) begin
   if(clk_inv) begin
   sel_rd_tmp <= sel_rd;
   sel_rd2_tmp <= sel_rd2;
   write_rd_tmp <= write_rd;
   write_rd2_tmp <= write_rd2;
   end
end
always @(*) begin
   if (write_rd&clk_inv) begin
      regfile_temp <=rd;
   end
end // always @ begin

always @(*) begin
   if(write_rd2&clk_inv) begin
      regfile_temp2 = rd2;
   end
end // always @ begin



   generate
      for (i=0; i<32; i=i+1) begin : loop
   always @(*) begin
      if(~rst_n) begin
         regfile[i] = 32'h0;
      end //if
      else begin
         if( (sel_rd_tmp[i]&write_rd_tmp&clk) & (sel_rd_tmp[0]!= 1 |
              allow_hidden_use_of_x0) ) begin
                    regfile[i] = regfile_temp;

         end //if
         else if( (sel_rd2_tmp[i]&write_rd2_tmp&clk) & (sel_rd2_tmp[0]!=1
               | allow_hidden_use_of_x0)    ) begin
                    regfile[i] = regfile_temp2;
         end //elseif
      end //else
         end //always
      end //for
endgenerate


assign porta = (sel_porta[0] != 1) || allow_hidden_use_of_x0  ?
                                                  ({32{sel_porta[0]}}&regfile[0]) |
                                                  ({32{sel_porta[1]}}&regfile[1]) |
                                                  ({32{sel_porta[2]}}&regfile[2]) |
                                                  ({32{sel_porta[3]}}&regfile[3]) |
                                                  ({32{sel_porta[4]}}&regfile[4]) |
                                                  ({32{sel_porta[5]}}&regfile[5]) |
                                                  ({32{sel_porta[6]}}&regfile[6]) |
                                                  ({32{sel_porta[7]}}&regfile[7]) |
                                                  ({32{sel_porta[8]}}&regfile[8]) |
                                                  ({32{sel_porta[9]}}&regfile[9]) |
                                                  ({32{sel_porta[10]}}&regfile
                                                       [10]) |
                                                  ({32{sel_porta[11]}}&regfile
                                                       [11]) |
                                                  ({32{sel_porta[12]}}&regfile
                                                       [12]) |
                                                  ({32{sel_porta[13]}}&regfile
                                                       [13]) |
                                            ({32{sel_porta[14]}}&regfile
                                                [14])|
                                            ({32{sel_porta[15]}}&regfile
                                                [15])|
                                            ({32{sel_porta[16]}}&regfile
                                                [16])|
                                            ({32{sel_porta[17]}}&regfile
                                                [17])|
                                            ({32{sel_porta[18]}}&regfile
                                                [18])|
                                            ({32{sel_porta[19]}}&regfile
                                                [19])|
                                            ({32{sel_porta[20]}}&regfile
                                                [20])|
                                            ({32{sel_porta[21]}}&regfile
                                                [21])|
                                            ({32{sel_porta[22]}}&regfile
                                                [22])|
                                            ({32{sel_porta[23]}}&regfile
                                                [23])|
                                            ({32{sel_porta[24]}}&regfile
                                                [24])|
                                            ({32{sel_porta[25]}}&regfile
                                                [25])|
                                            ({32{sel_porta[26]}}&regfile
                                                [26])|
                                            ({32{sel_porta[27]}}&regfile
                                                [27])|
                                            ({32{sel_porta[28]}}&regfile
                                                [28])|
                                            ({32{sel_porta[29]}}&regfile
                                                [29])|
                                            ({32{sel_porta[30]}}&regfile
                                                [30])|
                                            ({32{sel_porta[31]}}&regfile
                                                [31]) : 0;

assign portb = (sel_portb[0] != 1) || allow_hidden_use_of_x0 ?
                                            ({32{sel_portb[0]}}&regfile[0])|
                                            ({32{sel_portb[1]}}&regfile[1])|
                                            ({32{sel_portb[2]}}&regfile[2])|
                                            ({32{sel_portb[3]}}&regfile[3])|
                                            ({32{sel_portb[4]}}&regfile[4])|
                                            ({32{sel_portb[5]}}&regfile[5])|
                                            ({32{sel_portb[6]}}&regfile[6])|
                                            ({32{sel_portb[7]}}&regfile[7])|
                                            ({32{sel_portb[8]}}&regfile[8])|
                                            ({32{sel_portb[9]}}&regfile[9])|
                                            ({32{sel_portb[10]}}&regfile
                                                [10])|
                                            ({32{sel_portb[11]}}&regfile
                                                [11])|
                                            ({32{sel_portb[12]}}&regfile
                                                [12])|
                                            ({32{sel_portb[13]}}&regfile
                                                [13])|
                                            ({32{sel_portb[14]}}&regfile
                                                [14])|
                                            ({32{sel_portb[15]}}&regfile
                                                [15])|
                                            ({32{sel_portb[16]}}&regfile
      [16])|
({32{sel_portb[17]}}&regfile
      [17])|
({32{sel_portb[18]}}&regfile
      [18])|
({32{sel_portb[19]}}&regfile
      [19])|
({32{sel_portb[20]}}&regfile
      [20])|
({32{sel_portb[21]}}&regfile
      [21])|
({32{sel_portb[22]}}&regfile
      [22])|
({32{sel_portb[23]}}&regfile
      [23])|
({32{sel_portb[24]}}&regfile
      [24])|
({32{sel_portb[25]}}&regfile
      [25])|
({32{sel_portb[26]}}&regfile
      [26])|
({32{sel_portb[27]}}&regfile
      [27])|
({32{sel_portb[28]}}&regfile
      [28])|
({32{sel_portb[29]}}&regfile
      [29])|
({32{sel_portb[30]}}&regfile
      [30])|
({32{sel_portb[31]}}&regfile
      [31]) : 0 ;
```

```
wire   [31:0]  x0 ;
wire   [31:0]  x1 ;
wire   [31:0]  x2 ;
wire   [31:0]  x3 ;
wire   [31:0]  x4 ;
wire   [31:0]  x5 ;
wire   [31:0]  x6 ;
wire   [31:0]  x7 ;
wire   [31:0]  x8 ;
wire   [31:0]  x9 ;
wire   [31:0]  x10;
wire   [31:0]  x11;
wire   [31:0]  x12;
wire   [31:0]  x13;
wire   [31:0]  x14;
wire   [31:0]  x15;
wire   [31:0]  x16;
wire   [31:0]  x17;
wire   [31:0]  x18;
wire   [31:0]  x19;
wire   [31:0]  x20;
wire   [31:0]  x21;
wire   [31:0]  x22;
wire   [31:0]  x23;
wire   [31:0]  x24;
wire   [31:0]  x25;
wire   [31:0]  x26;
wire   [31:0]  x27;
wire   [31:0]  x28;
wire   [31:0]  x29;
wire   [31:0]  x30;
wire   [31:0]  x31;


    wire  [31:0]  ra  ;
    wire  [31:0]  s0  ;
    wire  [31:0]  fp  ;
    wire  [31:0]  s1  ;
    wire  [31:0]  s2  ;
    wire  [31:0]  s3  ;
    wire  [31:0]  s4  ;
    wire  [31:0]  s5  ;
    wire  [31:0]  s6  ;
    wire  [31:0]  s7;
    wire  [31:0]  s8;
    wire  [31:0]  s9;
    wire  [31:0]  s10;
    wire  [31:0]  s11;
    wire  [31:0]  sp;
    wire  [31:0]  v0;
    wire  [31:0]  v1;
    wire  [31:0]  a0;
    wire  [31:0]  a1;
    wire  [31:0]  a2;
    wire  [31:0]  a3;
    wire  [31:0]  a4;
    wire  [31:0]  a5;
    wire  [31:0]  a6;
    wire  [31:0]  a7;
    wire  [31:0]  t0;
    wire  [31:0]  t1;
    wire  [31:0]  t2;
    wire  [31:0]  t3;
    wire  [31:0]  t4;
    wire  [31:0]  t5;
    wire  [31:0]  t6;
    wire  [31:0]  tp;
    wire  [31:0]  gp;




assign      x0 =    regfile[0];
assign      x1 =    regfile[1];
assign      x2 =    regfile[2];
assign      x3 =    regfile[3];
assign      x4 =    regfile[4];
assign      x5 =    regfile[5];
assign      x6 =    regfile[6];
assign      x7 =    regfile[7];
assign      x8 =    regfile[8];
assign      x9 =    regfile[9];
assign      x10 =  regfile[10];
assign      x11 =  regfile[11];
assign      x12 =  regfile[12];
assign      x13 =  regfile[13];
assign      x14 =  regfile[14];
assign      x15 =  regfile[15];
assign      x16 =  regfile[16];
assign      x17 =  regfile[17];
assign      x18 =  regfile[18];
assign      x19 =  regfile[19];
assign      x20 =  regfile[20];
assign      x21 =  regfile[21];
assign      x22 =  regfile[22];
assign      x23 =  regfile[23];
assign      x24 =  regfile[24];
assign      x25 =  regfile[25];
assign      x26 =  regfile[26];
assign      x27 =  regfile[27];
assign      x28 =  regfile[28];
assign      x29 =  regfile[29];
assign      x30 =  regfile[30];
assign      x31 =  regfile[31];


   assign      ra =  regfile[1];
   assign      sp =  regfile[2];
   assign      gp =  regfile[3];
   assign      tp =  regfile[4];
   assign      t0 =  regfile[5];
   assign      t1 =  regfile[6];
   assign      t2 =  regfile[7];
   assign      s0 =  regfile[8];
   assign      fp =  regfile[8];
   assign      s1 =  regfile[9];
   assign      a0 =  regfile[10];
   assign      a1 =  regfile[11];
   assign      a2 =  regfile[12];
   assign      a3 =  regfile[13];
   assign      a4 =  regfile[14];
   assign      a5 =  regfile[15];
   assign      a6 =  regfile[16];
   assign      a7 =  regfile[17];
   assign      s2 =  regfile[18];
   assign      s3 =  regfile[19];
   assign      s4 =  regfile[20];
   assign      s5 =  regfile[21];
   assign      s6 =  regfile[22];
   assign      s7 =  regfile[23];
   assign      s8 =  regfile[24];
   assign      s9 =  regfile[25];
   assign      s10 =   regfile[26];
   assign      s11=  regfile[27];
   assign      t3 =  regfile[28];
   assign      t4 =  regfile[29];
   assign      t5 =  regfile[30];
   assign      t6 =  regfile[31];

   integer      jj;
   /* verilator lint_off STMTDLY */
   initial begin
      #1; // spyglass disable CheckDelayTimescale-ML
```

```
        for(jj=0;jj<32;jj=jj+1) begin
            regfile[jj] = 32'b0;
            regfile_temp2= 32'b0;
            regfile_temp= 32'b0;
        end
    end
endmodule // regfile_core
```

Listing 4: Register file core module of the Shared Slave-latch design

```
module regfile_core (/*AUTOARG*/
   porta, portb,

   // Inputs
  sel_porta, sel_portb, sel_rd, sel_rd2, rd, write_rd, rd2, write_rd2, clk, rst_n,
      allow_hidden_use_of_x0

   );

   parameter NUM_REGS=32;
`include "parameters.v"

  // Selection
   input [DATA_MSB:0]   sel_porta;
   input [DATA_MSB:0]   sel_portb;
   input [DATA_MSB:0]   sel_rd;
   input [DATA_MSB:0]   sel_rd2;

   // Data ports
   output [DATA_MSB:0]       porta;
   output [DATA_MSB:0]       portb;
   input [DATA_MSB:0]        rd;
   input                               write_rd;
   input [DATA_MSB:0]        rd2;
   input                               write_rd2;
   //input                               hreadyd;
   input                               clk;
   input                               rst_n;
   input             allow_hidden_use_of_x0;
   /*AUTOINPUT*/

  /*AUTOOUTPUT*/

  /*AUTOREG*/
  // Beginning of automatic regs (for this module's undeclared outputs)
  reg [DATA_MSB:0] porta;
  reg [DATA_MSB:0] portb;

  // End of automatics
  genvar i;
  /*AUTOWIRE*/


   wire  clk_inv;
   reg [DATA_MSB:0] regfile [NUM_REGS-1:0];
   reg  [32:0]           test;


   assign  clk_inv = ~clk;



   generate
      for (i=0; i<32; i=i+1) begin : loop
   always @(*) begin
      if(~rst_n) begin
         regfile[i] = 32'h0;
      end //if
      else begin
         if( (sel_rd[i]&write_rd&clk_inv) & (allow_hidden_use_of_x0 | !
                sel_rd[0] )) begin
                    regfile[i] = rd;

            end //if
            else if( (sel_rd2[i]&write_rd2&clk_inv) & (
                allow_hidden_use_of_x0 | !sel_rd2[0])) begin
                    regfile[i] = rd2;

            end //elseif
        end //else
            end //always
        end //for
endgenerate


    always @(*)begin
        if(clk) begin
        porta <= (sel_porta[0] != 1) | allow_hidden_use_of_x0 ?
                                                    ({32{sel_porta[0]}}&regfile[0])|
                                                    ({32{sel_porta[1]}}&regfile[1])|
                                                    ({32{sel_porta[2]}}&regfile[2])|
                                                    ({32{sel_porta[3]}}&regfile[3])|
                                                    ({32{sel_porta[4]}}&regfile[4])|
                                                    ({32{sel_porta[5]}}&regfile[5])|
                                                    ({32{sel_porta[6]}}&regfile[6])|
                                                    ({32{sel_porta[7]}}&regfile[7])|
                                                    ({32{sel_porta[8]}}&regfile[8])|
                                                    ({32{sel_porta[9]}}&regfile[9])|
                                                    ({32{sel_porta[10]}}&regfile
                                                        [10])|
                                                    ({32{sel_porta[11]}}&regfile
                                                        [11])|
                                                    ({32{sel_porta[12]}}&regfile
                                                        [12])|
                                                    ({32{sel_porta[13]}}&regfile
                                                        [13])|
                                                    ({32{sel_porta[14]}}&regfile
                                                        [14])|
                                                    ({32{sel_porta[15]}}&regfile
                                                        [15])|
                                                    ({32{sel_porta[16]}}&regfile
                                                        [16])|
                                                    ({32{sel_porta[17]}}&regfile
                                                        [17])|
                                                    ({32{sel_porta[18]}}&regfile
                                                        [18])|
                                                    ({32{sel_porta[19]}}&regfile
                                                        [19])|
                                                    ({32{sel_porta[20]}}&regfile
                                                        [20])|
                                                    ({32{sel_porta[21]}}&regfile
                                                        [21])|
                                                    ({32{sel_porta[22]}}&regfile
                                                        [22])|
                                                    ({32{sel_porta[23]}}&regfile
                                                        [23])|
                                                    ({32{sel_porta[24]}}&regfile
                                                        [24])|
                                                    ({32{sel_porta[25]}}&regfile
                                                        [25])|
                                                    ({32{sel_porta[26]}}&regfile
                                                      [26]) |
                                                 ({32{sel_porta[27]}}& regfile
                                                      [27]) |
                                                 ({32{sel_porta[28]}}& regfile
                                                      [28]) |
                                                 ({32{sel_porta[29]}}& regfile
                                                      [29]) |
                                                 ({32{sel_porta[30]}}& regfile
                                                      [30]) |
                                                 ({32{sel_porta[31]}}& regfile
                                                      [31])  :  0;
end

   end // always @ begin
  always @(*)begin
     if(clk) begin
        portb <= (sel_portb[0] != 1) | allow_hidden_use_of_x0 ?
                                                 ({32{sel_porta[0]}}& regfile[0]) |
                                                 ({32{sel_portb[1]}}& regfile[1]) |
                                                 ({32{sel_portb[2]}}& regfile[2]) |
                                                 ({32{sel_portb[3]}}& regfile[3]) |
                                                 ({32{sel_portb[4]}}& regfile[4]) |
                                                 ({32{sel_portb[5]}}& regfile[5]) |
                                                 ({32{sel_portb[6]}}& regfile[6]) |
                                                 ({32{sel_portb[7]}}& regfile[7]) |
                                                 ({32{sel_portb[8]}}& regfile[8]) |
                                                 ({32{sel_portb[9]}}& regfile[9]) |
                                                 ({32{sel_portb[10]}}& regfile
                                                      [10]) |
                                                 ({32{sel_portb[11]}}& regfile
                                                      [11]) |
                                                 ({32{sel_portb[12]}}& regfile
                                                      [12]) |
                                                 ({32{sel_portb[13]}}& regfile
                                                      [13]) |
                                                 ({32{sel_portb[14]}}& regfile
                                                      [14]) |
                                                 ({32{sel_portb[15]}}& regfile
                                                      [15]) |
                                                 ({32{sel_portb[16]}}& regfile
                                                      [16]) |
                                                 ({32{sel_portb[17]}}& regfile
                                                      [17]) |
                                                 ({32{sel_portb[18]}}& regfile
                                                      [18]) |
                                                 ({32{sel_portb[19]}}& regfile
                                                      [19]) |
                                                 ({32{sel_portb[20]}}& regfile
                                                      [20]) |
                                                 ({32{sel_portb[21]}}& regfile
                                                      [21]) |
                                                 ({32{sel_portb[22]}}& regfile
                                                      [22]) |
                                                 ({32{sel_portb[23]}}& regfile
                                                      [23]) |
                                                 ({32{sel_portb[24]}}& regfile
                                                      [24]) |
                                                 ({32{sel_portb[25]}}& regfile
                                                      [25]) |
                                                 ({32{sel_portb[26]}}& regfile
                                                      [26]) |
                                                    ({32{sel_portb[27]}}&regfile
                                                         [27])|
                                                    ({32{sel_portb[28]}}&regfile
                                                         [28])|
                                                    ({32{sel_portb[29]}}&regfile
                                                         [29])|
                                                    ({32{sel_portb[30]}}&regfile
                                                         [30])|
                                                    ({32{sel_portb[31]}}&regfile
                                                         [31]) : 0 ;
end
end // always @ begin


// synthesis translate_off
wire  [31:0] x0  ;
wire  [31:0] x1  ;
wire  [31:0] x2  ;
wire  [31:0] x3  ;
wire  [31:0] x4  ;
wire  [31:0] x5  ;
wire  [31:0] x6  ;
wire  [31:0] x7  ;
wire  [31:0] x8  ;
wire  [31:0] x9  ;
wire  [31:0] x10;
wire  [31:0] x11;
wire  [31:0] x12;
wire  [31:0] x13;
wire  [31:0] x14;
wire  [31:0] x15;
wire  [31:0] x16;
wire  [31:0] x17;
wire  [31:0] x18;
wire  [31:0] x19;
wire  [31:0] x20;
wire  [31:0] x21;
wire  [31:0] x22;
wire  [31:0] x23;
wire  [31:0] x24;
wire  [31:0] x25;
wire  [31:0] x26;
wire  [31:0] x27;
wire  [31:0] x28;
wire  [31:0] x29;
wire  [31:0] x30;
wire  [31:0] x31;


   wire [31:0] ra  ;
   wire [31:0] s0  ;
   wire [31:0] fp  ;
   wire [31:0] s1  ;
   wire [31:0] s2  ;
   wire [31:0] s3  ;
   wire [31:0] s4  ;
   wire [31:0] s5  ;
   wire [31:0] s6  ;
   wire [31:0] s7;
   wire [31:0] s8;
   wire [31:0] s9;
   wire [31:0] s10;
   wire [31:0] s11;
   wire [31:0] sp;
   wire [31:0] v0;
   wire [31:0] v1;
   wire [31:0] a0;
   wire [31:0] a1;
   wire [31:0] a2;
   wire [31:0] a3;
   wire [31:0] a4;
   wire [31:0] a5;
   wire [31:0] a6;
   wire [31:0] a7;
   wire [31:0] t0;
   wire [31:0] t1;
   wire [31:0] t2;
   wire [31:0] t3;
   wire [31:0] t4;
   wire [31:0] t5;
   wire [31:0] t6;
   wire [31:0] tp;
   wire [31:0] gp;




assign     x0 =   regfile[0];
assign     x1 =   regfile[1];
assign     x2 =   regfile[2];
assign     x3 =   regfile[3];
assign     x4 =   regfile[4];
assign     x5 =   regfile[5];
assign     x6 =   regfile[6];
assign     x7 =   regfile[7];
assign     x8 =   regfile[8];
assign     x9 =   regfile[9];
assign     x10 =  regfile[10];
assign     x11 =  regfile[11];
assign     x12 =  regfile[12];
assign     x13 =  regfile[13];
assign     x14 =  regfile[14];
assign     x15 =  regfile[15];
assign     x16 =  regfile[16];
assign     x17 =  regfile[17];
assign     x18 =  regfile[18];
assign     x19 =  regfile[19];
assign     x20 =  regfile[20];
assign     x21 =  regfile[21];
assign     x22 =  regfile[22];
assign     x23 =  regfile[23];
assign     x24 =  regfile[24];
assign     x25 =  regfile[25];
assign     x26 =  regfile[26];
assign     x27 =  regfile[27];
assign     x28 =  regfile[28];
assign     x29 =  regfile[29];
assign     x30 =  regfile[30];
assign     x31 =  regfile[31];
```

```
  assign      ra =  regfile[1];
  assign      sp =  regfile[2];
  assign      gp =  regfile[3];
  assign      tp =  regfile[4];
  assign      t0 =  regfile[5];
  assign      t1 =  regfile[6];
  assign      t2 =  regfile[7];
  assign      s0 =  regfile[8];
  assign      fp =  regfile[8];
  assign      s1 =  regfile[9];
  assign      a0 =  regfile[10];
  assign      a1 =  regfile[11];
  assign      a2 =  regfile[12];
  assign      a3 =  regfile[13];
  assign      a4 =  regfile[14];
  assign      a5 =  regfile[15];
  assign      a6 =  regfile[16];
  assign      a7 =  regfile[17];
  assign      s2 =  regfile[18];
  assign      s3 =  regfile[19];
  assign      s4 =  regfile[20];
  assign      s5 =  regfile[21];
  assign      s6 =  regfile[22];
  assign      s7 =  regfile[23];
  assign      s8 =  regfile[24];
  assign      s9 =  regfile[25];
  assign      s10 = regfile[26];
  assign      s11=  regfile[27];
  assign      t3 =  regfile[28];
  assign      t4 =  regfile[29];
  assign      t5 =  regfile[30];
  assign      t6 =  regfile[31];

  integer      jj;
  /* verilator lint_off STMTDLY */
  initial begin
    #1; // spyglass disable CheckDelayTimescale-ML
    for(jj=0;jj<32;jj=jj+1) begin
      regfile[jj] = 32'b0;

    end
  end
 // synthesis translate_on
endmodule // regfile_core
```

Listing 5: Relative placement generator

```
import svgwrite
from svgwrite import cm, mm
dwg = svgwrite.Drawing('test.svg', profile='tiny')
dwg.add(dwg.line((0, 0), (10, 0), stroke=svgwrite.rgb(10, 10, 16, '%')))
dwg.add(dwg.text('Test', insert=(0, 0.2), fill='red'))
dwg.save()

class Relative_position:
        def __init__(self,position):
            if (position == "right"):
                self.x_adjust = 1
                self.y_adjust = 0
            elif (position == "left"):
                self.x_adjust = -1
                self.y_adjust = 0
            elif (position == "under"):
                self.x_adjust = 0
                self.y_adjust = -1
            elif (position == "over"):
                self.x_adjust = 0
                self.y_adjust = 1
            elif (position == "TLTL"):
                self.x_adjust = -1
                self.y_adjust = 1
            elif (position == "TLTR"):
                self.x_adjust = 0
                self.y_adjust = 1
            elif (position == "TLBL"):
                self.x_adjust = -1
                self.y_adjust = 0
            elif (position == "TRTL"):
                self.x_adjust = 0
                self.y_adjust = 1
            elif (position == "TRTR"):
                self.x_adjust = 1
                self.y_adjust = 1
            elif (position == "TRBR"):
                self.x_adjust = 1
                self.y_adjust = 0
            elif (position == "BLTL"):
                self.x_adjust = -1
                self.y_adjust = 0
            elif (position == "BLBL"):
                self.x_adjust = -1
                self.y_adjust = -1
            elif (position == "BLBR"):
                self.x_adjust = 0
                self.y_adjust = -1
            elif (position == "BRTR"):
                self.x_adjust = 1
                self.y_adjust = 0
            elif (position == "BRBL"):
                self.x_adjust = 0
                self.y_adjust = -1
            elif (position == "BRBR"):
                self.x_adjust = 1
                self.y_adjust = -1
            else:
                self.x_adjust = 0
                self.y_adjust = 0
```

```
class Cell:
        def __init__(self,name,height,width):
            self.name = name
            self.height = height
            self.width = width
            self.x = None
            self.y = None

        def relative_place(self,neighboor,position):
            self.x = neighboor.x + position.x_adjust
            self.y = neighboor.y + position.y_adjust

class Bit:
        def __init__(self,ffname):
            self.flop = Cell(name=ffname,height=5,width=50)
            self.flop.x=0
            self.flop.y=0
            self.mux = Cell(name="Mux",height=10,width=10)
            self.mux.relative_place(self.flop,Relative_position("right"))
        def printBit(self):
            dwg = svgwrite.Drawing(filename="test.svg")
            shapes = dwg.add(dwg.g(id='shapes', fill='red'))
            print("self.flop.x:␣" + str(self.flop.x) + "␣self.mux.x:␣"+
                str(self.mux.x)+"␣self.flop.y:␣" + str(self.flop.y) + "␣
                self.mux.y:␣" + str(self.mux.y))
            flop_xpos = 10*self.flop.x * mm
            flop_ypos = -10*self.flop.y * mm
            mux_xpos =  (10*self.mux.x  + self.flop.width-10)*mm
            mux_ypos = (-10*self.mux.y  )*mm
            text_flop_xpos = (10*self.flop.x+4) * mm
            text_flop_ypos = (-10*self.flop.y +5)* mm
            text_mux_xpos = (10*self.mux.x  + self.flop.width-8)*mm
            text_mux_ypos = (-10*self.mux.y +5)*mm
            shapes.add(dwg.rect(insert=(flop_xpos,flop_ypos), size=(self.
                flop.width * mm, self.flop.height * mm),fill='white',
                stroke='red', stroke_width=3))
            shapes.add(dwg.rect(insert=(mux_xpos,mux_ypos ), size=(self.
                mux.width * mm, self.mux.height * mm), fill='white',
                stroke='red', stroke_width=3))
            g = dwg.g(style="font-size:12;font-family:Comic␣Sans␣MS,␣
                Arial;font-weight:bold;font-style:␣oblique;stroke:␣black;
                stroke␣-␣width:␣1;fill:␣black")
            g.add(dwg.text(self.flop.name, insert=(text_flop_xpos,
                text_flop_ypos ))) # settings are valid for all text
                added to 'g'
            g.add(dwg.text(self.mux.name, insert=(text_mux_xpos,
                text_mux_ypos )))
            dwg.add(g)
            dwg.save()
        def relative_place(self,neighboor,position):
            if (position == "right"):
                self.flop.x = neighboor.mux.x + 1
                self.flop.y = neighboor.mux.y
                self.mux.x = self.flop.x +1
                self.mux.y = self.flop.y
            elif (position == "left"):
                self.flop.x = neighboor.flop.x-2
                self.flop.y = neighboor.flop.y
                self.mux.x = neighboor.flop.x-1
        self.mux.y = neighboor.flop.y
    elif (position == "under"):
        self.flop.x = neighboor.flop.x
        self.flop.y = neighboor.flop.y + 1
        self.mux.x = neighboor.mux.x
        self.mux.y = neighboor.mux.y +1
    elif (position == "over"):
        self.flop.x = neighboor.flop.x
        self.flop.y = neighboor.flop.y + 1
        self.mux.x = neighboor.mux.x
        self.mux.y = neighboor.mux.y +1
    elif (position == "TLTL"):
        self.flop.x = neighboor.flop.x - 2
        self.flop.y = neighboor.flop.y +1
        self.mux.x = neighboor.mux.x - 2
        self.mux.y = neighboor.mux.y +1
    elif (position == "TLTR"):
        self.flop.x = neighboor.flop.x
        self.flop.y = neighboor.flop.y + 1
        self.mux.x = neighboor.mux.x
        self.mux.y = neighboor.mux.y + 1
    elif (position == "TLBL"):
        self.flop.x = neighboor.flop.x - 2
        self.flop.y = neighboor.flop.y
        self.mux.x = neighboor.mux.x - 2
        self.mux.y = neighboor.mux.y
    elif (position == "TRTL"):
        self.flop.x = neighboor.flop.x
        self.flop.y = neighboor.flop.y + 1
        self.mux.x = neighboor.mux.x
        self.mux.y = neighboor.mux.y + 1
    elif (position == "TRTR"):
        self.flop.x = neighboor.flop.x + 2
        self.flop.y = neighboor.flop.y + 1
        self.mux.x = neighboor.mux.x + 2
        self.mux.y = neighboor.mux.y + 1
    elif (position == "TRBR"):
        self.flop.x = neighboor.flop.x + 2
        self.flop.y = neighboor.flop.y
        self.mux.x = neighboor.mux.x + 2
        self.mux.y = neighboor.mux.y
    elif (position == "BLTL"):
        self.flop.x = neighboor.flop.x - 2
        self.flop.y = neighboor.flop.y
        self.mux.x = neighboor.mux.x - 2
        self.mux.y = neighboor.mux.y
    elif (position == "BLBL"):
        self.flop.x = neighboor.flop.x - 2
        self.flop.y = neighboor.flop.y - 1
        self.mux.x = neighboor.mux.x - 2
        self.mux.y = neighboor.mux.y - 1
    elif (position == "BLBR"):
        self.flop.x = neighboor.flop.x
        self.flop.y = neighboor.flop.y - 1
        self.mux.x = neighboor.mux.x
        self.mux.y = neighboor.mux.y - 1
    elif (position == "BRTR"):
        self.flop.x = neighboor.flop.x + 2
        self.flop.y = neighboor.flop.y
        self.mux.x = neighboor.mux.x + 2
        self.mux.y = neighboor.mux.y
        elif (position == "BRBL"):
            self.flop.x = neighboor.flop.x
            self.flop.y = neighboor.flop.y - 1
            self.mux.x = neighboor.mux.x
            self.mux.y = neighboor.mux.y - 1
        elif (position == "BRBR"):
            self.flop.x = neighboor.flop.x + 2
            self.flop.y = neighboor.flop.y - 1
            self.mux.x = neighboor.mux.x + 2
            self.mux.y = neighboor.mux.y - 1
        else:
            self.flop.x = neighboor.flop.x
            self.flop.y = neighboor.flop.y
            self.mux.x = neighboor.mux.x
            self.mux.y = neighboor.mux.y

class FourBits:
    def __init__(self):
        self.bit0 = Bit("FF")
        self.bit1 = Bit("FF")
        self.bit2 = Bit("FF")
        self.bit3 = Bit("FF")
        self.cg = Cell(name="CG", height=10, width=10)
        self.cg.x = 0
        self.cg.y = 0
        self.bit0.flop.relative_place(self.cg,Relative_position("
            under"))
        self.bit0.mux.relative_place(self.bit0.flop,Relative_position
            ("right"))
        self.bit1.relative_place(self.bit0,"under")
        self.bit2.relative_place(self.bit1, "under")
        self.bit3.relative_place(self.bit2, "under")

    def printBits(self):
        dwg = svgwrite.Drawing(filename="test.svg")
        shapes = dwg.add(dwg.g(id='shapes', fill='red'))


        shapes.add(dwg.rect(insert=(self.cg.x*10*mm, self.cg.y*10*mm)
            , size=(self.cg.width * mm, self.cg.height * mm), fill='
            white', stroke='black', stroke_width=3))#cg
        shapes.add(dwg.rect(insert=(10 * self.bit0.flop.x * mm, -10 *
             self.bit0.flop.y * mm), size=(self.bit0.flop.width * mm,
             self.bit0.flop.height * mm), fill='white', stroke='black
            ', stroke_width=3))#ff0
        shapes.add(dwg.rect(insert=((10 * self.bit0.mux.x + self.bit0
            .flop.width - 10) * mm, (-10 * self.bit0.mux.y) * mm),
            size=(self.bit0.mux.width * mm, self.bit0.mux.height * mm
            ), fill='white', stroke='black', stroke_width=3))#mux0
        shapes.add(dwg.rect(insert=(10 * self.bit1.flop.x * mm, -10 *
             self.bit1.flop.y * mm), size=(self.bit1.flop.width * mm,
             self.bit1.flop.height * mm), fill='white', stroke='black
            ', stroke_width=3))#ff1
        shapes.add(dwg.rect(insert=((10 * self.bit1.mux.x + self.bit1
            .flop.width - 10) * mm, (-10 * self.bit1.mux.y) * mm),
            size=(self.bit1.mux.width * mm, self.bit1.mux.height * mm
            ),fill='white', stroke='black', stroke_width=3))##mux1
        shapes.add(dwg.rect(insert=(10 * self.bit2.flop.x * mm, -10 *
             self.bit2.flop.y * mm), size=(self.bit2.flop.width * mm,
             self.bit2.flop.height * mm),fill='white', stroke='black'
            , stroke_width=3))#ff2
        shapes.add(dwg.rect(insert=((10 * self.bit2.mux.x + self.bit2
            .flop.width - 10) * mm, (-10 * self.bit2.mux.y) * mm),
            size=(self.bit2.mux.width * mm, self.bit2.mux.height * mm
            ), fill='white', stroke='black', stroke_width=3))#mux2
        shapes.add(dwg.rect(insert=(10 * self.bit3.flop.x * mm, -10 *
             self.bit3.flop.y * mm), size=(self.bit3.flop.width * mm,
             self.bit3.flop.height * mm), fill='white', stroke='black
            ', stroke_width=3))#ff3
        shapes.add(dwg.rect(insert=((10 * self.bit3.mux.x + self.bit3
            .flop.width - 10) * mm, (-10 * self.bit3.mux.y) * mm),
            size=(self.bit3.mux.width * mm, self.bit3.mux.height * mm
            ), fill='white', stroke='black', stroke_width=3))#mux3
        g = dwg.g(
            style="font-size:12;font-family:Comic␣Sans␣MS,␣Arial;font
                -weight:bold;font-style:␣oblique;stroke:␣black;stroke
                ␣-␣width:␣1;fill:␣black")
        g.add(dwg.text(self.cg.name, insert=((self.cg.x*10+3)*mm
            ,(-10* self.cg.y +6)*mm )))
        g.add(dwg.text(self.bit0.flop.name,insert=((10 * self.bit0.
            flop.x + 4) * mm, (-10 * self.bit0.flop.y + 5) * mm))) #
             settings are valid for all text added to 'g'
        g.add(dwg.text(self.bit0.mux.name, insert=((10 * self.bit0.
            mux.x + self.bit0.flop.width - 8) * mm, (-10 * self.bit0.
            mux.y + 5) * mm)))
        g.add(dwg.text(self.bit1.flop.name,insert=((10 * self.bit1.
            flop.x + 4) * mm, (-10 * self.bit1.flop.y + 5) * mm))) #
             settings are valid for all text added to 'g'
        g.add(dwg.text(self.bit1.mux.name, insert=((10 * self.bit1.
            mux.x + self.bit1.flop.width - 8) * mm, (-10 * self.bit1.
            mux.y + 5) * mm)))
        g.add(dwg.text(self.bit2.flop.name,insert=((10 * self.bit2.
            flop.x + 4) * mm, (-10 * self.bit2.flop.y + 5) * mm))) #
             settings are valid for all text added to 'g'
        g.add(dwg.text(self.bit2.mux.name, insert=((10 * self.bit2.
            mux.x + self.bit2.flop.width - 8) * mm, (-10 * self.bit2.
            mux.y + 5) * mm)))
        g.add(dwg.text(self.bit3.flop.name,insert=((10 * self.bit3.
            flop.x + 4) * mm, (-10 * self.bit3.flop.y + 5) * mm))) #
             settings are valid for all text added to 'g'
        g.add(dwg.text(self.bit3.mux.name, insert=((10 * self.bit3.
            mux.x + self.bit3.flop.width - 8) * mm, (-10 * self.bit3.
            mux.y + 5) * mm)))
        dwg.add(g)
        dwg.save()


class Register:
    def __init__(self,pos,xpos):
        number = str(pos)
        self.bit0 = Bit("regfile_reg_"+number+"___0_")
        self.bit1 = Bit("regfile_reg_"+number+"___1_")
        self.bit2 = Bit("regfile_reg_"+number+"___2_")
        self.bit3 = Bit("regfile_reg_"+number+"___3_")
        self.bit4 = Bit("regfile_reg_"+number+"___4_")
        self.bit5 = Bit("regfile_reg_"+number+"___5_")
        self.bit6 = Bit("regfile_reg_"+number+"___6_")
        self.bit7 = Bit("regfile_reg_"+number+"___7_")
        self.bit8 = Bit("regfile_reg_"+number+"___8_")
        self.bit9 = Bit("regfile_reg_"+number+"___9_")
        self.bit10 = Bit("regfile_reg_"+number+"___10_")
        self.bit11 = Bit("regfile_reg_"+number+"___11_")
        self.bit12 = Bit("regfile_reg_"+number+"___12_")
        self.bit13 = Bit("regfile_reg_"+number+"___13_")
        self.bit14 = Bit("regfile_reg_"+number+"___14_")
        self.bit15 = Bit("regfile_reg_"+number+"___15_")
        self.bit16 = Bit("regfile_reg_"+number+"___16_")
        self.bit17 = Bit("regfile_reg_"+number+"___17_")
        self.bit18 = Bit("regfile_reg_"+number+"___18_")
        self.bit19 = Bit("regfile_reg_"+number+"___19_")
        self.bit20 = Bit("regfile_reg_"+number+"___20_")
        self.bit21 = Bit("regfile_reg_"+number+"___21_")
        self.bit22 = Bit("regfile_reg_"+number+"___22_")
        self.bit23 = Bit("regfile_reg_"+number+"___23_")
        self.bit24 = Bit("regfile_reg_"+number+"___24_")
        self.bit25 = Bit("regfile_reg_"+number+"___25_")
        self.bit26 = Bit("regfile_reg_"+number+"___26_")
        self.bit27 = Bit("regfile_reg_"+number+"___27_")
        self.bit28 = Bit("regfile_reg_"+number+"___28_")
        self.bit29 = Bit("regfile_reg_"+number+"___29_")
        self.bit30 = Bit("regfile_reg_"+number+"___30_")
        self.bit31 = Bit("regfile_reg_"+number+"___31_")
        self.bit0.flop.x = xpos
        self.bit0.flop.y = 0
        self.bit0.mux.relative_place(self.bit0.flop,Relative_position
            ("right"))
        self.bit1.relative_place(self.bit0,"under")
        self.bit2.relative_place(self.bit1, "under")
        self.bit3.relative_place(self.bit2, "under")
        self.bit4.relative_place(self.bit3,"under")
        self.bit5.relative_place(self.bit4, "under")
        self.bit6.relative_place(self.bit5, "under")
        self.bit7.relative_place(self.bit6,"under")
        self.bit8.relative_place(self.bit7, "under")
        self.bit9.relative_place(self.bit8, "under")
        self.bit10.relative_place(self.bit9,"under")
        self.bit11.relative_place(self.bit10, "under")
        self.bit12.relative_place(self.bit11, "under")
        self.bit13.relative_place(self.bit12,"under")
        self.bit14.relative_place(self.bit13, "under")
        self.bit15.relative_place(self.bit14, "under")
        self.bit16.relative_place(self.bit15,"under")
        self.bit17.relative_place(self.bit16, "under")
        self.bit18.relative_place(self.bit17, "under")
        self.bit19.relative_place(self.bit18,"under")
        self.bit20.relative_place(self.bit19, "under")
        self.bit21.relative_place(self.bit20, "under")
        self.bit22.relative_place(self.bit21,"under")
        self.bit23.relative_place(self.bit22, "under")
        self.bit24.relative_place(self.bit23, "under")
        self.bit25.relative_place(self.bit24, "under")
        self.bit26.relative_place(self.bit25, "under")
        self.bit27.relative_place(self.bit26,"under")
        self.bit28.relative_place(self.bit27, "under")
        self.bit29.relative_place(self.bit28, "under")
        self.bit30.relative_place(self.bit29,"under")
        self.bit31.relative_place(self.bit30, "under")
def printBits(self):
            dwg = svgwrite.Drawing(filename="test.svg")
            shapes = dwg.add(dwg.g(id='shapes', fill='red'))

            shapes.add(dwg.rect(insert=(10 * self.bit0.flop.x * mm, 5
                 * self.bit0.flop.y * mm), size=(self.bit0.flop.width
```

```
     * mm, self.bit0.flop.height * mm), fill='white',
    stroke='black', stroke_width=3))#ff0
shapes.add(dwg.rect(insert=(10 * self.bit1.flop.x * mm, 5
     * self.bit1.flop.y * mm), size=(self.bit1.flop.width
     * mm, self.bit1.flop.height * mm), fill='white',
    stroke='black', stroke_width=3))#ff1
shapes.add(dwg.rect(insert=(10 * self.bit2.flop.x * mm, 5
     * self.bit2.flop.y * mm), size=(self.bit2.flop.width
     * mm, self.bit2.flop.height * mm),fill='white',
    stroke='black', stroke_width=3))#ff2
shapes.add(dwg.rect(insert=(10 * self.bit3.flop.x * mm, 5
     * self.bit3.flop.y * mm), size=(self.bit3.flop.width
     * mm, self.bit3.flop.height * mm), fill='white',
    stroke='black', stroke_width=3))#ff3
shapes.add(dwg.rect(insert=(10 * self.bit4.flop.x * mm, 5
     * self.bit4.flop.y * mm), size=(self.bit4.flop.width
     * mm, self.bit4.flop.height * mm), fill='white',
    stroke='black', stroke_width=3))#ff4
shapes.add(dwg.rect(insert=(10 * self.bit5.flop.x * mm, 5
     * self.bit5.flop.y * mm), size=(self.bit5.flop.width
     * mm, self.bit5.flop.height * mm), fill='white',
    stroke='black', stroke_width=3))#ff5
shapes.add(dwg.rect(insert=(10 * self.bit6.flop.x * mm, 5
     * self.bit6.flop.y * mm), size=(self.bit6.flop.width
     * mm, self.bit6.flop.height * mm),fill='white',
    stroke='black', stroke_width=3))#ff6
shapes.add(dwg.rect(insert=(10 * self.bit7.flop.x * mm, 5
     * self.bit7.flop.y * mm), size=(self.bit7.flop.width
     * mm, self.bit7.flop.height * mm), fill='white',
    stroke='black', stroke_width=3))#ff7
shapes.add(dwg.rect(insert=(10 * self.bit8.flop.x * mm, 5
     * self.bit8.flop.y * mm), size=(self.bit8.flop.width
     * mm, self.bit8.flop.height * mm), fill='white',
    stroke='black', stroke_width=3))#ff8
shapes.add(dwg.rect(insert=(10 * self.bit9.flop.x * mm, 5
     * self.bit9.flop.y * mm), size=(self.bit9.flop.width
     * mm, self.bit9.flop.height * mm), fill='white',
    stroke='black', stroke_width=3))#ff9
shapes.add(dwg.rect(insert=(10 * self.bit10.flop.x * mm,
    5 * self.bit10.flop.y * mm), size=(self.bit10.flop.
    width * mm, self.bit10.flop.height * mm),fill='white'
    , stroke='black', stroke_width=3))#ff10
shapes.add(dwg.rect(insert=(10 * self.bit11.flop.x * mm,
    5 * self.bit11.flop.y * mm), size=(self.bit11.flop.
    width * mm, self.bit11.flop.height * mm), fill='white
    ', stroke='black', stroke_width=3))#ff11
shapes.add(dwg.rect(insert=(10 * self.bit12.flop.x * mm,
    5 * self.bit12.flop.y * mm), size=(self.bit12.flop.
    width * mm, self.bit12.flop.height * mm), fill='white
    ', stroke='black', stroke_width=3))#ff12
shapes.add(dwg.rect(insert=(10 * self.bit13.flop.x * mm,
    5 * self.bit13.flop.y * mm), size=(self.bit13.flop.
    width * mm, self.bit13.flop.height * mm), fill='white
    ', stroke='black', stroke_width=3))#ff1
shapes.add(dwg.rect(insert=(10 * self.bit14.flop.x * mm,
    5 * self.bit14.flop.y * mm), size=(self.bit14.flop.
    width * mm, self.bit14.flop.height * mm),fill='white'
    , stroke='black', stroke_width=3))#ff14
shapes.add(dwg.rect(insert=(10 * self.bit15.flop.x * mm,
    5 * self.bit15.flop.y * mm), size=(self.bit15.flop.
    width * mm, self.bit15.flop.height * mm), fill='white
    ', stroke='black', stroke_width=3))#ff15
shapes.add(dwg.rect(insert=(10 * self.bit16.flop.x * mm,
    5 * self.bit16.flop.y * mm), size=(self.bit16.flop.
    width * mm, self.bit16.flop.height * mm), fill='white
    ', stroke='black', stroke_width=3))#ff16
shapes.add(dwg.rect(insert=(10 * self.bit17.flop.x * mm,
    5 * self.bit17.flop.y * mm), size=(self.bit17.flop.
    width * mm, self.bit17.flop.height * mm), fill='white
    ', stroke='black', stroke_width=3))#ff17
shapes.add(dwg.rect(insert=(10 * self.bit18.flop.x * mm,
    5 * self.bit18.flop.y * mm), size=(self.bit18.flop.
    width * mm, self.bit18.flop.height * mm),fill='white'
    , stroke='black', stroke_width=3))#ff18
shapes.add(dwg.rect(insert=(10 * self.bit19.flop.x * mm,
    5 * self.bit19.flop.y * mm), size=(self.bit19.flop.
    width * mm, self.bit19.flop.height * mm), fill='white
    ', stroke='black', stroke_width=3))#ff19
shapes.add(dwg.rect(insert=(10 * self.bit20.flop.x * mm,
    5 * self.bit20.flop.y * mm), size=(self.bit20.flop.
    width * mm, self.bit20.flop.height * mm), fill='white
    ', stroke='black', stroke_width=3))#ff20
shapes.add(dwg.rect(insert=(10 * self.bit21.flop.x * mm,
    5 * self.bit21.flop.y * mm), size=(self.bit21.flop.
    width * mm, self.bit21.flop.height * mm), fill='white
    ', stroke='black', stroke_width=3))#ff21
shapes.add(dwg.rect(insert=(10 * self.bit22.flop.x * mm,
    5 * self.bit22.flop.y * mm), size=(self.bit22.flop.
    width * mm, self.bit22.flop.height * mm),fill='white'
    , stroke='black', stroke_width=3))#ff22
shapes.add(dwg.rect(insert=(10 * self.bit23.flop.x * mm,
    5 * self.bit23.flop.y * mm), size=(self.bit23.flop.
    width * mm, self.bit23.flop.height * mm), fill='white
    ', stroke='black', stroke_width=3))#ff23
shapes.add(dwg.rect(insert=(10 * self.bit24.flop.x * mm,
    5 * self.bit24.flop.y * mm), size=(self.bit24.flop.
    width * mm, self.bit24.flop.height * mm), fill='white
    ', stroke='black', stroke_width=3))#ff24
shapes.add(dwg.rect(insert=(10 * self.bit25.flop.x * mm,
    5 * self.bit25.flop.y * mm), size=(self.bit25.flop.
    width * mm, self.bit25.flop.height * mm), fill='white
    ', stroke='black', stroke_width=3))#ff25
shapes.add(dwg.rect(insert=(10 * self.bit26.flop.x * mm,
    5 * self.bit26.flop.y * mm), size=(self.bit26.flop.
    width * mm, self.bit26.flop.height * mm),fill='white'
    , stroke='black', stroke_width=3))#ff26
shapes.add(dwg.rect(insert=(10 * self.bit27.flop.x * mm,
    5 * self.bit27.flop.y * mm), size=(self.bit27.flop.
    width * mm, self.bit27.flop.height * mm), fill='white
    ', stroke='black', stroke_width=3))#ff27
shapes.add(dwg.rect(insert=(10 * self.bit28.flop.x * mm,
    5 * self.bit28.flop.y * mm), size=(self.bit28.flop.
    width * mm, self.bit28.flop.height * mm), fill='white
    ', stroke='black', stroke_width=3))#ff28
shapes.add(dwg.rect(insert=(10 * self.bit29.flop.x * mm,
    5 * self.bit29.flop.y * mm), size=(self.bit29.flop.
    width * mm, self.bit29.flop.height * mm), fill='white
    ', stroke='black', stroke_width=3))#ff29
shapes.add(dwg.rect(insert=(10 * self.bit30.flop.x * mm,
    5 * self.bit30.flop.y * mm), size=(self.bit30.flop.
    width * mm, self.bit30.flop.height * mm),fill='white'
    , stroke='black', stroke_width=3))#ff30
```

```
shapes.add(dwg.rect(insert=(10 * self.bit31.flop.x * mm,
     5 * self.bit31.flop.y * mm), size=(self.bit31.flop.
     width * mm, self.bit31.flop.height * mm), fill='white
     ', stroke='black', stroke_width=3))#ff31
g = dwg.g(
style="font-size:12;font-family:Comic␣Sans␣MS,␣Arial;font
     -weight:bold;font-style:␣oblique;stroke:␣black;stroke
     ␣-␣width:␣1;fill:␣black")
g.add(dwg.text(self.bit0.flop.name,insert=((10 * self.
     bit0.flop.x + 4) * mm, (5 * self.bit0.flop.y + 3.5) *
      mm))) # settings are valid for all text added to 'g
     '
g.add(dwg.text(self.bit1.flop.name,insert=((10 * self.
     bit1.flop.x + 4) * mm, (5 * self.bit1.flop.y + 3.5) *
      mm))) # settings are valid for all text added to 'g
     '
g.add(dwg.text(self.bit2.flop.name,insert=((10 * self.
     bit2.flop.x + 4) * mm, (5 * self.bit2.flop.y + 3.5) *
      mm))) # settings are valid for all text added to 'g
     '
g.add(dwg.text(self.bit3.flop.name,insert=((10 * self.
     bit3.flop.x + 4) * mm, (5 * self.bit3.flop.y + 3.5) *
      mm))) # settings are valid for all text added to 'g
     '
g.add(dwg.text(self.bit4.flop.name,insert=((10 * self.
     bit4.flop.x + 4) * mm, (5 * self.bit4.flop.y + 3.5) *
      mm))) # settings are valid for all text added to 'g
     '
g.add(dwg.text(self.bit5.flop.name,insert=((10 * self.
     bit5.flop.x + 4) * mm, (5 * self.bit5.flop.y + 3.5) *
      mm))) # settings are valid for all text added to 'g
     '
g.add(dwg.text(self.bit6.flop.name,insert=((10 * self.
     bit6.flop.x + 4) * mm, (5 * self.bit6.flop.y + 3.5) *
      mm))) # settings are valid for all text added to 'g
     '
g.add(dwg.text(self.bit7.flop.name,insert=((10 * self.
     bit7.flop.x + 4) * mm, (5 * self.bit7.flop.y + 3.5) *
      mm))) # settings are valid for all text added to 'g
     '
g.add(dwg.text(self.bit8.flop.name,insert=((10 * self.
     bit8.flop.x + 4) * mm, (5 * self.bit8.flop.y + 3.5) *
      mm))) # settings are valid for all text added to 'g
     '
g.add(dwg.text(self.bit9.flop.name,insert=((10 * self.
     bit9.flop.x + 4) * mm, (5 * self.bit9.flop.y + 3.5) *
      mm))) # settings are valid for all text added to 'g
     '
g.add(dwg.text(self.bit10.flop.name,insert=((10 * self.
     bit10.flop.x + 4) * mm, (5 * self.bit10.flop.y + 3.5)
      * mm))) # settings are valid for all text added to
     'g'
g.add(dwg.text(self.bit11.flop.name,insert=((10 * self.
     bit11.flop.x + 4) * mm, (5 * self.bit11.flop.y + 3.5)
      * mm))) # settings are valid for all text added to
     'g'
g.add(dwg.text(self.bit12.flop.name,insert=((10 * self.
     bit12.flop.x + 4) * mm, (5 * self.bit12.flop.y + 3.5)
      * mm))) # settings are valid for all text added to
     'g'
g.add(dwg.text(self.bit13.flop.name,insert=((10 * self.
        bit13.flop.x + 4) * mm, (5 * self.bit13.flop.y + 3.5)
         * mm))) # settings are valid for all text added to
        'g'
    g.add(dwg.text(self.bit14.flop.name,insert=((10 * self.
        bit14.flop.x + 4) * mm, (5 * self.bit14.flop.y + 3.5)
         * mm))) # settings are valid for all text added to
        'g'
    g.add(dwg.text(self.bit15.flop.name,insert=((10 * self.
        bit15.flop.x + 4) * mm, (5 * self.bit15.flop.y + 3.5)
         * mm))) # settings are valid for all text added to
        'g'
    g.add(dwg.text(self.bit16.flop.name,insert=((10 * self.
        bit16.flop.x + 4) * mm, (5 * self.bit16.flop.y + 3.5)
         * mm))) # settings are valid for all text added to
        'g'
    g.add(dwg.text(self.bit17.flop.name,insert=((10 * self.
        bit17.flop.x + 4) * mm, (5 * self.bit17.flop.y + 3.5)
         * mm))) # settings are valid for all text added to
        'g'
    g.add(dwg.text(self.bit18.flop.name,insert=((10 * self.
        bit18.flop.x + 4) * mm, (5 * self.bit18.flop.y + 3.5)
         * mm))) # settings are valid for all text added to
        'g'
    g.add(dwg.text(self.bit19.flop.name,insert=((10 * self.
        bit19.flop.x + 4) * mm, (5 * self.bit19.flop.y + 3.5)
         * mm))) # settings are valid for all text added to
        'g'
    g.add(dwg.text(self.bit20.flop.name,insert=((10 * self.
        bit20.flop.x + 4) * mm, (5 * self.bit20.flop.y + 3.5)
         * mm))) # settings are valid for all text added to
        'g'
    g.add(dwg.text(self.bit21.flop.name,insert=((10 * self.
        bit21.flop.x + 4) * mm, (5 * self.bit21.flop.y + 3.5)
         * mm))) # settings are valid for all text added to
        'g'
    g.add(dwg.text(self.bit22.flop.name,insert=((10 * self.
        bit22.flop.x + 4) * mm, (5 * self.bit22.flop.y + 3.5)
         * mm))) # settings are valid for all text added to
        'g'
    g.add(dwg.text(self.bit23.flop.name,insert=((10 * self.
        bit23.flop.x + 4) * mm, (5 * self.bit23.flop.y + 3.5)
         * mm))) # settings are valid for all text added to
        'g'
    g.add(dwg.text(self.bit24.flop.name,insert=((10 * self.
        bit24.flop.x + 4) * mm, (5 * self.bit24.flop.y + 3.5)
         * mm))) # settings are valid for all text added to
        'g'
    g.add(dwg.text(self.bit25.flop.name,insert=((10 * self.
        bit25.flop.x + 4) * mm, (5 * self.bit25.flop.y + 3.5)
         * mm))) # settings are valid for all text added to
        'g'
    g.add(dwg.text(self.bit26.flop.name,insert=((10 * self.
        bit26.flop.x + 4) * mm, (5 * self.bit26.flop.y + 3.5)
         * mm))) # settings are valid for all text added to
        'g'
    g.add(dwg.text(self.bit27.flop.name,insert=((10 * self.
        bit27.flop.x + 4) * mm, (5 * self.bit27.flop.y + 3.5)
         * mm))) # settings are valid for all text added to
        'g'
    g.add(dwg.text(self.bit28.flop.name,insert=((10 * self.
        bit28.flop.x + 4) * mm, (5 * self.bit28.flop.y + 3.5)
```

```
                 * mm))) # settings are valid for all text added to
                 'g'
            g.add(dwg.text(self.bit29.flop.name,insert=((10 * self.
                 bit29.flop.x + 4) * mm, (5 * self.bit29.flop.y + 3.5)
                 * mm))) # settings are valid for all text added to
                 'g'
            g.add(dwg.text(self.bit30.flop.name,insert=((10 * self.
                 bit30.flop.x + 4) * mm, (5 * self.bit30.flop.y + 3.5)
                 * mm))) # settings are valid for all text added to
                 'g'
            g.add(dwg.text(self.bit31.flop.name,insert=((10 * self.
                 bit31.flop.x + 4) * mm, (5 * self.bit31.flop.y + 3.5)
                 * mm))) # settings are valid for all text added to
                 'g'

            dwg.add(g)
            dwg.save()
    def writeRpGroup(self,file,rp_group):
            #f = open(file,'w')
            f.write(rp_group+self.bit0.flop.name+"␣-column␣"+str(self
                 .bit0.flop.x)+"␣-row␣"+str(self.bit0.flop.y)+"␣\n")
            f.write(rp_group+self.bit1.flop.name+"␣-column␣"+str(self
                 .bit1.flop.x)+"␣-row␣"+str(self.bit1.flop.y)+"␣\n")
            f.write(rp_group+self.bit2.flop.name+"␣-column␣"+str(self
                 .bit2.flop.x)+"␣-row␣"+str(self.bit2.flop.y)+"␣\n")
            f.write(rp_group+self.bit3.flop.name+"␣-column␣"+str(self
                 .bit3.flop.x)+"␣-row␣"+str(self.bit3.flop.y)+"␣\n")
            f.write(rp_group+self.bit4.flop.name+"␣-column␣"+str(self
                 .bit4.flop.x)+"␣-row␣"+str(self.bit4.flop.y)+"␣\n")
            f.write(rp_group+self.bit5.flop.name+"␣-column␣"+str(self
                 .bit5.flop.x)+"␣-row␣"+str(self.bit5.flop.y)+"␣\n")
            f.write(rp_group+self.bit6.flop.name+"␣-column␣"+str(self
                 .bit6.flop.x)+"␣-row␣"+str(self.bit6.flop.y)+"␣\n")
            f.write(rp_group+self.bit7.flop.name+"␣-column␣"+str(self
                 .bit7.flop.x)+"␣-row␣"+str(self.bit7.flop.y)+"␣\n")
            f.write(rp_group+self.bit8.flop.name+"␣-column␣"+str(self
                 .bit8.flop.x)+"␣-row␣"+str(self.bit8.flop.y)+"␣\n")
            f.write(rp_group+self.bit9.flop.name+"␣-column␣"+str(self
                 .bit9.flop.x)+"␣-row␣"+str(self.bit9.flop.y)+"␣\n")
            f.write(rp_group+self.bit10.flop.name+"␣-column␣"+str(
                 self.bit10.flop.x)+"␣-row␣"+str(self.bit10.flop.y)+"␣
                 \n")
            f.write(rp_group+self.bit11.flop.name+"␣-column␣"+str(
                 self.bit11.flop.x)+"␣-row␣"+str(self.bit11.flop.y)+"␣
                 \n")
            f.write(rp_group+self.bit12.flop.name+"␣-column␣"+str(
                 self.bit12.flop.x)+"␣-row␣"+str(self.bit12.flop.y)+"␣
                 \n")
            f.write(rp_group+self.bit13.flop.name+"␣-column␣"+str(
                 self.bit13.flop.x)+"␣-row␣"+str(self.bit13.flop.y)+"␣
                 \n")
            f.write(rp_group+self.bit14.flop.name+"␣-column␣"+str(
                 self.bit14.flop.x)+"␣-row␣"+str(self.bit14.flop.y)+"␣
                 \n")
            f.write(rp_group+self.bit15.flop.name+"␣-column␣"+str(
                 self.bit15.flop.x)+"␣-row␣"+str(self.bit15.flop.y)+"␣
                 \n")
            f.write(rp_group+self.bit16.flop.name+"␣-column␣"+str(
                 self.bit16.flop.x)+"␣-row␣"+str(self.bit16.flop.y)+"␣
                 \n")
            f.write(rp_group+self.bit17.flop.name+"␣-column␣"+str(
                        self.bit17.flop.x)+"␣-row␣"+str(self.bit17.flop.y)+"␣
                        \n")
                f.write(rp_group+self.bit18.flop.name+"␣-column␣"+str(
                        self.bit18.flop.x)+"␣-row␣"+str(self.bit18.flop.y)+"␣
                        \n")
                f.write(rp_group+self.bit19.flop.name+"␣-column␣"+str(
                        self.bit19.flop.x)+"␣-row␣"+str(self.bit19.flop.y)+"␣
                        \n")
                f.write(rp_group+self.bit20.flop.name+"␣-column␣"+str(
                        self.bit20.flop.x)+"␣-row␣"+str(self.bit20.flop.y)+"␣
                        \n")
                f.write(rp_group+self.bit21.flop.name+"␣-column␣"+str(
                        self.bit21.flop.x)+"␣-row␣"+str(self.bit21.flop.y)+"␣
                        \n")
                f.write(rp_group+self.bit22.flop.name+"␣-column␣"+str(
                        self.bit22.flop.x)+"␣-row␣"+str(self.bit22.flop.y)+"␣
                        \n")
                f.write(rp_group+self.bit23.flop.name+"␣-column␣"+str(
                        self.bit23.flop.x)+"␣-row␣"+str(self.bit23.flop.y)+"␣
                        \n")
                f.write(rp_group+self.bit24.flop.name+"␣-column␣"+str(
                        self.bit24.flop.x)+"␣-row␣"+str(self.bit24.flop.y)+"␣
                        \n")
                f.write(rp_group+self.bit25.flop.name+"␣-column␣"+str(
                        self.bit25.flop.x)+"␣-row␣"+str(self.bit25.flop.y)+"␣
                        \n")
                f.write(rp_group+self.bit26.flop.name+"␣-column␣"+str(
                        self.bit26.flop.x)+"␣-row␣"+str(self.bit26.flop.y)+"␣
                        \n")
                f.write(rp_group+self.bit27.flop.name+"␣-column␣"+str(
                        self.bit27.flop.x)+"␣-row␣"+str(self.bit27.flop.y)+"␣
                        \n")
                f.write(rp_group+self.bit28.flop.name+"␣-column␣"+str(
                        self.bit28.flop.x)+"␣-row␣"+str(self.bit28.flop.y)+"␣
                        \n")
                f.write(rp_group+self.bit29.flop.name+"␣-column␣"+str(
                        self.bit29.flop.x)+"␣-row␣"+str(self.bit29.flop.y)+"␣
                        \n")
                f.write(rp_group+self.bit30.flop.name+"␣-column␣"+str(
                        self.bit30.flop.x)+"␣-row␣"+str(self.bit30.flop.y)+"␣
                        \n")
                f.write(rp_group+self.bit31.flop.name+"␣-column␣"+str(
                        self.bit31.flop.x)+"␣-row␣"+str(self.bit31.flop.y)+"␣
                        \n")
f = open('test.tcl','w')
f.write("create_rp_group␣rp1␣-design␣pil␣-columns␣32␣-rows␣32␣\n")
rp_group_str = "add_to_rp_group␣pil::rp1␣-leaf␣U_CPU/U_REG_FILE/"
temp = Register(0,0)
temp.printBits()
for i in range(0,32):
        temp = Register(i,i)
        temp.writeRpGroup(f,rp_group_str)
        f.write("\n")

f.close()
```

Listing 6: Relative placement directives

```
create_rp_group rp1 -design pil -columns 32 -rows 32
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___0_ -column
      0 -row 0
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___1_ -column
      0 -row 1
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___2_ -column
      0 -row 2
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___3_ -column
      0 -row 3
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___4_ -column
      0 -row 4
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___5_ -column
      0 -row 5
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___6_ -column
      0 -row 6
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___7_ -column
      0 -row 7
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___8_ -column
      0 -row 8
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___9_ -column
      0 -row 9
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___10_
     -column 0 -row 10
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___11_
     -column 0 -row 11
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___12_
     -column 0 -row 12
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___13_
     -column 0 -row 13
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___14_
     -column 0 -row 14
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___15_
     -column 0 -row 15
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___16_
     -column 0 -row 16
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___17_
     -column 0 -row 17
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___18_
     -column 0 -row 18
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___19_
     -column 0 -row 19
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___20_
     -column 0 -row 20
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___21_
     -column 0 -row 21
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___22_
     -column 0 -row 22
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___23_
     -column 0 -row 23
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___24_
     -column 0 -row 24
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___25_
     -column 0 -row 25
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___26_
     -column 0 -row 26
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___27_
     -column 0 -row 27
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___28_
     -column 0 -row 28
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_0___29_
```

```
    −column 0 −row 29
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_0___30_
    −column 0 −row 30
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_0___31_
    −column 0 −row 31

add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___0_ −column
     1 −row 0
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___1_ −column
     1 −row 1
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___2_ −column
     1 −row 2
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___3_ −column
     1 −row 3
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___4_ −column
     1 −row 4
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___5_ −column
     1 −row 5
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___6_ −column
     1 −row 6
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___7_ −column
     1 −row 7
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___8_ −column
     1 −row 8
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___9_ −column
     1 −row 9
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___10_
    −column 1 −row 10
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___11_
    −column 1 −row 11
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___12_
    −column 1 −row 12
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___13_
    −column 1 −row 13
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___14_
    −column 1 −row 14
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___15_
    −column 1 −row 15
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___16_
    −column 1 −row 16
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___17_
    −column 1 −row 17
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___18_
    −column 1 −row 18
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___19_
    −column 1 −row 19
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___20_
    −column 1 −row 20
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___21_
    −column 1 −row 21
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___22_
    −column 1 −row 22
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___23_
    −column 1 −row 23
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___24_
    −column 1 −row 24
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___25_
    −column 1 −row 25
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___26_
    −column 1 −row 26
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_1___27_
```

```
    -column 1 -row 27
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_1___28_
    -column 1 -row 28
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_1___29_
    -column 1 -row 29
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_1___30_
    -column 1 -row 30
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_1___31_
    -column 1 -row 31

add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_2___0_ -column
     2 -row 0
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_2___1_ -column
     2 -row 1
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_2___2_ -column
     2 -row 2
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_2___3_ -column
     2 -row 3
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_2___4_ -column
     2 -row 4
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_2___5_ -column
     2 -row 5
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_2___6_ -column
     2 -row 6
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_2___7_ -column
     2 -row 7
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_2___8_ -column
     2 -row 8
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_2___9_ -column
     2 -row 9
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_2___10_
    -column 2 -row 10
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_2___11_
    -column 2 -row 11
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_2___12_
    -column 2 -row 12
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_2___13_
    -column 2 -row 13
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_2___14_
    -column 2 -row 14
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_2___15_
    -column 2 -row 15
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_2___16_
    -column 2 -row 16
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_2___17_
    -column 2 -row 17
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_2___18_
    -column 2 -row 18
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_2___19_
    -column 2 -row 19
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_2___20_
    -column 2 -row 20
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_2___21_
    -column 2 -row 21
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_2___22_
    -column 2 -row 22
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_2___23_
    -column 2 -row 23
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_2___24_
    -column 2 -row 24
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_2___25_
```

```
    −column 2 −row 25
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_2___26_
    −column 2 −row 26
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_2___27_
    −column 2 −row 27
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_2___28_
    −column 2 −row 28
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_2___29_
    −column 2 −row 29
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_2___30_
    −column 2 −row 30
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_2___31_
    −column 2 −row 31

add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___0_ −column
     3 −row 0
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___1_ −column
     3 −row 1
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___2_ −column
     3 −row 2
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___3_ −column
     3 −row 3
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___4_ −column
     3 −row 4
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___5_ −column
     3 −row 5
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___6_ −column
     3 −row 6
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___7_ −column
     3 −row 7
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___8_ −column
     3 −row 8
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___9_ −column
     3 −row 9
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___10_
    −column 3 −row 10
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___11_
    −column 3 −row 11
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___12_
    −column 3 −row 12
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___13_
    −column 3 −row 13
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___14_
    −column 3 −row 14
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___15_
    −column 3 −row 15
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___16_
    −column 3 −row 16
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___17_
    −column 3 −row 17
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___18_
    −column 3 −row 18
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___19_
    −column 3 −row 19
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___20_
    −column 3 −row 20
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___21_
    −column 3 −row 21
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___22_
    −column 3 −row 22
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___23_
```

```
    −column 3 −row 23
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___24_
    −column 3 −row 24
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___25_
    −column 3 −row 25
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___26_
    −column 3 −row 26
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___27_
    −column 3 −row 27
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___28_
    −column 3 −row 28
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___29_
    −column 3 −row 29
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___30_
    −column 3 −row 30
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_3___31_
    −column 3 −row 31

add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___0_ −column
     4 −row 0
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___1_ −column
     4 −row 1
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___2_ −column
     4 −row 2
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___3_ −column
     4 −row 3
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___4_ −column
     4 −row 4
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___5_ −column
     4 −row 5
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___6_ −column
     4 −row 6
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___7_ −column
     4 −row 7
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___8_ −column
     4 −row 8
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___9_ −column
     4 −row 9
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___10_
    −column 4 −row 10
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___11_
    −column 4 −row 11
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___12_
    −column 4 −row 12
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___13_
    −column 4 −row 13
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___14_
    −column 4 −row 14
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___15_
    −column 4 −row 15
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___16_
    −column 4 −row 16
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___17_
    −column 4 −row 17
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___18_
    −column 4 −row 18
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___19_
    −column 4 −row 19
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___20_
    −column 4 −row 20
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___21_
```

```
    −column 4 −row 21
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___22_
    −column 4 −row 22
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___23_
    −column 4 −row 23
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___24_
    −column 4 −row 24
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___25_
    −column 4 −row 25
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___26_
    −column 4 −row 26
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___27_
    −column 4 −row 27
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___28_
    −column 4 −row 28
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___29_
    −column 4 −row 29
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___30_
    −column 4 −row 30
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_4___31_
    −column 4 −row 31

add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_5___0_ −column
     5 −row 0
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_5___1_ −column
     5 −row 1
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_5___2_ −column
     5 −row 2
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_5___3_ −column
     5 −row 3
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_5___4_ −column
     5 −row 4
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_5___5_ −column
     5 −row 5
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_5___6_ −column
     5 −row 6
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_5___7_ −column
     5 −row 7
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_5___8_ −column
     5 −row 8
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_5___9_ −column
     5 −row 9
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_5___10_
    −column 5 −row 10
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_5___11_
    −column 5 −row 11
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_5___12_
    −column 5 −row 12
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_5___13_
    −column 5 −row 13
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_5___14_
    −column 5 −row 14
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_5___15_
    −column 5 −row 15
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_5___16_
    −column 5 −row 16
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_5___17_
    −column 5 −row 17
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_5___18_
    −column 5 −row 18
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_5___19_
```

```
     -column 5 -row 19
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_5___20_
     -column 5 -row 20
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_5___21_
     -column 5 -row 21
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_5___22_
     -column 5 -row 22
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_5___23_
     -column 5 -row 23
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_5___24_
     -column 5 -row 24
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_5___25_
     -column 5 -row 25
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_5___26_
     -column 5 -row 26
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_5___27_
     -column 5 -row 27
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_5___28_
     -column 5 -row 28
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_5___29_
     -column 5 -row 29
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_5___30_
     -column 5 -row 30
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_5___31_
     -column 5 -row 31

add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_6___0_ -column
      6 -row 0
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_6___1_ -column
      6 -row 1
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_6___2_ -column
      6 -row 2
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_6___3_ -column
      6 -row 3
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_6___4_ -column
      6 -row 4
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_6___5_ -column
      6 -row 5
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_6___6_ -column
      6 -row 6
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_6___7_ -column
      6 -row 7
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_6___8_ -column
      6 -row 8
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_6___9_ -column
      6 -row 9
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_6___10_
     -column 6 -row 10
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_6___11_
     -column 6 -row 11
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_6___12_
     -column 6 -row 12
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_6___13_
     -column 6 -row 13
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_6___14_
     -column 6 -row 14
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_6___15_
     -column 6 -row 15
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_6___16_
     -column 6 -row 16
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_6___17_
```

```
     −column 6 −row 17
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_6___18_
     −column 6 −row 18
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_6___19_
     −column 6 −row 19
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_6___20_
     −column 6 −row 20
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_6___21_
     −column 6 −row 21
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_6___22_
     −column 6 −row 22
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_6___23_
     −column 6 −row 23
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_6___24_
     −column 6 −row 24
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_6___25_
     −column 6 −row 25
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_6___26_
     −column 6 −row 26
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_6___27_
     −column 6 −row 27
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_6___28_
     −column 6 −row 28
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_6___29_
     −column 6 −row 29
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_6___30_
     −column 6 −row 30
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_6___31_
     −column 6 −row 31

add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___0_ −column
      7 −row 0
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___1_ −column
      7 −row 1
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___2_ −column
      7 −row 2
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___3_ −column
      7 −row 3
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___4_ −column
      7 −row 4
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___5_ −column
      7 −row 5
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___6_ −column
      7 −row 6
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___7_ −column
      7 −row 7
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___8_ −column
      7 −row 8
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___9_ −column
      7 −row 9
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___10_
     −column 7 −row 10
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___11_
     −column 7 −row 11
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___12_
     −column 7 −row 12
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___13_
     −column 7 −row 13
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___14_
     −column 7 −row 14
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___15_
```

```
     −column 7 −row 15
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___16_
     −column 7 −row 16
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___17_
     −column 7 −row 17
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___18_
     −column 7 −row 18
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___19_
     −column 7 −row 19
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___20_
     −column 7 −row 20
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___21_
     −column 7 −row 21
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___22_
     −column 7 −row 22
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___23_
     −column 7 −row 23
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___24_
     −column 7 −row 24
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___25_
     −column 7 −row 25
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___26_
     −column 7 −row 26
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___27_
     −column 7 −row 27
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___28_
     −column 7 −row 28
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___29_
     −column 7 −row 29
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___30_
     −column 7 −row 30
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_7___31_
     −column 7 −row 31

add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_8___0_ −column
      8 −row 0
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_8___1_ −column
      8 −row 1
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_8___2_ −column
      8 −row 2
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_8___3_ −column
      8 −row 3
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_8___4_ −column
      8 −row 4
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_8___5_ −column
      8 −row 5
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_8___6_ −column
      8 −row 6
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_8___7_ −column
      8 −row 7
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_8___8_ −column
      8 −row 8
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_8___9_ −column
      8 −row 9
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_8___10_
     −column 8 −row 10
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_8___11_
     −column 8 −row 11
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_8___12_
     −column 8 −row 12
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_8___13_
```

```
    -column 8 -row 13
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_8___14_
    -column 8 -row 14
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_8___15_
    -column 8 -row 15
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_8___16_
    -column 8 -row 16
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_8___17_
    -column 8 -row 17
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_8___18_
    -column 8 -row 18
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_8___19_
    -column 8 -row 19
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_8___20_
    -column 8 -row 20
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_8___21_
    -column 8 -row 21
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_8___22_
    -column 8 -row 22
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_8___23_
    -column 8 -row 23
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_8___24_
    -column 8 -row 24
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_8___25_
    -column 8 -row 25
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_8___26_
    -column 8 -row 26
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_8___27_
    -column 8 -row 27
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_8___28_
    -column 8 -row 28
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_8___29_
    -column 8 -row 29
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_8___30_
    -column 8 -row 30
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_8___31_
    -column 8 -row 31

add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___0_ -column
     9 -row 0
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___1_ -column
     9 -row 1
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___2_ -column
     9 -row 2
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___3_ -column
     9 -row 3
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___4_ -column
     9 -row 4
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___5_ -column
     9 -row 5
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___6_ -column
     9 -row 6
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___7_ -column
     9 -row 7
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___8_ -column
     9 -row 8
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___9_ -column
     9 -row 9
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___10_
    -column 9 -row 10
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___11_
```

```
    -column 9 -row 11
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___12_
    -column 9 -row 12
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___13_
    -column 9 -row 13
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___14_
    -column 9 -row 14
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___15_
    -column 9 -row 15
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___16_
    -column 9 -row 16
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___17_
    -column 9 -row 17
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___18_
    -column 9 -row 18
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___19_
    -column 9 -row 19
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___20_
    -column 9 -row 20
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___21_
    -column 9 -row 21
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___22_
    -column 9 -row 22
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___23_
    -column 9 -row 23
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___24_
    -column 9 -row 24
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___25_
    -column 9 -row 25
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___26_
    -column 9 -row 26
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___27_
    -column 9 -row 27
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___28_
    -column 9 -row 28
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___29_
    -column 9 -row 29
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___30_
    -column 9 -row 30
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_9___31_
    -column 9 -row 31

add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___0_
    -column 10 -row 0
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___1_
    -column 10 -row 1
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___2_
    -column 10 -row 2
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___3_
    -column 10 -row 3
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___4_
    -column 10 -row 4
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___5_
    -column 10 -row 5
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___6_
    -column 10 -row 6
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___7_
    -column 10 -row 7
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___8_
    -column 10 -row 8
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___9_
```

```
      -column 10 -row 9
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___10_
      -column 10 -row 10
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___11_
      -column 10 -row 11
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___12_
      -column 10 -row 12
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___13_
      -column 10 -row 13
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___14_
      -column 10 -row 14
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___15_
      -column 10 -row 15
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___16_
      -column 10 -row 16
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___17_
      -column 10 -row 17
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___18_
      -column 10 -row 18
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___19_
      -column 10 -row 19
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___20_
      -column 10 -row 20
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___21_
      -column 10 -row 21
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___22_
      -column 10 -row 22
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___23_
      -column 10 -row 23
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___24_
      -column 10 -row 24
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___25_
      -column 10 -row 25
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___26_
      -column 10 -row 26
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___27_
      -column 10 -row 27
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___28_
      -column 10 -row 28
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___29_
      -column 10 -row 29
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___30_
      -column 10 -row 30
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_10___31_
      -column 10 -row 31

add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___0_
      -column 11 -row 0
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___1_
      -column 11 -row 1
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___2_
      -column 11 -row 2
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___3_
      -column 11 -row 3
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___4_
      -column 11 -row 4
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___5_
      -column 11 -row 5
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___6_
      -column 11 -row 6
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___7_
```

```
    -column 11 -row 7
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___8_
    -column 11 -row 8
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___9_
    -column 11 -row 9
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___10_
    -column 11 -row 10
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___11_
    -column 11 -row 11
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___12_
    -column 11 -row 12
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___13_
    -column 11 -row 13
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___14_
    -column 11 -row 14
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___15_
    -column 11 -row 15
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___16_
    -column 11 -row 16
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___17_
    -column 11 -row 17
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___18_
    -column 11 -row 18
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___19_
    -column 11 -row 19
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___20_
    -column 11 -row 20
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___21_
    -column 11 -row 21
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___22_
    -column 11 -row 22
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___23_
    -column 11 -row 23
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___24_
    -column 11 -row 24
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___25_
    -column 11 -row 25
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___26_
    -column 11 -row 26
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___27_
    -column 11 -row 27
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___28_
    -column 11 -row 28
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___29_
    -column 11 -row 29
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___30_
    -column 11 -row 30
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_11___31_
    -column 11 -row 31

add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___0_
    -column 12 -row 0
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___1_
    -column 12 -row 1
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___2_
    -column 12 -row 2
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___3_
    -column 12 -row 3
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___4_
    -column 12 -row 4
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___5_
```

```
    -column 12 -row 5
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___6_
    -column 12 -row 6
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___7_
    -column 12 -row 7
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___8_
    -column 12 -row 8
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___9_
    -column 12 -row 9
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___10_
    -column 12 -row 10
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___11_
    -column 12 -row 11
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___12_
    -column 12 -row 12
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___13_
    -column 12 -row 13
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___14_
    -column 12 -row 14
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___15_
    -column 12 -row 15
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___16_
    -column 12 -row 16
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___17_
    -column 12 -row 17
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___18_
    -column 12 -row 18
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___19_
    -column 12 -row 19
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___20_
    -column 12 -row 20
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___21_
    -column 12 -row 21
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___22_
    -column 12 -row 22
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___23_
    -column 12 -row 23
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___24_
    -column 12 -row 24
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___25_
    -column 12 -row 25
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___26_
    -column 12 -row 26
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___27_
    -column 12 -row 27
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___28_
    -column 12 -row 28
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___29_
    -column 12 -row 29
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___30_
    -column 12 -row 30
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_12___31_
    -column 12 -row 31

add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___0_
    -column 13 -row 0
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___1_
    -column 13 -row 1
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___2_
    -column 13 -row 2
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___3_
```

```
    -column 13 -row 3
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___4_
    -column 13 -row 4
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___5_
    -column 13 -row 5
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___6_
    -column 13 -row 6
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___7_
    -column 13 -row 7
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___8_
    -column 13 -row 8
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___9_
    -column 13 -row 9
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___10_
    -column 13 -row 10
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___11_
    -column 13 -row 11
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___12_
    -column 13 -row 12
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___13_
    -column 13 -row 13
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___14_
    -column 13 -row 14
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___15_
    -column 13 -row 15
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___16_
    -column 13 -row 16
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___17_
    -column 13 -row 17
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___18_
    -column 13 -row 18
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___19_
    -column 13 -row 19
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___20_
    -column 13 -row 20
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___21_
    -column 13 -row 21
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___22_
    -column 13 -row 22
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___23_
    -column 13 -row 23
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___24_
    -column 13 -row 24
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___25_
    -column 13 -row 25
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___26_
    -column 13 -row 26
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___27_
    -column 13 -row 27
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___28_
    -column 13 -row 28
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___29_
    -column 13 -row 29
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___30_
    -column 13 -row 30
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_13___31_
    -column 13 -row 31

add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14___0_
    -column 14 -row 0
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14___1_
```

```
    -column 14 -row 1
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__2_
    -column 14 -row 2
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__3_
    -column 14 -row 3
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__4_
    -column 14 -row 4
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__5_
    -column 14 -row 5
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__6_
    -column 14 -row 6
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__7_
    -column 14 -row 7
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__8_
    -column 14 -row 8
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__9_
    -column 14 -row 9
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__10_
    -column 14 -row 10
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__11_
    -column 14 -row 11
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__12_
    -column 14 -row 12
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__13_
    -column 14 -row 13
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__14_
    -column 14 -row 14
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__15_
    -column 14 -row 15
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__16_
    -column 14 -row 16
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__17_
    -column 14 -row 17
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__18_
    -column 14 -row 18
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__19_
    -column 14 -row 19
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__20_
    -column 14 -row 20
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__21_
    -column 14 -row 21
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__22_
    -column 14 -row 22
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__23_
    -column 14 -row 23
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__24_
    -column 14 -row 24
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__25_
    -column 14 -row 25
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__26_
    -column 14 -row 26
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__27_
    -column 14 -row 27
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__28_
    -column 14 -row 28
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__29_
    -column 14 -row 29
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__30_
    -column 14 -row 30
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_14__31_
    -column 14 -row 31
```

```
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___0_
    -column 15 -row 0
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___1_
    -column 15 -row 1
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___2_
    -column 15 -row 2
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___3_
    -column 15 -row 3
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___4_
    -column 15 -row 4
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___5_
    -column 15 -row 5
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___6_
    -column 15 -row 6
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___7_
    -column 15 -row 7
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___8_
    -column 15 -row 8
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___9_
    -column 15 -row 9
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___10_
    -column 15 -row 10
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___11_
    -column 15 -row 11
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___12_
    -column 15 -row 12
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___13_
    -column 15 -row 13
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___14_
    -column 15 -row 14
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___15_
    -column 15 -row 15
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___16_
    -column 15 -row 16
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___17_
    -column 15 -row 17
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___18_
    -column 15 -row 18
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___19_
    -column 15 -row 19
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___20_
    -column 15 -row 20
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___21_
    -column 15 -row 21
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___22_
    -column 15 -row 22
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___23_
    -column 15 -row 23
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___24_
    -column 15 -row 24
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___25_
    -column 15 -row 25
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___26_
    -column 15 -row 26
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___27_
    -column 15 -row 27
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___28_
    -column 15 -row 28
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_15___29_
    -column 15 -row 29
```

```
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_15__30_
     −column 15 −row 30
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_15__31_
     −column 15 −row 31

add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__0_
     −column 16 −row 0
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__1_
     −column 16 −row 1
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__2_
     −column 16 −row 2
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__3_
     −column 16 −row 3
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__4_
     −column 16 −row 4
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__5_
     −column 16 −row 5
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__6_
     −column 16 −row 6
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__7_
     −column 16 −row 7
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__8_
     −column 16 −row 8
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__9_
     −column 16 −row 9
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__10_
     −column 16 −row 10
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__11_
     −column 16 −row 11
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__12_
     −column 16 −row 12
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__13_
     −column 16 −row 13
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__14_
     −column 16 −row 14
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__15_
     −column 16 −row 15
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__16_
     −column 16 −row 16
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__17_
     −column 16 −row 17
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__18_
     −column 16 −row 18
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__19_
     −column 16 −row 19
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__20_
     −column 16 −row 20
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__21_
     −column 16 −row 21
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__22_
     −column 16 −row 22
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__23_
     −column 16 −row 23
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__24_
     −column 16 −row 24
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__25_
     −column 16 −row 25
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__26_
     −column 16 −row 26
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_16__27_
     −column 16 −row 27
```

```
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_16__28_
    -column 16 -row 28
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_16__29_
    -column 16 -row 29
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_16__30_
    -column 16 -row 30
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_16__31_
    -column 16 -row 31

add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__0_
    -column 17 -row 0
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__1_
    -column 17 -row 1
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__2_
    -column 17 -row 2
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__3_
    -column 17 -row 3
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__4_
    -column 17 -row 4
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__5_
    -column 17 -row 5
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__6_
    -column 17 -row 6
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__7_
    -column 17 -row 7
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__8_
    -column 17 -row 8
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__9_
    -column 17 -row 9
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__10_
    -column 17 -row 10
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__11_
    -column 17 -row 11
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__12_
    -column 17 -row 12
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__13_
    -column 17 -row 13
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__14_
    -column 17 -row 14
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__15_
    -column 17 -row 15
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__16_
    -column 17 -row 16
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__17_
    -column 17 -row 17
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__18_
    -column 17 -row 18
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__19_
    -column 17 -row 19
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__20_
    -column 17 -row 20
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__21_
    -column 17 -row 21
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__22_
    -column 17 -row 22
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__23_
    -column 17 -row 23
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__24_
    -column 17 -row 24
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__25_
    -column 17 -row 25
```

```
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__26_
    -column 17 -row 26
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__27_
    -column 17 -row 27
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__28_
    -column 17 -row 28
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__29_
    -column 17 -row 29
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__30_
    -column 17 -row 30
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_17__31_
    -column 17 -row 31

add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18__0_
    -column 18 -row 0
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18__1_
    -column 18 -row 1
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18__2_
    -column 18 -row 2
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18__3_
    -column 18 -row 3
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18__4_
    -column 18 -row 4
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18__5_
    -column 18 -row 5
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18__6_
    -column 18 -row 6
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18__7_
    -column 18 -row 7
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18__8_
    -column 18 -row 8
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18__9_
    -column 18 -row 9
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18__10_
    -column 18 -row 10
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18__11_
    -column 18 -row 11
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18__12_
    -column 18 -row 12
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18__13_
    -column 18 -row 13
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18__14_
    -column 18 -row 14
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18__15_
    -column 18 -row 15
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18__16_
    -column 18 -row 16
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18__17_
    -column 18 -row 17
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18__18_
    -column 18 -row 18
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18__19_
    -column 18 -row 19
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18__20_
    -column 18 -row 20
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18__21_
    -column 18 -row 21
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18__22_
    -column 18 -row 22
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18__23_
    -column 18 -row 23
```

```
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18___24_
     -column 18 -row 24
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18___25_
     -column 18 -row 25
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18___26_
     -column 18 -row 26
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18___27_
     -column 18 -row 27
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18___28_
     -column 18 -row 28
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18___29_
     -column 18 -row 29
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18___30_
     -column 18 -row 30
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_18___31_
     -column 18 -row 31

add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_19___0_
     -column 19 -row 0
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_19___1_
     -column 19 -row 1
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_19___2_
     -column 19 -row 2
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_19___3_
     -column 19 -row 3
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_19___4_
     -column 19 -row 4
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_19___5_
     -column 19 -row 5
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_19___6_
     -column 19 -row 6
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_19___7_
     -column 19 -row 7
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_19___8_
     -column 19 -row 8
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_19___9_
     -column 19 -row 9
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_19___10_
     -column 19 -row 10
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_19___11_
     -column 19 -row 11
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_19___12_
     -column 19 -row 12
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_19___13_
     -column 19 -row 13
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_19___14_
     -column 19 -row 14
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_19___15_
     -column 19 -row 15
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_19___16_
     -column 19 -row 16
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_19___17_
     -column 19 -row 17
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_19___18_
     -column 19 -row 18
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_19___19_
     -column 19 -row 19
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_19___20_
     -column 19 -row 20
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_19___21_
     -column 19 -row 21
```

```
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_19__22_
    −column 19 −row 22
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_19__23_
    −column 19 −row 23
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_19__24_
    −column 19 −row 24
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_19__25_
    −column 19 −row 25
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_19__26_
    −column 19 −row 26
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_19__27_
    −column 19 −row 27
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_19__28_
    −column 19 −row 28
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_19__29_
    −column 19 −row 29
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_19__30_
    −column 19 −row 30
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_19__31_
    −column 19 −row 31

add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_20__0_
    −column 20 −row 0
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_20__1_
    −column 20 −row 1
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_20__2_
    −column 20 −row 2
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_20__3_
    −column 20 −row 3
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_20__4_
    −column 20 −row 4
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_20__5_
    −column 20 −row 5
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_20__6_
    −column 20 −row 6
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_20__7_
    −column 20 −row 7
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_20__8_
    −column 20 −row 8
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_20__9_
    −column 20 −row 9
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_20__10_
    −column 20 −row 10
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_20__11_
    −column 20 −row 11
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_20__12_
    −column 20 −row 12
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_20__13_
    −column 20 −row 13
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_20__14_
    −column 20 −row 14
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_20__15_
    −column 20 −row 15
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_20__16_
    −column 20 −row 16
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_20__17_
    −column 20 −row 17
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_20__18_
    −column 20 −row 18
add_to_rp_group pil::rp1 −leaf U_CPU/U_REG_FILE/regfile_reg_20__19_
    −column 20 −row 19
```

```
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_20___20_
     -column 20 -row 20
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_20___21_
     -column 20 -row 21
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_20___22_
     -column 20 -row 22
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_20___23_
     -column 20 -row 23
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_20___24_
     -column 20 -row 24
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_20___25_
     -column 20 -row 25
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_20___26_
     -column 20 -row 26
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_20___27_
     -column 20 -row 27
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_20___28_
     -column 20 -row 28
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_20___29_
     -column 20 -row 29
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_20___30_
     -column 20 -row 30
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_20___31_
     -column 20 -row 31

add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21___0_
     -column 21 -row 0
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21___1_
     -column 21 -row 1
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21___2_
     -column 21 -row 2
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21___3_
     -column 21 -row 3
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21___4_
     -column 21 -row 4
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21___5_
     -column 21 -row 5
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21___6_
     -column 21 -row 6
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21___7_
     -column 21 -row 7
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21___8_
     -column 21 -row 8
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21___9_
     -column 21 -row 9
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21___10_
     -column 21 -row 10
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21___11_
     -column 21 -row 11
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21___12_
     -column 21 -row 12
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21___13_
     -column 21 -row 13
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21___14_
     -column 21 -row 14
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21___15_
     -column 21 -row 15
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21___16_
     -column 21 -row 16
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21___17_
     -column 21 -row 17
```

```
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21__18_
    -column 21 -row 18
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21__19_
    -column 21 -row 19
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21__20_
    -column 21 -row 20
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21__21_
    -column 21 -row 21
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21__22_
    -column 21 -row 22
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21__23_
    -column 21 -row 23
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21__24_
    -column 21 -row 24
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21__25_
    -column 21 -row 25
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21__26_
    -column 21 -row 26
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21__27_
    -column 21 -row 27
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21__28_
    -column 21 -row 28
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21__29_
    -column 21 -row 29
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21__30_
    -column 21 -row 30
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_21__31_
    -column 21 -row 31

add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22__0_
    -column 22 -row 0
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22__1_
    -column 22 -row 1
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22__2_
    -column 22 -row 2
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22__3_
    -column 22 -row 3
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22__4_
    -column 22 -row 4
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22__5_
    -column 22 -row 5
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22__6_
    -column 22 -row 6
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22__7_
    -column 22 -row 7
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22__8_
    -column 22 -row 8
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22__9_
    -column 22 -row 9
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22__10_
    -column 22 -row 10
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22__11_
    -column 22 -row 11
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22__12_
    -column 22 -row 12
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22__13_
    -column 22 -row 13
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22__14_
    -column 22 -row 14
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22__15_
    -column 22 -row 15
```

```
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22___16_
     -column 22 -row 16
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22___17_
     -column 22 -row 17
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22___18_
     -column 22 -row 18
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22___19_
     -column 22 -row 19
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22___20_
     -column 22 -row 20
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22___21_
     -column 22 -row 21
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22___22_
     -column 22 -row 22
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22___23_
     -column 22 -row 23
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22___24_
     -column 22 -row 24
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22___25_
     -column 22 -row 25
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22___26_
     -column 22 -row 26
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22___27_
     -column 22 -row 27
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22___28_
     -column 22 -row 28
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22___29_
     -column 22 -row 29
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22___30_
     -column 22 -row 30
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_22___31_
     -column 22 -row 31

add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23___0_
     -column 23 -row 0
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23___1_
     -column 23 -row 1
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23___2_
     -column 23 -row 2
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23___3_
     -column 23 -row 3
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23___4_
     -column 23 -row 4
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23___5_
     -column 23 -row 5
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23___6_
     -column 23 -row 6
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23___7_
     -column 23 -row 7
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23___8_
     -column 23 -row 8
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23___9_
     -column 23 -row 9
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23___10_
     -column 23 -row 10
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23___11_
     -column 23 -row 11
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23___12_
     -column 23 -row 12
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23___13_
     -column 23 -row 13
```

```
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23__14_
     -column 23 -row 14
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23__15_
     -column 23 -row 15
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23__16_
     -column 23 -row 16
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23__17_
     -column 23 -row 17
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23__18_
     -column 23 -row 18
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23__19_
     -column 23 -row 19
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23__20_
     -column 23 -row 20
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23__21_
     -column 23 -row 21
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23__22_
     -column 23 -row 22
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23__23_
     -column 23 -row 23
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23__24_
     -column 23 -row 24
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23__25_
     -column 23 -row 25
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23__26_
     -column 23 -row 26
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23__27_
     -column 23 -row 27
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23__28_
     -column 23 -row 28
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23__29_
     -column 23 -row 29
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23__30_
     -column 23 -row 30
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_23__31_
     -column 23 -row 31

add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24__0_
     -column 24 -row 0
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24__1_
     -column 24 -row 1
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24__2_
     -column 24 -row 2
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24__3_
     -column 24 -row 3
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24__4_
     -column 24 -row 4
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24__5_
     -column 24 -row 5
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24__6_
     -column 24 -row 6
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24__7_
     -column 24 -row 7
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24__8_
     -column 24 -row 8
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24__9_
     -column 24 -row 9
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24__10_
     -column 24 -row 10
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24__11_
     -column 24 -row 11
```

```
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24___12_
     -column 24 -row 12
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24___13_
     -column 24 -row 13
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24___14_
     -column 24 -row 14
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24___15_
     -column 24 -row 15
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24___16_
     -column 24 -row 16
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24___17_
     -column 24 -row 17
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24___18_
     -column 24 -row 18
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24___19_
     -column 24 -row 19
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24___20_
     -column 24 -row 20
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24___21_
     -column 24 -row 21
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24___22_
     -column 24 -row 22
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24___23_
     -column 24 -row 23
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24___24_
     -column 24 -row 24
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24___25_
     -column 24 -row 25
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24___26_
     -column 24 -row 26
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24___27_
     -column 24 -row 27
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24___28_
     -column 24 -row 28
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24___29_
     -column 24 -row 29
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24___30_
     -column 24 -row 30
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_24___31_
     -column 24 -row 31

add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___0_
     -column 25 -row 0
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___1_
     -column 25 -row 1
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___2_
     -column 25 -row 2
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___3_
     -column 25 -row 3
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___4_
     -column 25 -row 4
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___5_
     -column 25 -row 5
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___6_
     -column 25 -row 6
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___7_
     -column 25 -row 7
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___8_
     -column 25 -row 8
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___9_
     -column 25 -row 9
```

```
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___10_
     -column 25 -row 10
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___11_
     -column 25 -row 11
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___12_
     -column 25 -row 12
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___13_
     -column 25 -row 13
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___14_
     -column 25 -row 14
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___15_
     -column 25 -row 15
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___16_
     -column 25 -row 16
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___17_
     -column 25 -row 17
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___18_
     -column 25 -row 18
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___19_
     -column 25 -row 19
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___20_
     -column 25 -row 20
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___21_
     -column 25 -row 21
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___22_
     -column 25 -row 22
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___23_
     -column 25 -row 23
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___24_
     -column 25 -row 24
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___25_
     -column 25 -row 25
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___26_
     -column 25 -row 26
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___27_
     -column 25 -row 27
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___28_
     -column 25 -row 28
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___29_
     -column 25 -row 29
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___30_
     -column 25 -row 30
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_25___31_
     -column 25 -row 31

add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___0_
     -column 26 -row 0
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___1_
     -column 26 -row 1
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___2_
     -column 26 -row 2
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___3_
     -column 26 -row 3
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___4_
     -column 26 -row 4
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___5_
     -column 26 -row 5
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___6_
     -column 26 -row 6
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___7_
     -column 26 -row 7
```

```
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___8_
     -column 26 -row 8
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___9_
     -column 26 -row 9
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___10_
     -column 26 -row 10
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___11_
     -column 26 -row 11
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___12_
     -column 26 -row 12
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___13_
     -column 26 -row 13
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___14_
     -column 26 -row 14
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___15_
     -column 26 -row 15
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___16_
     -column 26 -row 16
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___17_
     -column 26 -row 17
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___18_
     -column 26 -row 18
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___19_
     -column 26 -row 19
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___20_
     -column 26 -row 20
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___21_
     -column 26 -row 21
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___22_
     -column 26 -row 22
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___23_
     -column 26 -row 23
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___24_
     -column 26 -row 24
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___25_
     -column 26 -row 25
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___26_
     -column 26 -row 26
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___27_
     -column 26 -row 27
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___28_
     -column 26 -row 28
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___29_
     -column 26 -row 29
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___30_
     -column 26 -row 30
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_26___31_
     -column 26 -row 31

add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___0_
     -column 27 -row 0
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___1_
     -column 27 -row 1
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___2_
     -column 27 -row 2
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___3_
     -column 27 -row 3
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___4_
     -column 27 -row 4
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___5_
     -column 27 -row 5
```

```
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___6_
    -column 27 -row 6
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___7_
    -column 27 -row 7
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___8_
    -column 27 -row 8
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___9_
    -column 27 -row 9
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___10_
    -column 27 -row 10
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___11_
    -column 27 -row 11
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___12_
    -column 27 -row 12
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___13_
    -column 27 -row 13
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___14_
    -column 27 -row 14
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___15_
    -column 27 -row 15
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___16_
    -column 27 -row 16
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___17_
    -column 27 -row 17
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___18_
    -column 27 -row 18
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___19_
    -column 27 -row 19
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___20_
    -column 27 -row 20
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___21_
    -column 27 -row 21
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___22_
    -column 27 -row 22
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___23_
    -column 27 -row 23
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___24_
    -column 27 -row 24
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___25_
    -column 27 -row 25
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___26_
    -column 27 -row 26
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___27_
    -column 27 -row 27
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___28_
    -column 27 -row 28
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___29_
    -column 27 -row 29
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___30_
    -column 27 -row 30
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_27___31_
    -column 27 -row 31

add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28___0_
    -column 28 -row 0
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28___1_
    -column 28 -row 1
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28___2_
    -column 28 -row 2
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28___3_
    -column 28 -row 3
```

```
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__4_
     -column 28 -row 4
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__5_
     -column 28 -row 5
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__6_
     -column 28 -row 6
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__7_
     -column 28 -row 7
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__8_
     -column 28 -row 8
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__9_
     -column 28 -row 9
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__10_
     -column 28 -row 10
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__11_
     -column 28 -row 11
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__12_
     -column 28 -row 12
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__13_
     -column 28 -row 13
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__14_
     -column 28 -row 14
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__15_
     -column 28 -row 15
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__16_
     -column 28 -row 16
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__17_
     -column 28 -row 17
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__18_
     -column 28 -row 18
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__19_
     -column 28 -row 19
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__20_
     -column 28 -row 20
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__21_
     -column 28 -row 21
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__22_
     -column 28 -row 22
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__23_
     -column 28 -row 23
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__24_
     -column 28 -row 24
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__25_
     -column 28 -row 25
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__26_
     -column 28 -row 26
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__27_
     -column 28 -row 27
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__28_
     -column 28 -row 28
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__29_
     -column 28 -row 29
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__30_
     -column 28 -row 30
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_28__31_
     -column 28 -row 31

add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__0_
     -column 29 -row 0
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__1_
     -column 29 -row 1
```

```
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__2_
    -column 29 -row 2
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__3_
    -column 29 -row 3
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__4_
    -column 29 -row 4
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__5_
    -column 29 -row 5
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__6_
    -column 29 -row 6
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__7_
    -column 29 -row 7
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__8_
    -column 29 -row 8
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__9_
    -column 29 -row 9
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__10_
    -column 29 -row 10
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__11_
    -column 29 -row 11
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__12_
    -column 29 -row 12
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__13_
    -column 29 -row 13
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__14_
    -column 29 -row 14
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__15_
    -column 29 -row 15
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__16_
    -column 29 -row 16
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__17_
    -column 29 -row 17
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__18_
    -column 29 -row 18
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__19_
    -column 29 -row 19
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__20_
    -column 29 -row 20
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__21_
    -column 29 -row 21
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__22_
    -column 29 -row 22
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__23_
    -column 29 -row 23
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__24_
    -column 29 -row 24
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__25_
    -column 29 -row 25
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__26_
    -column 29 -row 26
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__27_
    -column 29 -row 27
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__28_
    -column 29 -row 28
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__29_
    -column 29 -row 29
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__30_
    -column 29 -row 30
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_29__31_
    -column 29 -row 31
```

```
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__0_
     -column 30 -row 0
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__1_
     -column 30 -row 1
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__2_
     -column 30 -row 2
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__3_
     -column 30 -row 3
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__4_
     -column 30 -row 4
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__5_
     -column 30 -row 5
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__6_
     -column 30 -row 6
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__7_
     -column 30 -row 7
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__8_
     -column 30 -row 8
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__9_
     -column 30 -row 9
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__10_
     -column 30 -row 10
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__11_
     -column 30 -row 11
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__12_
     -column 30 -row 12
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__13_
     -column 30 -row 13
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__14_
     -column 30 -row 14
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__15_
     -column 30 -row 15
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__16_
     -column 30 -row 16
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__17_
     -column 30 -row 17
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__18_
     -column 30 -row 18
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__19_
     -column 30 -row 19
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__20_
     -column 30 -row 20
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__21_
     -column 30 -row 21
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__22_
     -column 30 -row 22
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__23_
     -column 30 -row 23
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__24_
     -column 30 -row 24
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__25_
     -column 30 -row 25
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__26_
     -column 30 -row 26
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__27_
     -column 30 -row 27
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__28_
     -column 30 -row 28
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__29_
     -column 30 -row 29
add_to_rp_group pil::rp1  -leaf U_CPU/U_REG_FILE/regfile_reg_30__30_
```

```
    -column 30 -row 30
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_30__31_
    -column 30 -row 31

add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__0_
    -column 31 -row 0
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__1_
    -column 31 -row 1
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__2_
    -column 31 -row 2
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__3_
    -column 31 -row 3
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__4_
    -column 31 -row 4
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__5_
    -column 31 -row 5
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__6_
    -column 31 -row 6
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__7_
    -column 31 -row 7
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__8_
    -column 31 -row 8
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__9_
    -column 31 -row 9
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__10_
    -column 31 -row 10
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__11_
    -column 31 -row 11
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__12_
    -column 31 -row 12
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__13_
    -column 31 -row 13
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__14_
    -column 31 -row 14
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__15_
    -column 31 -row 15
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__16_
    -column 31 -row 16
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__17_
    -column 31 -row 17
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__18_
    -column 31 -row 18
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__19_
    -column 31 -row 19
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__20_
    -column 31 -row 20
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__21_
    -column 31 -row 21
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__22_
    -column 31 -row 22
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__23_
    -column 31 -row 23
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__24_
    -column 31 -row 24
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__25_
    -column 31 -row 25
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__26_
    -column 31 -row 26
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__27_
    -column 31 -row 27
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__28_
```

```
    -column 31 -row 28
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__29_
    -column 31 -row 29
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__30_
    -column 31 -row 30
add_to_rp_group pil::rp1 -leaf U_CPU/U_REG_FILE/regfile_reg_31__31_
    -column 31 -row 31
```