FPGA implementation of a public key block cipher

Stig Fjellskaalnes

December 17, 2008

Abstract

This report will cover the VHDL implementation of a public key algorithm based on multivariate quadratic quasigroups. Results from the simulation and synthesis will be analyzed and discussed.

Contents

1 Project Assignment **9**

2 Theory **11**
2.1 Public key algorithms in general . . . 11
2.2 RSA . . . 11
2.3 Diffie-Hellman . . . 12
2.4 Elliptic curve cryptography . . . 12
2.5 MQQ . . . 12

3 Hardware Implementation **13**
3.1 Why hardware implementation? . . . 13
3.2 Decryption . . . 13
3.3 Private Matrix . . . 14
3.3.1 Original implementation of Private Matrix . . . 14
3.3.2 Modified implementation of Private Matrix . . . 16
3.4 Dobbertin ROM . . . 17
3.5 Sequencer . . . 17
3.5.1 Master ROM . . . 19

4 Results **21**
4.1 Synthesis of the design . . . 21
4.2 Simulation . . . 21
4.2.1 Decryption . . . 21
4.2.2 Private Matrix . . . 22
4.2.3 Dobbertin ROM . . . 22
4.2.4 Sequencer . . . 22

5 Discussion and conlusion **25**
5.1 Discussion . . . 25
5.1.1 Synthesis . . . 25
5.1.2 Implementation and simulation . . . 26
5.2 Conclusion . . . 28
5.3 Contributors . . . 28

6 Abbreviations **29**

A Code listing **33**
A.1 VHDL code . . . 33
A.1.1 Decryption . . . 33
A.1.2 Private Matrix T . . . 38

A.1.3 Dobbertin ROM 41
A.1.4 Sequencer 43
A.1.5 Master ROM 48
A.2 C/C++ code 51

B Synthesis report **57**

List of Figures

3.1 Decryption top level block diagram 14
3.2 Decryption top level state diagram 15
3.3 Private Matrix block diagram 16
3.4 Private Matrix state diagram 17
3.5 Dobbertin ROM block diagram 18
3.6 Sequencer block diagram 18
3.7 Sequencer state diagram 19
3.8 Master ROM block diagram 20

4.1 Simulation of startup 22
4.2 Simulation of Private matrix 22
4.3 Simulation of Dobbertin ROM 23
4.4 Simulation of Sequencer 23
4.5 Simulation of Master ROM 23

5.1 Problem with Private Matrix 26
5.2 Dobbertin ROM simulation deviation 27
5.3 Master ROM simulation problem 27

List of Tables

3.1 Decryption algorithm 15

4.1 Synthesis results of the Multivariate Quadratic Quasigroups (MQQ) decryption procedure 21

Chapter 1

Project Assignment

Student's name:	Stig Fjellskaalnes
Course:	TTM4530 Information Security Specialization Course
Title:	FPGA Implementation of a public key block cipher

Description (composed by the student):
Recently, a new public key algorithm have been proposed by Gligoroski, Markovski and Knapskog [2]. The algorithm belongs to the class of public keys algorithms realized by multivariate quadratic equations. The authors found out a new class of quasigroups that have special form when expressed as Boolean functions. The quasigroups are multivariate quadratic.

One important characteristic for this new public key algorithm is that it is very fast. Realized in software it can produce digital signatures around 300 times faster than RSA (1024 bit public key length). However, in hardware the algorithm can achieve speeds equivalent to symmetric key primitives both in signature generation and in its verification. That means the algorithm realized in hardware can be 1,000 to 10,000 times faster than corresponding public key algorithms (RSA, Diffie-Helman or Elliptic Curve algorithms) realized also in hardware.

The student will have a task to write a VHDL code and to realize the algorithm in FPGA.

Deadline:	17/12-2008
Submission date:	17/12-2008
Department:	**Department of Telematics**
Supervisors:	**Danilo Gligoroski, NTNU** **Mohamed El-Hadedy, NTNU**

Trondheim, 2008

Danilo Gligoroski **Date**

Chapter 2

Theory

In this chapter the most used public key algorithms will be presented.

2.1 Public key algorithms in general

A public key algorithm is an assymetric crypto-algorithm, meaning that a key owner uses two different keys. One private key that is secret for everybody but the key owner, the other key is known for others. This means if a key owner wants to send a message to another person encrypted, the sender encrypts the message with the receiver's public key that is known to all. When the recipient receives the message, the receiver uses its own private key to decrypt the message, verifying that the message or data is authentic. Given that the key is not broken, this is also valid for digital signature.

2.2 RSA

The Rivest-Shamir-Adleman (RSA) public key algorithm was first presented by R. L. Rivest, A. Shamir, and L. Adleman from Massachusetts Institute of Technology (MIT) laboratory in 1977 [1], which means that the RSA has existed for a very long time, and is still being widely used. Areas where RSA is in use are in Secure SHell (SSH) keys, Secure Socket Layer (SSL) X.509 certificates, digital signatures and so on. The encryption is done by doing the arithmetric operation $C = M^e \pmod{n}$, where C is the encrypted message, M is the original message and n is the product of two prime numbers p and q. n is used as a modulus. Decryption is done by doing an arithmetric operation the same way as in encryption, $M = C^e \pmod{n}$.

This algorithm is sequential in nature. Since the algorithm is based on calculating the given equation, this will make the algorithm slow, especially when the key size increases. If an RSA key is to be secure enough today, the key size must be at least 1024 bit, preferably 2048 bit, or even 4096 bit, as the computing power increases. The calculation will take more and more time, as the key size increases. Doing "pen and paper" calculations of $C = M^e \pmod{n}$ is time-consuming and not very efficient. Therefore, the Montgomery algorithm is a perferrable method for multiplication and modular exponentiation [4], which will increase performance of an RSA implementation. This is especially important in hardware implementations of RSA, but also in software implementations.

2.3 Diffie-Hellman

Diffie-Hellman (DH) key exchange is another public key algorithm, invented in 1976 by Whitfield Diffie and Martin Hellman. It was the first algorithm for distributing a shared secret over an unsecured communication channel [5]. DH itself is an anonymous key-agreement protocol, but is still being used as a basis for several authenticated protocols. OpenVPN, an open-source VPN server protocol based on OpenSSL, uses DH in form of a parameter as one of the server keys. It also provides perfect forward secrecy in Transport Layer Security (TLS) ephemeral modes.

DH works as follows.

1. Person A and B select a prime number p and a base g
2. A chooses a secret integer i, then sends B ($g^i \bmod p$)
3. B chooses a secret integer j, then sends B ($g^j \bmod p$)
4. A computes $(g^j \bmod p)^i \pmod p$
5. B computes $(g^i \bmod p)^a \pmod p$

For more information about DH, see [5].

2.4 Elliptic curve cryptography

Elliptic curve cryptography (ECC)[7] is a public key algorithm which uses the algebraic structures of elliptic curves[6]. It was proposed independently by Neal Koblitz and Victor S. Miller in 1985.

2.5 MQQ

MQQ is a new public key algorithm porposed by Danilo Gligoroski, Svein J. Knapskog and Smile Markovski. This algorithm was first published in 2008 [2]. In contradiction to older public key algorithms, this algorithm can easily be implemented to run in parallel. This has dramatic impacts on speed. Early implementations of this algorithm has shown tremendous performance compared to similar implementations of existing public key algorithms, such as RSA, DH and ECC [3].

Another reason why this algorithm is fast is that all calculations are based on simple logical operations such as AND and XOR instead of more complex arithmetric operations, for instance the operation done in RSA.

Chapter 3

Hardware Implementation

In this chapter, the actual hardware implementation of the MQQ algorithm will be presented. This report covers the implementation of the decryption procedure of MQQ. The hardware description language used is Very-High-Speed Integrated Circuits (VHSIC) Hardware Description Language (VHDL). Xilinx ISE 10.1 is used as the synthesis tool for the design, and ModelSim SE is used for simulation and verification of the design. This algorithm has been implemented earlier, by Mohamed El-Hadedy. The first implementation was designed for speed only. This implementation will have a stronger focus on reducing area cost, therefore it will run slower than the first implementation. Another moment that is important to mention is that this report describes the implementation of the functionality. Real data or keys have not been used in this design, which will be discussed more thoroughly in the Discussion section of this report. Shortened versions of the VHDL source code are listed in the appendix and the full source code files are located on a CD added to the report. Physical realization on an Field-programmable Gate Array (FPGA) has not been possible due to lack of an FPGA development board. The focus will be on design, synthesis and simulation.

3.1 Why hardware implementation?

In order to fully demonstrate the speed of an algorithm, a hardware implementation is needed. In a software environment a program runs on a microprocessor. There the speed is dependent on the CPU utilization. Hardware implementations are useful because the encryption/decryption, and in many cases, key generation, will use dedicated hardware in its operation. This will increase speed, decrease delay and use of resources. In systems where hardware implementations of a crypto algorithm is used, the hardware implementations is referred to as a hardware accelerator. Hardware accelereators are especially useful in embedded systems when processing resources are limited. The procedure of encryption/decryption does not change, only the keys and data to be used. Therefore, in an embedded system it will save time, and probably energy to implement the algorithm in hardware.

3.2 Decryption

This is the top module of the decryption procedure in MQQ. As shown in figure 3.1, Decryption has four subcomponents, Private Matrix T, Dobbertin Read Only Memory (ROM), Sequencer and Private Matrix S. The Decryption top module is implemented as a clocked Moore type Finite State Machine (FSM). The design is fully synchronous without combinatorial paths except the decode of next state, and with synchronous active high reset. The main reason for this choice is

that there are many signals in the design that must be remembered in every state. It is possible to define variables, as normally used in software programming. But when it comes to hardware implementation variables are not synchronous. This might generate timing problems, which will make it harder to detect and correct possible bugs caused by timing problems. A variable won't appear in the simulation done in ModelSim either, which makes it difficult to test and verify their correctness. Therefore, only signals are used in the entire design. In a combinatorial circuit (a circuit without a clock signal) all signals must be set in all states to avoid latches. This will result in much unnessecary code, so to save design time and code lines, the design is made entirely synchronous, except the next state decode. Another reason why the design is made synchronous is to have better control with timing. Latches are signals that don't change values, and can cause timing problems if they exist in the design. Other unexpected behavior might occur when latches exist. In addition to set the right values in each signal, the signal must also arrive at the right time. To accomplish this, the easiest way is to have a synchronous design. The state diagram is shown in figure 3.2 and the decryption algorithm for MQQ is listed in table 3.2.

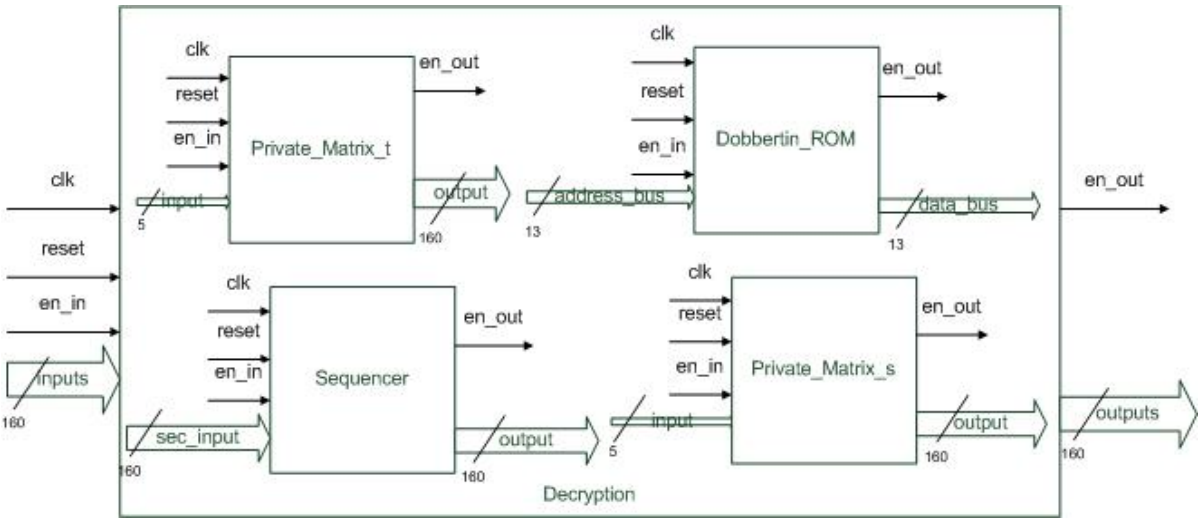

Figure 3.1: Decryption top level block diagram

3.3 Private Matrix

There are two instances of Private Matrix, Private_Matrix_T and Private_Matrix_S. What it does is to implement the step 1 and 7 in the decryption sequence of MQQ. Private_Matrix_T implements step 1, Private_Matrix_S implements step 7. In this design Private_Matrix_T and Private_Matrix_S is instantiated by the same code, which means that there are two instances of the same implementation of Private Matrix. Many of the code lines are almost identical, since there are 160 chains inside Private Matrix, as shown in figure 3.3. That means that there exist 160 from_rom signals, 160 AND operations and so on. Small programs in C or C++ has been written to write these lines, source code of these programs have been added in the appendix. The same thing has been done for Dobbertin ROM and Master ROM.

3.3.1 Original implementation of Private Matrix

In the first implementation of MQQ the Private Matrix takes in a vector of length 160 bits. It then does a logical AND operation with a 160 bit vector stored inside a ROM. The ANDed

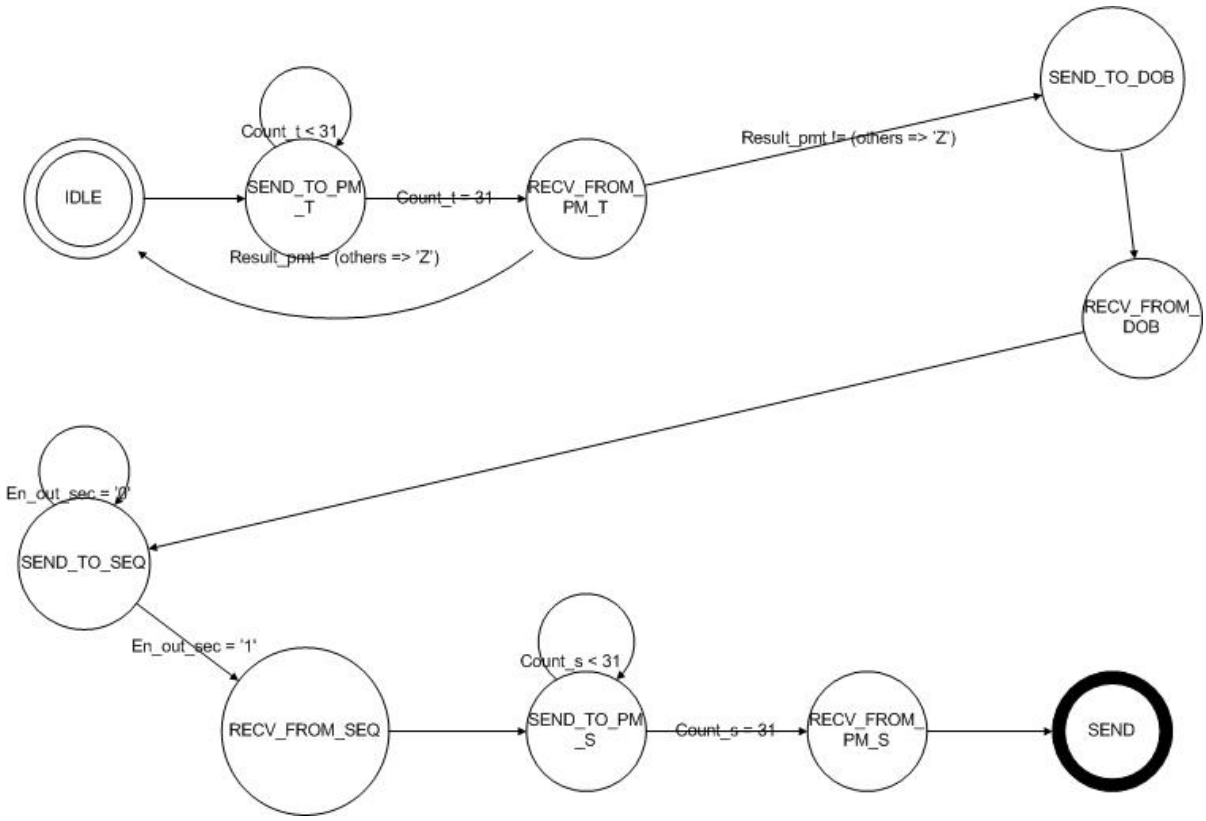

Figure 3.2: Decryption top level state diagram

Algorithm for decryption with the private key $(T, S, *_1 ... *_8)$
Input: A vector $y = (y_1, ..., y_n)$
Output: a vector $x = (x_1, ..., xn)$ such that $\mathbf{P}(x) = y$
1. Set $y' = T^{-1}(y)$ 2. Set $W = (y'_1, y'_2, y'_3, y'_4, y'_5, y'_6, y'_{11}, y'_{16}, y'_{21}, y'_{26}, y'_{31}, y'_{36}, y'_{41}) = \mathbf{Dob}^{-1}(W)$ 3. Compute $Z = (Z_1, Z_2, Z_3, Z_4, Z_5, Z_6, Z_{11}, Z_{16}, Z_{21}, Z_6, Z_{31}, Z_{36}, Z_{41})$ 4. Set $y_1 \leftarrow Z_1$, $y_2 \leftarrow Z_2$, $y_3 \leftarrow Z_3$, $y_4 \leftarrow Z_4$, $y_5 \leftarrow Z_5$, $y_6 \leftarrow Z_6$, $y_{11} \leftarrow Z_7$, $y_{16} \leftarrow Z_8$, $y_{21} \leftarrow Z_9$, $y_{26} \leftarrow Z_{10}$, $y_{31} \leftarrow Z_{11}$, $y_{36} \leftarrow Z_{12}$, $y_{41} \leftarrow Z_{13}$ 5. Represent y' as $y' = Y_1 ... Y_k$ where Y_i are vectors of dimension 5 6. By using the left parastrophes $\backslash_i$ of the quasigroups $*_i$, $i = 1, ..., 8$, obtain $x' = X_1 ... X_k$, such that: $X_1 = Y_1$, $X_2 = X_1 \backslash_1 Y_2$, $X_3 = X_2 \backslash_2 Y_3$ and $X_i = X_{i-1} \backslash_{3+((i+2) mod 6} Y_i$ 7. Compute $x = S^{-1}(x')$

Table 3.1: Decryption algorithm

vector is then XORed bit by bit, so that the 160bit vector is reduced to a single bit. The XORed value is then placed in a synchronization register called Register_X. The Register_X is the result, and is sent to component Dobbertin_ROM. This is the description of the chain in Private Matrix, and since the input vector and output vector is 160 bits, the Private Matrix consists of 160 chains as described.

3.3.2 Modified implementation of Private Matrix

Since input vectors at 160 bits use much area on an integrated circuit, modifications of the original design is needed so that area can be saved. The modified implementation will take in 32 vectors of length 5, once at a time. The vectors stored in a ROM is now implemented as an array with 32 vectors of length 5, which corresponds to the input value. Prvate Matrix is implemented according to block diagram shown as figure 3.3

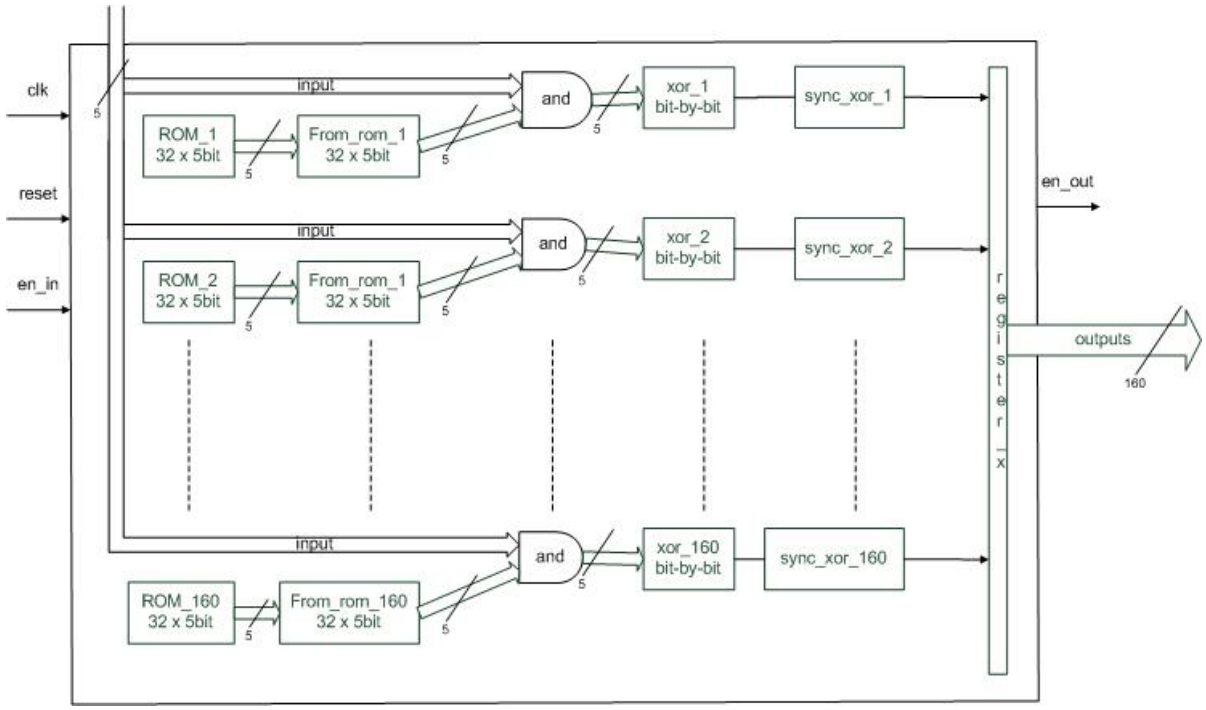

Figure 3.3: Private Matrix block diagram

The Private Matrix component is implemented as a clocked Moore FSM with five different states, IDLE, ANDOP, XORING, SYNC and PUSH_OUT, as shown in the figure 3.4.

In IDLE, values from the ROMs are written to corresponding signals and en_out is set to '0' (low). When this is done and the control logic has verified the writing to the signal from_rom, state ANDOP is initiated.

In ANDOP, the logical AND operation is done between the input vector and the corresponding stored vector from the ROM. To illustrate this better, when the counter (cnt) has value 10, the Private Matrix runs at 11th time. Signal from_rom_xxx (xxx is a number between 1 and 160) will have stored the 11th vector from the ROM array. When this is done, ANDOP state is finished, and next state will be XORING.

The bit-by-bit XOR operation takes place in the state XORING. This state uses an internal counter, count_xor to keep track of how may times the XOR opeation is done. A temporary signal, tmp_xxx is used to store the temporary value corresponding to the position of the vector and_rom_in_xxx. An XOR is then done between the tmp_xxx and and_rom_in_xxx at position

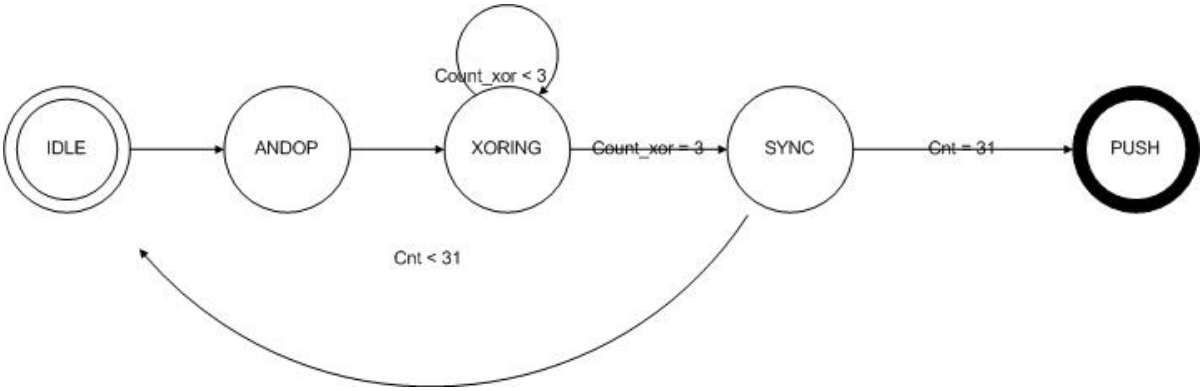

Figure 3.4: Private Matrix state diagram

counter+1. When the counter reaches 3, the 5 bit result from anding input and the stored ROM vector is bit-by-bit XORed into a single bit, and the state machine shifts state to SYNC.

A modification from the original design is that a new step is being introduced, a SYNC state. This is to keep syncronization, and to complete the XOR step. Here is a description why this step is needed. When the Private Matrix runs for the first time, the sync_xor_xxx get the value from the xor_xxx, the result after the bit-by-bit XOR operation. For the second run and so on, a new XOR operation is initiated between the xor_xxx and the sync_xor_xxx, the latter signal has the value of the previous XOR operation. This is necessary to make sure that the bit-by-bit XOR operation runs as many times as it is supposed to. And when this operation is done, en_out is set to '1' (high). That means that the value can be written to register_x, a synchronization register for the XORed bits.

When the counter reaches value 31, it means that the register_x contains the result, and the Private Matrix is ready to send the data to Dobbertin_ROM component. This is done in state PUSH_OUT.

3.4 Dobbertin ROM

The module Dobbertin ROM is simply a ROM with $2^{13} * 13$ bits size. It implements step 2, 3 and 4 in the decryption procedure of MQQ and is implemented according to figure 3.5 The decryption top module makes a 13 bit vector of the resultant (ref. table 3.2, step 2) vector from Private_Matrix_T which acts as an address. This vector is sent into the ROM. The ROM then selects the stored value corresponding to the address (ref. table 3.2), and pushes this value on the outgoing port, data_bus. The 13 bit data vector is then written back to the 160 bit register in the same order as the address vector was made from the Private_Matrix_T resultant vector (ref. table 3.2, step 4).

3.5 Sequencer

The sequencer module implements step 5 and 6 in the decryption procedure. It is the most complex module in the design, as seen in figure 3.6. The state diagram is listed in figure 3.7.

Again, a FSM type Moore is used to implement the functionality of the sequencer. In the INIT state, the 160bit input signal from the Dobbertin ROM is split up in an array consisting of 32 vector with length of 5 bits. When this is done, current state changes to MUX2_OUT. Here the first element of the array is being sent through the multiplexer since the selector is

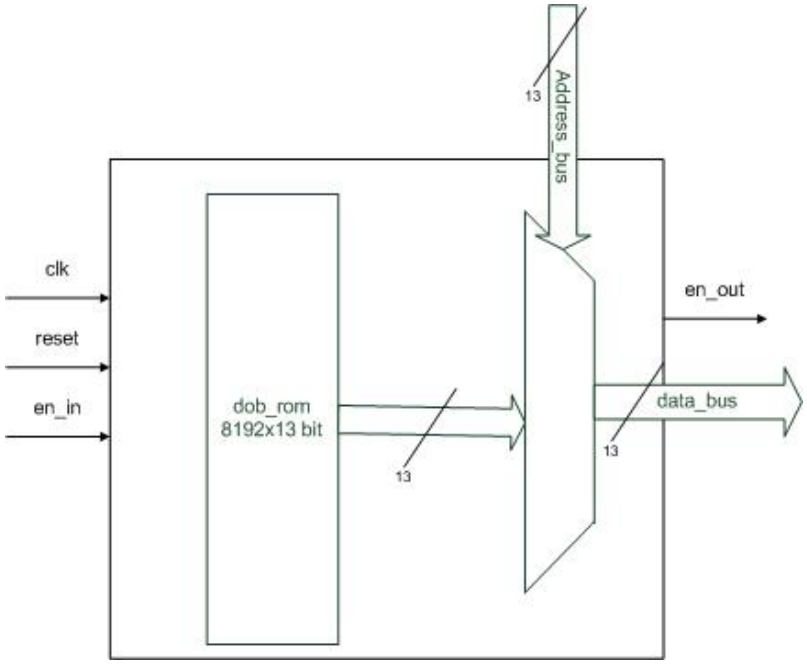

Figure 3.5: Dobbertin ROM block diagram

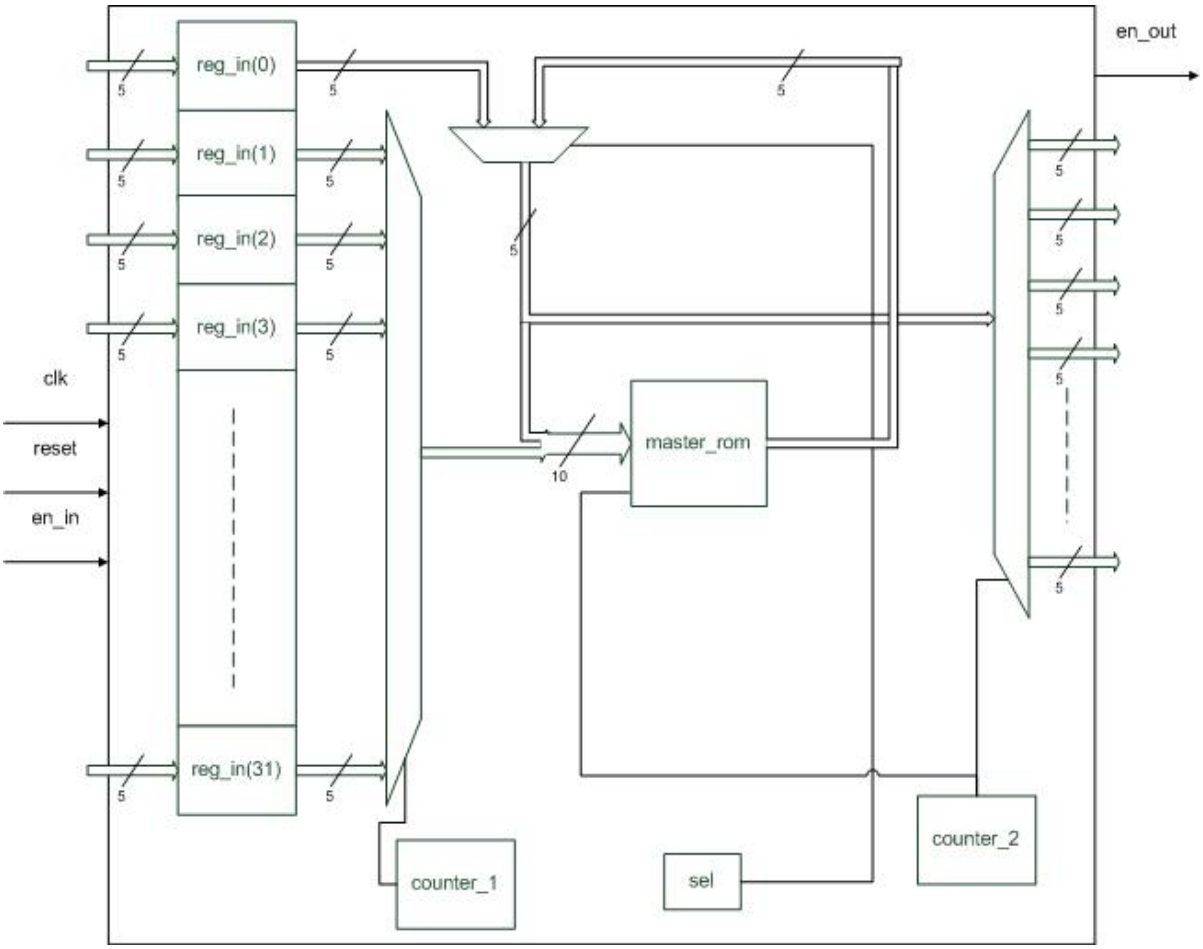

Figure 3.6: Sequencer block diagram

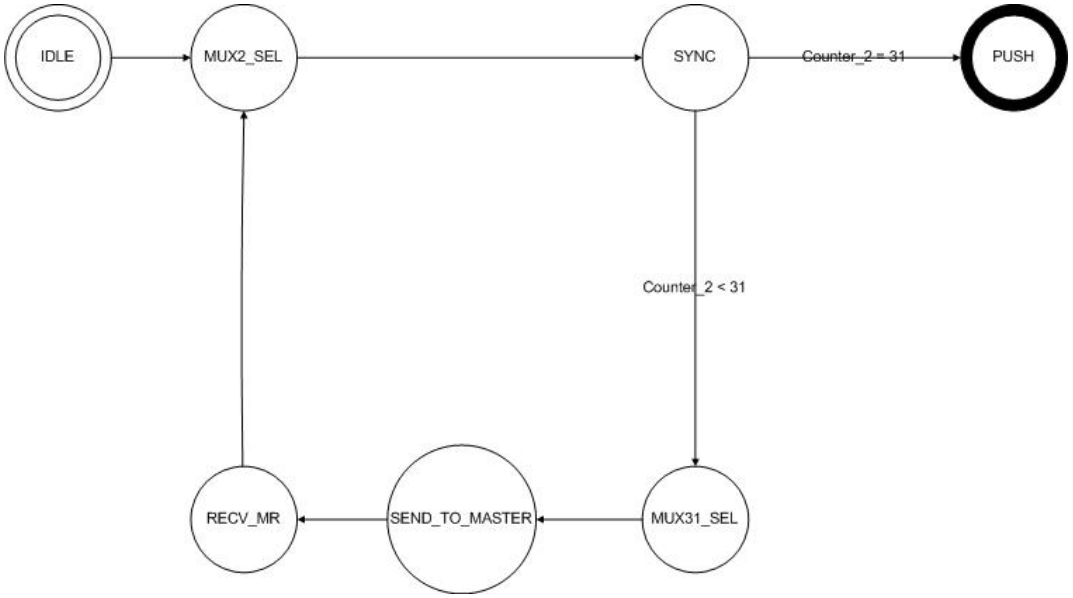

Figure 3.7: Sequencer state diagram

set to '0',and becomes the first element in a similar array which will be the result that the secuencer module generates. The first element is written in the output array in state SYNC. The counters are increased by 1, and the state machine shifts to state MUX31_out. Selector of MUX_2 is set to '1' because the values that are supposed to go through that MUX will not be the first element. Then the output from MUX_31 will be the 5 least significant bits in a 10 bit vector, the most sigificant bits are the 5 bit vector from MUX_2. The 10 bit vector is an address, which is the input to the Master_ROM. After Master_ROM has done its job, the result will go through MUX_2 and written on the output register, and also be used as feedback for the new address to be sent into Master_ROM. This procedure will continue until counter_2 reaches value 31, which means that all 160 bits are processed by the sequencer and are ready to be written to the output register, and the value is used by Private_Matrix_S as input.

3.5.1 Master ROM

The Master ROM is a subcomponent to the sequencer. It has 8 ROM units consisting of 5 bit vectors. Each ROM has an address space of 10 bits, which means that each ROM has 2^{10} vectors stored. There is also a control ROM, which has 2^5 3bit vectors, which is a selector that determines which of the 8 ROM is to be used. The control ROM is being controlled by a counter. In this implementation this counter is located in the sequencer(counter_2), and the value from the counter is one of the input ports. This is shown in figure 3.8.

When a 10 bit address is generated in the sequencer (a join of the values from mux31 and mux2 with the value from mux2 as the most significant bits), the address is written on the output port. The Master ROM then selects the corresponding 5 bit value from all 8 ROMs. Then the counter selects a value from the control ROM, and it is this value who selects which data is to be written on the data_bus output port.

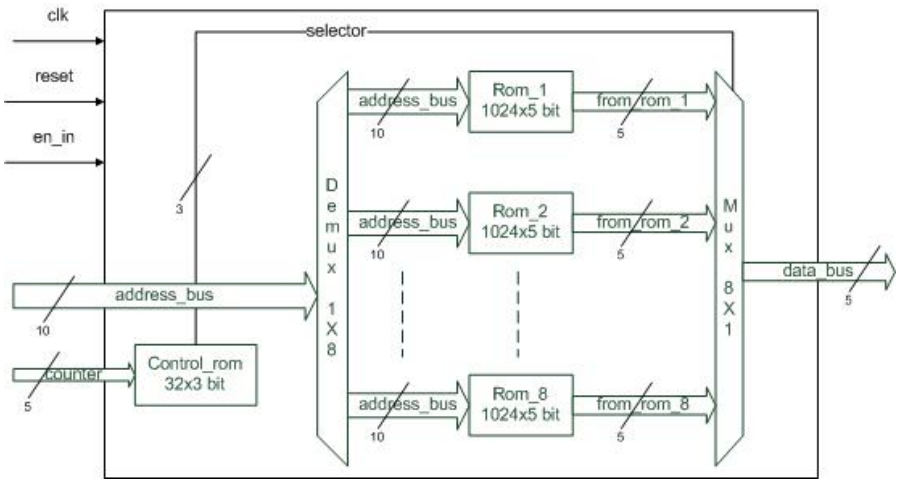

Figure 3.8: Master ROM block diagram

Chapter 4

Results

In this chapter, the results from the simulation will be presented and analyzed.

4.1 Synthesis of the design

The design has been synthesized against two different FPGAs. These are Xilinx Virtex 2, model xc2v1000-fg456 (speed grade -4) and Xilinx Virtex 5, xc5vfx70-1ff1136 (speed grade -1). The Virtex 5 FPGA has a PowerPC CPU core embedded. The reason that two different FPGAs were used was to illustrate differences in area cost and why a modern FPGA will be required for further implementation of this algorithm. Table 4.1 shows an overview of the synthesis results against the two different FPGAs, the full synthesis report for Virtex 5 is added in the appendix.

4.2 Simulation

To verify the functionality, simulation has been done to make sure that the components work as intended and uncover design errors. All the components have been simulated and verified separately during the design period, and the design has also been simulated as one unit.

4.2.1 Decryption

This is the top module which controls all sub components. It is in Decryption the data flow from one module to the next is controlled. Figure 4.1 shows the startup for the Decryption.

When reset = '1' (high), the most important signals are set to '0' (low). This is necessary to rule out odd behaviour from the start, and to make sure that the right initial values are set in the different signals. Before en_in is set to '1', there is a period of three clock sycles where both reset and en_in is '0'. This is being done to make sure nothing happens. The intension is that the design should run only when every component's en_in = '1'.

FPGA	Slice Registers	Slice Register usage	LUTs	LUT usage	Frequency
xc2v1000-fg456	6,629	64%	17,773	173%	90.754 MHz
xc5vfx70-1ff1136	5,910	13%	7,703	17%	214.000 MHz

Table 4.1: Synthesis results of the MQQ decryption procedure

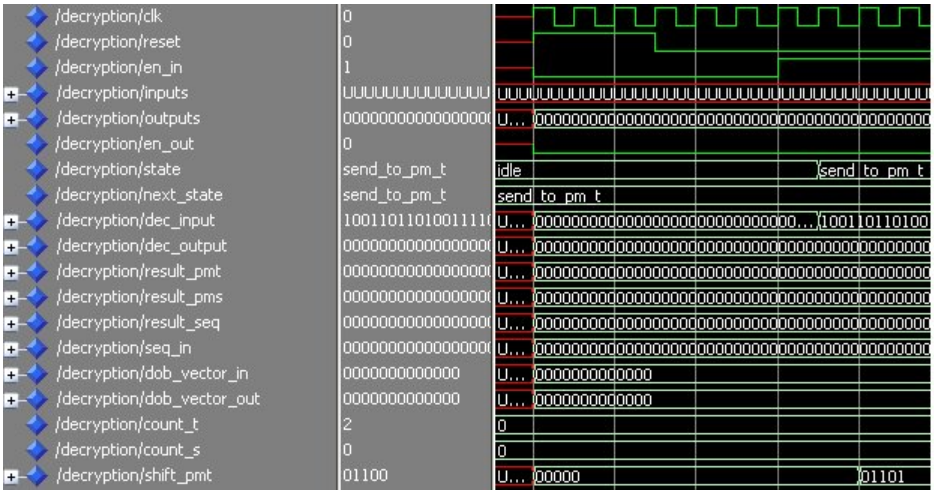

Figure 4.1: Simulation of startup

4.2.2 Private Matrix

Simulation of Private_Matrix_T and Private_Matrix_S is shown in figure 4.2. Since there are two instances of Private Matrix (two instances from the same source code), only simulation results from Private_Matrix_T will be presented at first.

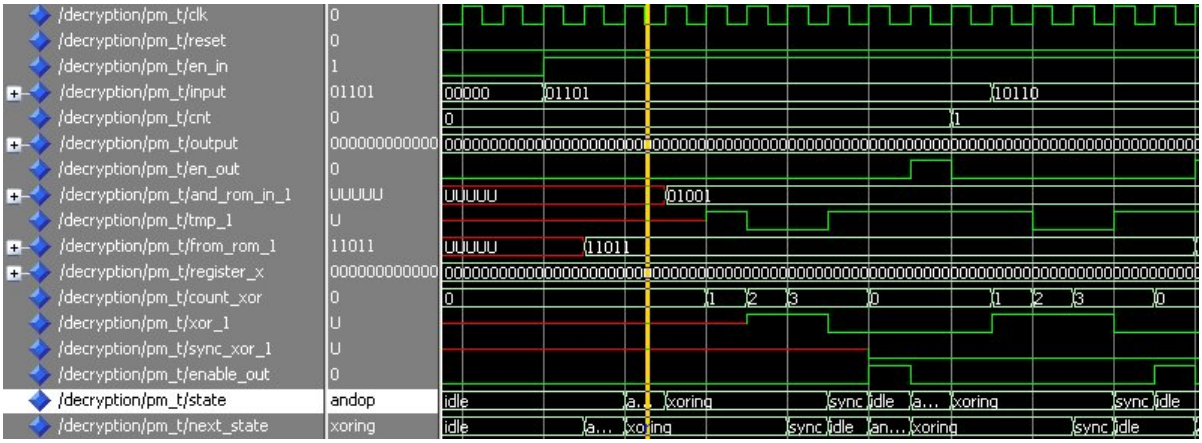

Figure 4.2: Simulation of Private matrix

4.2.3 Dobbertin ROM

The only thing Dobbertin ROM does, is to select a value from ROM based on the 13 bit address_bus which serves as an input. Figure 4.3 shows that this is the case.

4.2.4 Sequencer

Simulation results from the startup of sequencer is shown in figure 4.4.

As mentioned before, the sequencer will not process the input vector before the sequencer's en_in signal is set to '1'. From there, the sequencer shows behavior as expected. But this will be further discussed in the chapter Discussion.

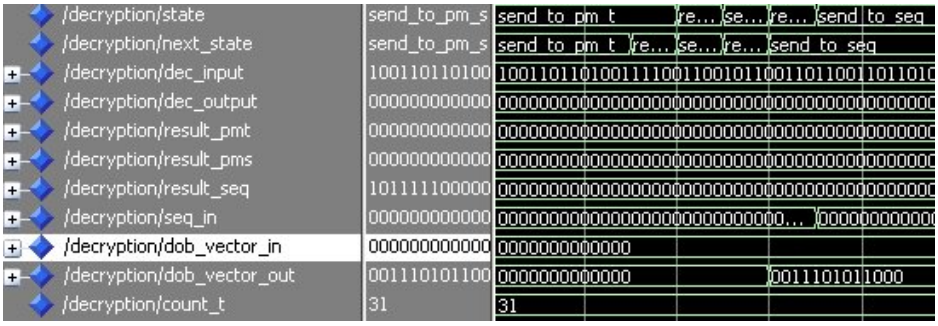

Figure 4.3: Simulation of Dobbertin ROM

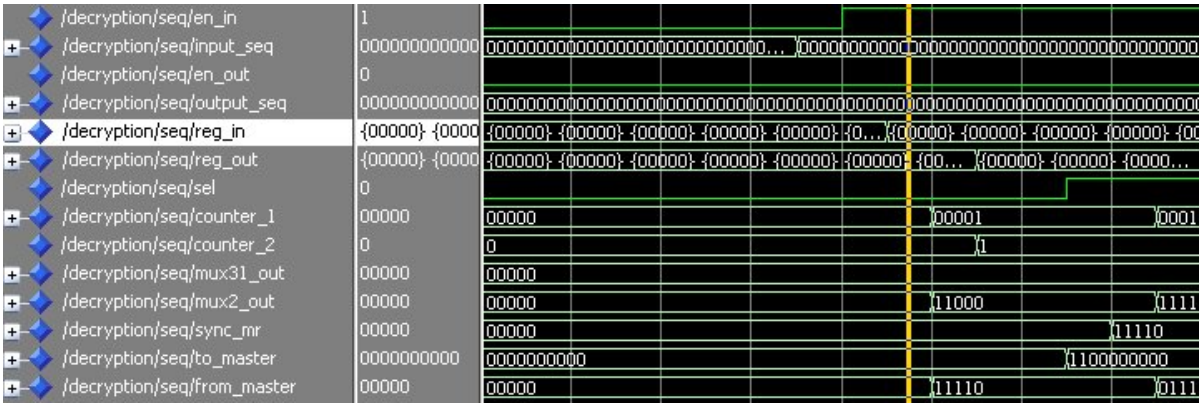

Figure 4.4: Simulation of Sequencer

Master ROM

The Master ROM component is an underlying circuit which is being controlled by the sequencer. Simulation results are presented in figure 4.5.

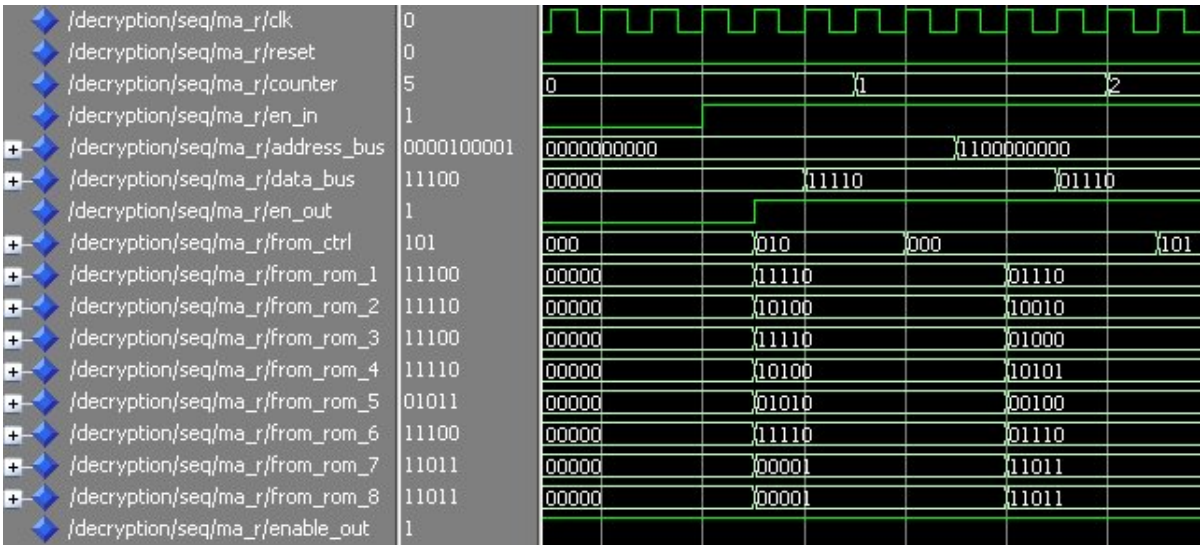

Figure 4.5: Simulation of Master ROM

Chapter 5

Discussion and conlusion

5.1 Discussion

Here the results from the synthesis and simulation will be analyzed and discussed. The synthesis uncovered problems with the design that should be analyzed and corrected, as part of future improvements. So, suggestions for future work will also be presented in this section.

5.1.1 Synthesis

As the synthesis results indicated by table 4.1, the design takes up quite much area of the FPGA. The oldest FPGA the design was synthesized aginst, xc2v1000-fg456 didn't have enough available resources so that the design could be physically implemented onto this FPGA. With the use of xc5vfx70-1ff1136 as the FPGA, the design could be implemented. But the results also show that physically area consumption of the design is significally higher on the older xc2v1000 than using xc5vfx70. The reason for this is that xc2v1000 offer 4-input Look-Up Table (LUT)s, but the newer xc5vfx70 offer 6-input LUTs. This causes more efficient area exploitation, therefore the number of used LUTs are lower with xc5vfx70 than with xc2v1000. By doing synthesis against two different FPGAs made it possible to see wether this design was possible to realize on an older FPGA. The xc2v1000 is available for use when needed without spending financial resources.

Another problem that was uncovered after the synthesis was seemingly caused by the synthesis tool itself. Both FPGAs have dedicated Random Access Memory (RAM) blocks available, socalled block RAM. The main reason that this design takes up so much resources of an FPGA is because of all the ROM blocks in the design. There are many ROM blocks, and especially the ROMs in Dobbertin ROM and Master ROM use a large amout of resources on the FPGA ($2^{13} * 13$ bit for Dobbertin ROM and eight $2^{10} * 5$ bit ROM units inside the Master ROM). During synthesis one of the messages was that the synthesis tool was unable to implement the ROM blocks as read-only block RAM. Instead LUTs were used to implement the ROMs. The user was left to look for further software updates. It is clear that if the ROM blocks had been implemented as read-only block RAM, the area usage would been significally reduced. The reason for this can either be that the synthesis tool currently is unable to use the block RAM resources, that the ROMs have not been defined properly in the source code, or that the ROMs don't fit the shape of the block RAM. Further investigation of this problem is needed, because solving this problem can reduce area consumption tremendously.

Synthesis against xc5vfx70 also showed that this implementation infact takes up larger area than the original implementation [3]. The reason for this is at this point unknown and has to be further investigated and corrected.

5.1.2 Implementation and simulation

During the simulation, some irregularities were uncovered in some of the modules. It is possible that these problems can cause the hardware implementation to produce unwanted results.

Private Matrix

During the Private Matrix operation (both Private_Matrix_T and Private_Matrix_S) the 160 bit output vector only consisted of '0'. Figure 5.1 shows this.

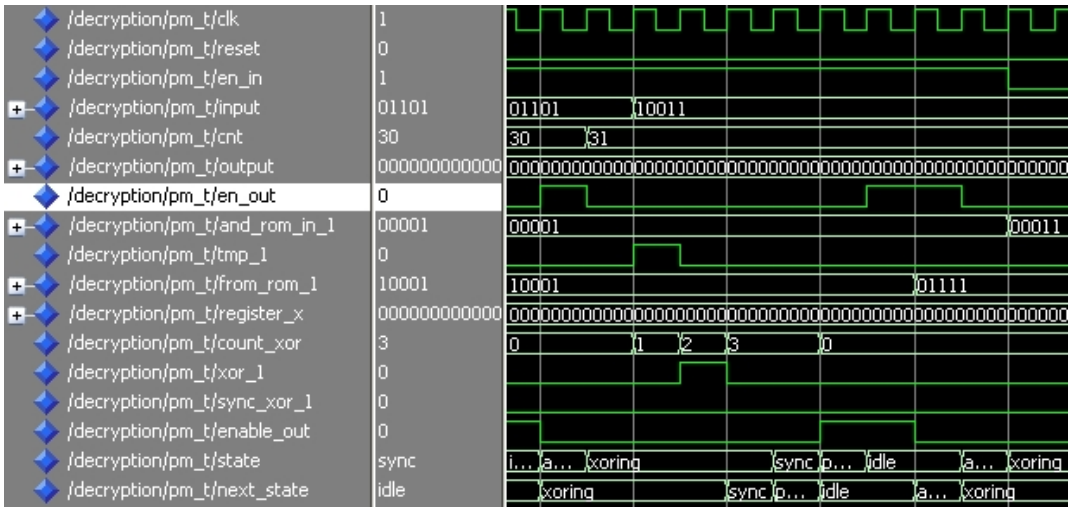

Figure 5.1: Problem with Private Matrix

Though it seems that the Private Matrix state machine works as expected, it is unclear why the register_x vector, and the following output vector elements always are '0'. It can have something to do with the values itself. The input vector is to be ANDed with a corresponding ROM vector, and then bit-by-bit XORed downto a single bit. It is possible that the current values, both the test vector defined as dec_input in the Decryption top module and the stored values inside the ROMs in Private Matrix after an AND and bit-by-bit XOR can give '0' as a result. All ROMs have the same values in this implementation, so if one chain gives '0', all chains will give '0' at the end.

There was difficult to create ROMs with different values inside Private Matrix. Therefore, future work will include giving different values to the 160 ROMs inside Private Matrix. Hopefully, the simulation result will then be different than showed here.

Dobbertin ROM

When reset = '1' all signals defined when reset = '1' will be set to '0' at once. But for en_out to be '0' after a reset, it takes one extra clock cycle, as shown in figure 5.2. But the reason for this is that en_out always has the current value of en_in. Therefore, when en_in is forced to '0' in the simulation, it will take one extra clock cycle before en_out is '0'. In general, when en_in changes value, it will take one clock cycle before en_out gets the same value. The Dobbertin ROM works as expected anyway, so this doesn't impact the results. But a problem will be that this signal doesn't reset properly when reset = '1'.

Figure 5.2: Dobbertin ROM simulation deviation

Master ROM

As figure 4.5 shows, Master ROM works as expected at first. But after Master ROM has been running a few times, it starts to show abnormal behaviour, as figure 5.3 will show.

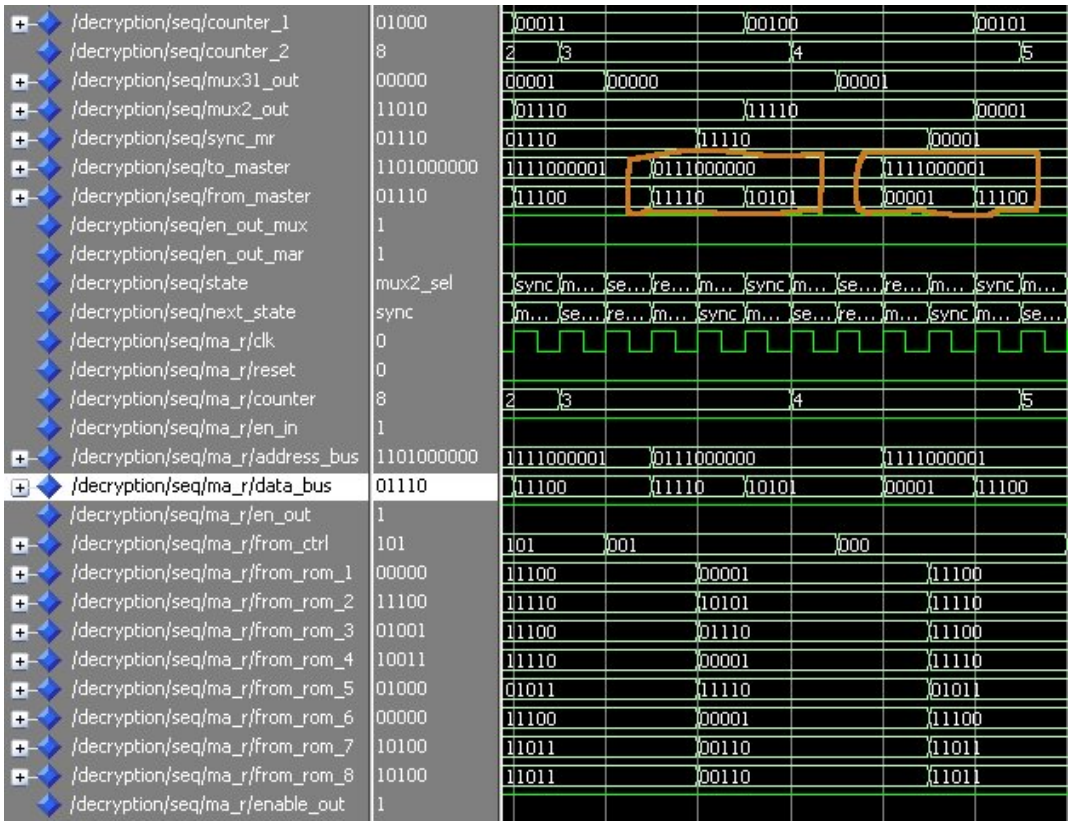

Figure 5.3: Master ROM simulation problem

When counter in Master ROM increases by 1, it takes one clock cycle before the mux31 sends out the next value. After two clock cycles the address_bus is sent into the Master ROM. At the same time, a value is written on the data_bus. Accordingly, this value is written to sync_mr, the syncronization register in the sequencer for values from Master ROM on the next clock cycle, and in the next clock cycle this value is sent through the mux2 in the sequencer. But a new value is written on the data_bus inside Master ROM two clock periodes after the

data_bus has been written first time after the counter increase. This incident is marked in figure 5.3. By studying the simulation it appears that it is the ROM values fetched from the previous iteration of Master ROM that are being written to the data_bus, and then sent back to the sequencer. The last write is the right value, but is not being used as it should be. This is a timing problem, which must be solved in order to realize this design. The output port data_bus can't get the right value in time before the old value is being sent into the sequencer. One possible solution is to implement the Master ROM as an FSM. In that way it will be easier to control the data flow and make sure that the right value is being used at the right time. The Master ROM is currently not implemented as an FSM, therefore no data flow control exists for Master ROM.

Data and keys

In this implementation real keys and authentic data has not been used. The main reason for this is that the focus has been on implementing the functionality of the decryption scheme. When the actual realization is finished, key generation has been implemented, either in software or hardware. There is no module that generates keys in this design. The test data has been a 160bit vector with random data ('0' and '1'), the register dec_input in the Decryption top module.

To temporary implement the private key, ROM blocks with random data has been implemented. In an actual realization of MQQ the ROM blocks must be replaced with RAM blocks. This will make it possible to change the keys in use, which again makes it possible to use the completed design as a general purpose MQQ hardware accelerator. A realization with fixed keys will not be useful in an eventual production of the design.

5.2 Conclusion

This report has covered the hardware implementation of MQQ. The functionality has been implemented and verified through simulation in Modelsim, but there exist design flaws which must be corrected before physical realization of MQQ on an FPGA can be implemented. The flaws are uncovered by simulation and synthesis of the design.

5.3 Contributors

There are people who have contributed so that the author has been able to implement the decryption scheme of MQQ. Especially PhD student Mohamed El-Hadedy at Q2S has contributed with lots of information about the original implementation. This information has been crucial in the design process.

Chapter 6

Abbreviations

RSA Rivest-Shamir-Adleman

DH Diffie-Hellman

AES Advanced Encryption Standard

DES Data Encryption Standard

3DES Triple Data Encryption Standard (DES)

MQQ Multivariate Quadratic Quasigroups

SSH Secure SHell

SSL Secure Socket Layer

MIT Massachusetts Institute of Technology

ECC Elliptic curve cryptography

FSM Finite State Machine

VHSIC Very-High-Speed Integrated Circuits

VHDL VHSIC Hardware Description Language

FPGA Field-programmable Gate Array

LUT Look-Up Table

ROM Read Only Memory

RAM Random Access Memory

TLS Transport Layer Security

Bibliography

[1] A Method for Obtaining Digital Signatures and Public-Key Cryptosystems
http://people.csail.mit.edu/rivest/Rsapaper.pdf

[2] A Public Key Block Cipher Based on Multivariate Quadratic Quasigroups
http://arxiv.org/abs/0808.0247

[3] High Performance Implementation of a Public Key Block Cipher - MQQ, for FPGA Platforms
http://eprint.iacr.org/2008/339

[4] Montgomery Reduction
http://en.wikipedia.org/w/index.php?title=Montgomery_reduction&oldid=255244240

[5] Diffie-Hellman key exchange
http://en.wikipedia.org/w/index.php?title=Diffie-Hellman_key_exchange&oldid=257292482

[6] Elliptic curve
http://en.wikipedia.org/w/index.php?title=Elliptic_curve&oldid=255908715

[7] Elliptic curve cryptography
http://en.wikipedia.org/w/index.php?title=Elliptic_curve_cryptography&oldid=255909821

Appendix A

Code listing

A.1 VHDL code

A.1.1 Decryption

Listing A.1: Decryption top level component

```
1  ----------------------------------------------------------------------------------
2  -- Company: NTNU
3  -- Engineer: Stig Fjellskaalnes
4  --
5  -- Create Date:    13:14:48 09/26/2008
6  -- Design Name:
7  -- Module Name:    decryption - rtl
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity decryption is
31   port (
32     clk    : in std_logic;
33     reset  : in std_logic;
34     en_in  : in std_logic;
35     inputs : in std_logic_vector (159 downto 0);
36     outputs : out std_logic_vector (159 downto 0);
37     en_out : out std_logic
38     );
39
40 end decryption;
41
42 architecture rtl of decryption is
43
44   type states is (IDLE,SEND_TO_PM_T,RECV_FROM_PM_T,SEND_TO_DOB,
45              RECV_FROM_DOB,SEND_TO_SEQ,RECV_FROM_SEQ,
46              SEND_TO_PM_S,RECV_FROM_PM_S,SEND);
47
48   component private_matrix_t
49     port (
50       clk     : in std_logic;
51       reset   : in std_logic;
52       en_in   : in std_logic;
53       input   : in std_logic_vector (4 downto 0);
```

```
54 |       cnt      : in integer range 0 to 31;
55 |       output     : out std_logic_vector (159 downto 0);
56 |       en_out     : out std_logic
57 |     );
58 |   end component;
59 |
60 |   component dobbertin_rom      -- defining the Dobbertin component
61 |                        -- in the decryption top level
62 |     port (
63 |       clk      : in std_logic;
64 |       reset      : in std_logic;
65 |       en_in      : in std_logic;
66 |       address_bus : in std_logic_vector (12 downto 0);
67 |       en_out     : out std_logic;
68 |       data_bus   : out std_logic_vector (12 downto 0)
69 |     );
70 |   end component;
71 |
72 |   component sequencer
73 |     port (
74 |         clk        : in  std_logic;
75 |         reset      : in  std_logic;
76 |         en_in      : in  std_logic;
77 |         input_seq  : in   std_logic_vector (159 downto 0);
78 |         en_out     : out std_logic;
79 |         output_seq   : out std_logic_vector (159 downto 0)
80 |     );
81 |   end component;
82 |
83 |   component private_matrix_s
84 |     port (
85 |       clk      : in std_logic;
86 |       reset      : in std_logic;
87 |       en_in      : in std_logic;
88 |       input      : in std_logic_vector (4 downto 0);
89 |       cnt      : in integer range 0 to 31;
90 |       output     : out std_logic_vector (159 downto 0);
91 |       en_out     : out std_logic
92 |     );
93 |   end component;
94 |
95 |   signal state,next_state : states;
96 |   signal dec_input       : std_logic_vector (159 downto 0);
97 |   signal dec_output      : std_logic_vector (159 downto 0);
98 |   signal result_pmt      : std_logic_vector (159 downto 0);
99 |   signal result_pms      : std_logic_vector (159 downto 0);
100|   signal result_seq      : std_logic_vector (159 downto 0);
101|   signal seq_in        : std_logic_vector (159 downto 0);
102|   signal dob_vector_in     : std_logic_vector (12 downto 0);
103|   signal dob_vector_out  : std_logic_vector (12 downto 0);
104|   signal count_t         : integer range 0 to 31;
105|   signal count_s         : integer range 0 to 31;
106|   signal shift_pmt       : std_logic_vector (4 downto 0);
107|   signal shift_pms       : std_logic_vector (4 downto 0);
108|   signal en_in_pm_t      : std_logic;
109|   signal en_in_pm_s      : std_logic;
110|   signal en_in_dob       : std_logic;
111|   signal en_in_seq       : std_logic;
112|   signal en_out_seq      : std_logic;
113|   signal en_out_pm_t     : std_logic;
114|   signal en_out_pm_s     : std_logic;
115|   signal en_out_dob      : std_logic;
116|
117| begin
118|   DR: DOBBERTIN_ROM        -- initializing the Dobbertin
119|                        -- ROM component in the decryption                    --  circuit
120|   port map(
121|     clk      =>  clk,
122|     reset      =>  reset,
123|     en_in      =>  en_in_dob,
124|     en_out     =>  en_out_dob,
125|     address_bus =>  dob_vector_in,
126|     data_bus     => dob_vector_out
127|     );
128|
129|   PM_T: PRIVATE_MATRIX_T       -- initializing the Private Matrix
130|                        -- (T) component in the decryption
131|                        --  circuit
132|   port map(
133|     clk      =>  clk,
134|     reset      =>  reset,
135|     en_in      =>  en_in_pm_t,
136|     input      =>  shift_pmt,
137|     cnt      => count_t,
```

```
    output     =>  result_pmt ,
    en_out     =>  en_out_pm_t
    );

  SEQ: SEQUENCER              -- initilizing the sequencer
                          -- component in the decryption
                          -- circuit

  port map(
    clk      => clk ,
    reset      =>  reset ,
    en_in      => en_in_seq ,
    input_seq => seq_in ,
    en_out     => en_out_seq ,
    output_seq  => result_seq
    );

  PM_S: PRIVATE_MATRIX_T       -- initializing the Private Matrix
                          -- (T) component in the decryption
                          --   circuit
  port map(
    clk      =>  clk ,
    reset      =>  reset ,
    en_in      =>  en_in_pm_s ,
    input      =>  shift_pms ,
    cnt      => count_s ,
    output     =>  result_pms ,
    en_out     =>  en_out_pm_s
    );

  running: process (clk , en_in)
  begin
    if (clk'event and clk = '1') then
      if (reset = '1') then
        state <= IDLE;

      elsif (en_in = '1') then
        state <= next_state;

      end if;
    end if;
  end process;

  output_dec: process (clk,en_in,state)
  begin
    if (clk'event and clk = '1') then
      if (reset = '1') then
        dob_vector_in <= (others => '0');
        dec_input <= (others => '0');
        dec_output <= (others => '0');
--          result_pmt <= (others => '0');
--          result_pms <= (others => '0');
--          result_seq <= (others => '0');
        outputs <= (others => '0');
        shift_pmt <= (others => '0');
        shift_pms <= (others => '0');
        en_out <= '0';
        en_in_dob <= '0';
        en_in_pm_t <= '0';
        en_in_pm_s <= '0';
--          en_out_dob <= '0';
--          en_out_seq <= '0';
        en_in_seq <= '0';
        count_t <= 0;
        count_s <= 0;

        seq_in <= (others => '0');


      elsif (en_in = '1') then
        case (state) is
          when IDLE =>

            dec_input <= "
                10011011010011110011001011001101100110110100111100110010110011011001101101001111001100101100110110011 01
                ";
            en_in_pm_t <= '0';
            if (count_t <= 31) then
              count_t <= 0;
            end if;


          when SEND_TO_PM_T =>
            en_in_pm_t <= '1';
```

```
            if (count_t = 31 and en_out_pm_t = '1') then
              null;
            else
              shift_pmt <= dec_input ((4+(count_t*5)) downto (0+(count_t*5)));
              if (en_out_pm_t = '1') then
                count_t <= count_t + 1;
              end if;
            end if;

          when RECV_FROM_PM_T =>

            case (result_pmt) is
              when ("
                  ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
                  ") =>
                null;
              when others =>
                en_in_dob <= '1';
            end case;

          when SEND_TO_DOB =>
            en_in_pm_t <= '0';
            en_in_dob <= en_out_pm_t;
            dob_vector_in(5 downto 0) <= result_pmt(5 downto 0);
            dob_vector_in(6) <= result_pmt(10);
            dob_vector_in(7) <= result_pmt(15);
            dob_vector_in(8) <= result_pmt(20);
            dob_vector_in(9) <= result_pmt(25);
            dob_vector_in(10) <= result_pmt(30);
            dob_vector_in(11) <= result_pmt(35);
            dob_vector_in(12) <= result_pmt(40);

          when RECV_FROM_DOB =>
            if (en_out_dob = '1') then
              seq_in <= result_pmt;
              case (dob_vector_out) is
                when "ZZZZZZZZZZZZZ" =>
                  null;

                when others =>
                  seq_in(5 downto 0) <= dob_vector_out(5 downto 0);
                  seq_in(10) <= dob_vector_out(6);
                  seq_in(15) <= dob_vector_out(7);
                  seq_in(20) <= dob_vector_out(8);
                  seq_in(25) <= dob_vector_out(9);
                  seq_in(30) <= dob_vector_out(10);
                  seq_in(35) <= dob_vector_out(11);
                  seq_in(40) <= dob_vector_out(12);
                  --en_out_dob <= '0';
              end case;
            end if;

          when SEND_TO_SEQ =>
            en_in_seq <= '1';

          when RECV_FROM_SEQ =>
            en_in_seq <= '0';
          when SEND_TO_PM_S =>
            en_in_pm_s <= '1';
            if (count_s = 31 and en_out_pm_s = '1') then
              null;
            else
              shift_pms <= result_seq((4+(count_s*5)) downto (0+(count_s*5)));
              if (en_out_pm_s = '1') then
                count_s <= count_s + 1;
              end if;
            end if;

          when RECV_FROM_PM_S =>
            en_in_pm_s <= '0';
            dec_output <= result_pms;

          when SEND =>
            outputs <= dec_output;
          when others =>
            null;
        end case;
      end if;
    end if;
  end process;

  -- calculates next state

  next_state_dec: process(state,count_t,en_out_pm_t,
```

```
                en_out_seq , count_s , en_out_pm_s )
  begin
    next_state <= state ;

    case ( state ) is
      --
      --
      when IDLE =>
        next_state <= SEND_TO_PM_T ;

      -- receives data from private matrix only when
      -- counter reaches 31, when the private matrix is done
      -- processing the 160bit data
      when SEND_TO_PM_T =>
        if ( count_t = 31 and en_out_pm_t = '1') then
            next_state <= RECV_FROM_PM_T ;
          else
            next_state <= SEND_TO_PM_T ;

        end if ;

      -- if the data is not valid , new data is being imported
      -- and the state machine moves back to IDLE.
      -- else , send data to dobbertin ROM
      when RECV_FROM_PM_T =>
        case result_pmt is
          when ("
              ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
              ") =>
            next_state <= IDLE ;
          when others =>
            next_state <= SEND_TO_DOB ;
        end case ;

      when SEND_TO_DOB =>
        next_state <= RECV_FROM_DOB ;

      when RECV_FROM_DOB =>
        next_state <= SEND_TO_SEQ ;

      when SEND_TO_SEQ =>
        case ( en_out_seq ) is
          when '0' =>
            next_state <= SEND_TO_SEQ ;
          when '1' =>
            next_state <= RECV_FROM_SEQ ;
          when others =>
            next_state <= IDLE ;
        end case ;

      when RECV_FROM_SEQ =>
        next_state <= SEND_TO_PM_S ;
      when SEND_TO_PM_S =>
        if ( count_s = 31 and en_out_pm_s = '1') then
          next_state <= RECV_FROM_PM_S ;
        else
          next_state <= SEND_TO_PM_S ;
        end if ;

      when RECV_FROM_PM_S =>
        case result_pmt is
          when ("
              ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ
              ") =>
            next_state <= IDLE ;
          when others =>
            next_state <= SEND ;
        end case ;

      when SEND =>
        next_state <= IDLE ;

      when others =>
        next_state <= IDLE ;

    end case ;
  end process ;


end rtl ;
```

A.1.2 Private Matrix T

Listing A.2: Private Matrix (T) component

```
1  ----------------------------------------------------------------------------------
2  -- Company: NTNU
3  -- Engineer: Stig Fjellskaalnes
4  --
5  -- Create Date:    13:15:38 09/26/2008
6  -- Design Name:
7  -- Module Name:    private_matrix - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity private_matrix_t is
31   port (
32     clk    : in std_logic;
33     reset    : in std_logic;
34     en_in    : in std_logic;
35     input    : in std_logic_vector (4 downto 0);
36     cnt    : in integer range 0 to 31;
37     output   : out std_logic_vector (159 downto 0);
38     en_out   : out std_logic
39     );
40
41 end private_matrix_t;
42
43 architecture Behavioral of private_matrix_t is
44
45   type priv_state is (IDLE,ANDOP,XORING,SYNC,PUSH_OUT);
46
47   -- the ROM components
48   type pm_rom is array ((32-1) downto 0) of std_logic_vector (4 downto 0);
49
50   constant rom_1 : pm_rom := (
51   "01111","10001","10001","11001","01110","11000","00111","01001",
52   "11111","01111","01000","01001","00110","00100","01011","11000",
53   "11011","01000","10000","00011","01000","10010","00101","01101",
54   "11011","00001","10111","11110","10110","01110","10110","11011"
55   );
56   |
57   |
58   |
59   |
60   |
61
62   constant rom_160 : pm_rom := (
63   "01111","10001","10001","11001","01110","11000","00111","01001",
64   "11111","01111","01000","01001","00110","00100","01011","11000",
65   "11011","01000","10000","00011","01000","10010","00101","01101",
66   "11011","00001","10111","11110","10110","01110","10110","11011"
67   );
68
69
70   signal and_rom_in_1       :   std_logic_vector (4 downto 0);
71   - - - -
72   signal and_rom_in_160       :   std_logic_vector (4 downto 0);
73
74   signal tmp_1       : std_logic;
75   - - - -
76   signal tmp_160       : std_logic;
77
78   signal from_rom_1        : std_logic_vector (4 downto 0);
79   - - - -
```

```
80  |   signal from_rom_160        : std_logic_vector (4 downto 0); signal register_x     : std_logic_vector
    |       (159 downto 0);
81  |   --  signal cnt          : integer range 0 to 31 := 0;
82  |   signal count_xor       : integer range 0 to 4 := 0;
83  |
84  |   signal xor_1       : std_logic;
85  |   signal sync_xor_1        : std_logic;
86  |   - - - -
87  |   signal xor_160       : std_logic;
88  |   signal sync_xor_160        : std_logic;
89  |
90  |   signal enable_out     : std_logic;
91  |   signal state,next_state : priv_state;
92  |
93  | begin
94  |   sync_run: process(clk,en_in,enable_out)
95  |   begin
96  |     if (clk'event and clk = '1') then
97  |       if (reset = '1') then
98  |         register_x <= (others => '0');
99  |         en_out <= '0';
100 |         state <= IDLE;
101 |         --cnt <= 0;
102 |         --count_xor <= 0;
103 |       elsif (en_in = '1') then
104 |         state <= next_state;
105 |         en_out <= enable_out;
106 |         if (cnt = 31 and enable_out = '1') then
107 |           register_x(0) <= sync_xor_1;
108 |           - - - -
109 |           register_x(159) <= sync_xor_160;
110 |         end if;
111 |       end if;
112 |     end if;
113 |   end process;
114 |
115 |   -- output_dec is made synchronous, so that all signals remember their
116 |   -- value at all times when not set again
117 |   output_dec: process (clk,state,cnt,count_xor,input,register_x)
118 |   begin
119 |     if (clk'event and clk = '1') then
120 |
121 |       if (reset = '1') then
122 |         output <= (others => '0');
123 |         enable_out <= '0';
124 |       elsif (en_in = '1') then
125 |
126 |         case (state) is
127 |
128 |           when IDLE =>
129 |
130 |             -- makes sure that en_out port is set to '0'
131 |             enable_out <= '0';
132 |             output <= (others => '0');
133 |             from_rom_1 <= rom_1(cnt);
134 |             - - -
135 |             from_rom_160 <= rom_160(cnt);
136 |
137 |           when ANDOP =>
138 |             -- makes the logical and operation between the input
139 |             -- value and the stored value inside the respective
140 |             -- ROM
141 |             and_rom_in_1 <= from_rom_1 and input;
142 |             - - -
143 |             and_rom_in_160 <= from_rom_160 and input;
144 |
145 |           when XORING =>
146 |             -- the actual bit-by-bit xor operation
147 |             tmp_1 <= and_rom_in_1(count_xor);
148 |             xor_1 <= and_rom_in_1(count_xor+1) xor tmp_1;
149 |             - - - - -
150 |             tmp_160 <= and_rom_in_160(count_xor);
151 |             xor_160 <= and_rom_in_160(count_xor+1) xor tmp_160;
152 |
153 |             if not (count_xor = 3) then
154 |               count_xor <= count_xor + 1;
155 |             end if;
156 |
157 |           when SYNC =>
158 |             -- sets the xor counter to 0
159 |             count_xor <= 0;
160 |             -- keep synchronization of the xor'ed bits
161 |             if (cnt = 0) then
162 |               sync_xor_1 <= xor_1;
```

```
163 |               - - -
164 |               sync_xor_160 <= xor_160;
165 |
166 |               enable_out <= '1';
167 |               -- does the last xor step between the previous
168 |               -- and the current 5 bit input vector
169 |             else
170 |               sync_xor_1 <= xor_1 xor sync_xor_1;
171 |               - - - -
172 |               sync_xor_160 <= xor_160 xor sync_xor_160;
173 |               enable_out <= '1';
174 |             end if;
175 |
176 |           when PUSH_OUT =>
177 |             -- pushes out the processed vector on output
178 |             enable_out <= '1';
179 |             output <= register_x;
180 |
181 |           when others =>
182 |
183 |             enable_out <= '0';
184 |             from_rom_1 <= (others => '0');
185 |             - - -
186 |             from_rom_160 <= (others => '0');
187 |
188 |         end case;
189 |       end if;
190 |     end if;
191 |   end process;
192 |
193 |   next_state_dec: process (state,from_rom_1,input,cnt,count_xor)
194 |   begin
195 |     next_state <= state;
196 |     case (state) is
197 |       when IDLE =>
198 |         -- makes sure that from_rom has the correct value
199 |         -- as to the running time given by signal cnt
200 |         if (from_rom_1 = rom_1(cnt)) then
201 |           next_state <= ANDOP;
202 |         end if;
203 |       when ANDOP =>
204 |         next_state <= XORING;
205 |       when XORING =>
206 |         if (count_xor < 3) then
207 |           next_state <= XORING;
208 |         else
209 |           next_state <= SYNC;
210 |         end if;
211 |       when SYNC =>
212 |         if (cnt = 31) then
213 |           next_state <= PUSH_OUT;
214 |         else
215 |           next_state <= IDLE;
216 |         end if;
217 |       when PUSH_OUT =>
218 |         next_state <= IDLE;
219 |       when others =>
220 |       next_state <= IDLE;
221 |     end case;
222 |   end process;
223 |
224 | end Behavioral;
```

A.1.3 Dobbertin ROM

Listing A.3: Dobbertin ROM component

```
1  ----------------------------------------------------------------------------------
2  -- Company: NTNU
3  -- Engineer: Stig Fjellskaalnes
4  --
5  -- Create Date:    13:16:20 09/26/2008
6  -- Design Name:
7  -- Module Name:    dobbertin_rom - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24 use IEEE.STD_LOGIC_TEXTIO.ALL;
25 --use IEEE.STD_ULOGIC_1164.ALL;
26 use std.textio.all;
27
28 ---- Uncomment the following library declaration if instantiating
29 ---- any Xilinx primitives in this code.
30 --library UNISIM;
31 --use UNISIM.VComponents.all;
32
33 entity dobbertin_rom is
34   port (
35     clk   :   in std_logic;
36     reset   :   in std_logic;
37     en_in   :   in std_logic;
38     address_bus : in std_logic_vector (12 downto 0);
39     data_bus  : out std_logic_vector (12 downto 0);
40     en_out    : out std_logic
41     );
42 end dobbertin_rom;
43
44 architecture Behavioral of dobbertin_rom is
45   type d_rom is array ((2**13)-1 downto 0) of std_logic_vector (12 downto 0);
46   constant dob_rom : d_rom :=(
47   "1011110011010","1100000101100","0111100011101","0111101001010",
48   - - - - - - - -
49   - - - - - - - -
50   - - - - - - - -
51   "1011000011101","1010001110100","0010100100100","0011101011000");
52   --signal sel:std_logic_vector (2 downto 0);
53 begin
54   run: process (clk, reset, en_in) is
55
56   begin
57
58     if(clk'event and clk = '1') then
59       if (reset = '1') then
60         data_bus<= (others =>'0');
61
62       elsif (en_in='1') then
63         case address_bus is
64           when "0000000000000" =>   data_bus <= dob_rom(0);
65           when "0000000000001" =>   data_bus <= dob_rom(1);
66             |
67             |
68             |
69             |
70           when "1111111111110" =>   data_bus <= dob_rom(8190);
71           when "1111111111111" =>   data_bus <= dob_rom(8191);
72
73           when others =>
74           data_bus<=(others=>'Z');
75         end case;
76       end if;
77       en_out<=en_in;
78     end if;
79   end process;
```

```
80
81
82
83 end Behavioral;
```

A.1.4 Sequencer

Listing A.4: Sequencer

```
1
2  -- Company: NTNU
3  -- Engineer: Stig Fjellskaalnes
4  --
5  -- Create Date:     11:26:53 10/22/2008
6  -- Design Name:
7  -- Module Name:     sequencer - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity sequencer is
31     port (
32       clk       : in  std_logic;
33       reset     : in  std_logic;
34       en_in     : in  std_logic;
35       input_seq : in   std_logic_vector (159 downto 0);
36       en_out    : out std_logic;
37       output_seq  : out std_logic_vector (159 downto 0)
38     );
39 end sequencer;
40
41 architecture Behavioral of sequencer is
42
43   component master_rom
44     port (
45       clk     : in std_logic;
46       reset     : in std_logic;
47       en_in     : in std_logic;
48       counter   : in integer range 0 to 31;
49       address_bus : in std_logic_vector (9 downto 0);
50       data_bus    : out std_logic_vector (4 downto 0);
51       en_out    : out std_logic
52     );
53
54   end component;
55
56   type reg_input is array ((2**5)-1 downto 0) of std_logic_vector (4 downto 0);
57   type states is (IDLE,MUX2_SEL,SEND_TO_MASTER,RECV_MR,MUX31_SEL,SYNC,PUSH);
58   signal reg_in,reg_out : reg_input;
59   signal sel            : std_logic;
60   signal counter_1      : std_logic_vector (4 downto 0);
61   signal counter_2      : integer range 0 to 31;
62   signal mux31_out      : std_logic_vector (4 downto 0);
63   signal mux2_out,sync_mr : std_logic_vector (4 downto 0);
64   signal to_master      : std_logic_vector (9 downto 0);
65   signal from_master    : std_logic_vector (4 downto 0);
66   signal en_out_mux     : std_logic;
67   signal en_out_mar     : std_logic;
68   signal state,next_state : states;
69
70
71 begin
72
73   MA_R: master_rom        -- initializing the master rom
74                       -- component in the sequencer
75                       --  circuit
76   port map(
77     clk     => clk,
78     reset     => reset,
79     counter   => counter_2,
```

```
    en_in     =>  en_in,
    en_out    =>  en_out_mar,
    address_bus =>  to_master,
    data_bus    => from_master
    );
  sec_run : process (clk, en_in, counter_1, counter_2)

  begin

  if (clk'event and clk = '1') then
    if (reset = '1') then
      state <= IDLE;


    elsif (en_in = '1') then
      state <= next_state;

    end if;
  end if;

  end process;

  output_dec: process(clk,en_in,state)
  begin

    if (clk'event and clk = '1') then

      if (reset = '1') then
        -- all signals is set to '0' when reset = '1'
        reg_in(0) <= (others => '0');
        reg_in(1) <= (others => '0');
        reg_in(2) <= (others => '0');
        reg_in(3) <= (others => '0');
        reg_in(4) <= (others => '0');
        reg_in(5) <= (others => '0');
        reg_in(6) <= (others => '0');
        reg_in(7) <= (others => '0');
        reg_in(8) <= (others => '0');
        reg_in(9) <= (others => '0');
        reg_in(10) <= (others => '0');
        reg_in(11) <= (others => '0');
        reg_in(12) <= (others => '0');
        reg_in(13) <= (others => '0');
        reg_in(14) <= (others => '0');
        reg_in(15) <= (others => '0');
        reg_in(16) <= (others => '0');
        reg_in(17) <= (others => '0');
        reg_in(18) <= (others => '0');
        reg_in(19) <= (others => '0');
        reg_in(20) <= (others => '0');
        reg_in(21) <= (others => '0');
        reg_in(22) <= (others => '0');
        reg_in(23) <= (others => '0');
        reg_in(24) <= (others => '0');
        reg_in(25) <= (others => '0');
        reg_in(26) <= (others => '0');
        reg_in(27) <= (others => '0');
        reg_in(28) <= (others => '0');
        reg_in(29) <= (others => '0');
        reg_in(30) <= (others => '0');
        reg_in(31) <= (others => '0');
        to_master <= (others => '0');
        en_out <= '0';
        sel <= '0';
        en_out_mux <= '0';
        mux31_out <= (others => '0');
        sync_mr <= (others => '0');
        mux2_out <= (others => '0');
        --from_master <= (others => '0');
        output_seq <= (others => '0');
        reg_out <= reg_in;
        counter_1 <= "00000";

      elsif (en_in = '1') then
        case (state) is
          -- input_seq input port is split up to an array of
          -- 5 bit vectors
          when IDLE =>
            reg_in(0) <= input_seq(4 downto 0);
            reg_in(1) <= input_seq(9 downto 5);
            reg_in(2) <= input_seq(14 downto 10);
            reg_in(3) <= input_seq(19 downto 15);
            reg_in(4) <= input_seq(24 downto 20);
            reg_in(5) <= input_seq(29 downto 25);
```

```
164 |             reg_in(6) <= input_seq(34 downto 30);
165 |             reg_in(7) <= input_seq(39 downto 35);
166 |             reg_in(8) <= input_seq(44 downto 40);
167 |             reg_in(9) <= input_seq(49 downto 45);
168 |             reg_in(10) <= input_seq(54 downto 50);
169 |             reg_in(11) <= input_seq(59 downto 55);
170 |             reg_in(12) <= input_seq(64 downto 60);
171 |             reg_in(13) <= input_seq(69 downto 65);
172 |             reg_in(14) <= input_seq(74 downto 70);
173 |             reg_in(15) <= input_seq(79 downto 75);
174 |             reg_in(16) <= input_seq(84 downto 80);
175 |             reg_in(17) <= input_seq(89 downto 85);
176 |             reg_in(18) <= input_seq(94 downto 90);
177 |             reg_in(19) <= input_seq(99 downto 95);
178 |             reg_in(20) <= input_seq(104 downto 100);
179 |             reg_in(21) <= input_seq(109 downto 105);
180 |             reg_in(22) <= input_seq(114 downto 110);
181 |             reg_in(23) <= input_seq(119 downto 115);
182 |             reg_in(24) <= input_seq(124 downto 120);
183 |             reg_in(25) <= input_seq(129 downto 125);
184 |             reg_in(26) <= input_seq(134 downto 130);
185 |             reg_in(27) <= input_seq(139 downto 135);
186 |             reg_in(28) <= input_seq(144 downto 140);
187 |             reg_in(29) <= input_seq(149 downto 145);
188 |             reg_in(30) <= input_seq(154 downto 150);
189 |             reg_in(31) <= input_seq(159 downto 155);
190 |             sel <= '0';
191 |             en_out <= '0';
192 |
193 |
194 |         when MUX2_SEL =>
195 |             counter_1 <= counter_1 + 1;
196 |             case (sel) is
197 |               when '1' => mux2_out <= sync_mr;
198 |               when '0' => mux2_out <= reg_in(0);
199 |               when others => mux2_out <= (others => 'Z');
200 |             end case;
201 |
202 |
203 |         -- generates the address bus, mux2_out will be the most sigifiant bits
204 |         when SEND_TO_MASTER =>
205 |             sel <= '1';
206 |             to_master (9 downto 5) <= mux2_out;
207 |             to_master (4 downto 0) <= mux31_out;
208 |
209 |         -- delay the data flow with 1 clk cycle
210 |         when RECV_MR =>
211 |             sync_mr <= from_master;
212 |
213 |         -- selects which value of the array reg_in to use
214 |         when MUX31_SEL =>
215 |
216 |             case (counter_1) is
217 |               when "00001" =>
218 |                 mux31_out <= reg_in(1);
219 |                 en_out_mux <= '1';
220 |               when "00010" =>
221 |                 mux31_out <= reg_in(2);
222 |                 en_out_mux <= '1';
223 |               when "00011" =>
224 |                 mux31_out <= reg_in(3);
225 |                 en_out_mux <= '1';
226 |               when "00100" =>
227 |                 mux31_out <= reg_in(4);
228 |                 en_out_mux <= '1';
229 |               when "00101" =>
230 |                 mux31_out <= reg_in(5);
231 |                 en_out_mux <= '1';
232 |               when "00110" =>
233 |                 mux31_out <= reg_in(6);
234 |                 en_out_mux <= '1';
235 |               when "00111" =>
236 |                 mux31_out <= reg_in(7);
237 |                 en_out_mux <= '1';
238 |               when "01000" =>
239 |                 mux31_out <= reg_in(8);
240 |                 en_out_mux <= '1';
241 |               when "01001" =>
242 |                 mux31_out <= reg_in(9);
243 |                 en_out_mux <= '1';
244 |               when "01010" =>
245 |                 mux31_out <= reg_in(10);
246 |                 en_out_mux <= '1';
247 |               when "01011" =>
```

```
248 |                 mux31_out <= reg_in(11);                                |
249 |                 en_out_mux <= '1';                                      |
250 |               when "01100" =>                                           |
251 |                 mux31_out <= reg_in(12);                                |
252 |                 en_out_mux <= '1';                                      |
253 |               when "01101" =>                                           |
254 |                 mux31_out <= reg_in(13);                                |
255 |                 en_out_mux <= '1';                                      |
256 |               when "01110" =>                                           |
257 |                 mux31_out <= reg_in(14);                                |
258 |                 en_out_mux <= '1';                                      |
259 |               when "01111" =>                                           |
260 |                 mux31_out <= reg_in(15);                                |
261 |                 en_out_mux <= '1';                                      |
262 |               when "10000" =>                                           |
263 |                 mux31_out <= reg_in(16);                                |
264 |                 en_out_mux <= '1';                                      |
265 |               when "10001" =>                                           |
266 |                 mux31_out <= reg_in(17);                                |
267 |                 en_out_mux <= '1';                                      |
268 |               when "10010" =>                                           |
269 |                 mux31_out <= reg_in(18);                                |
270 |                 en_out_mux <= '1';                                      |
271 |               when "10011" =>                                           |
272 |                 mux31_out <= reg_in(19);                                |
273 |                 en_out_mux <= '1';                                      |
274 |               when "10100" =>                                           |
275 |                 mux31_out <= reg_in(20);                                |
276 |                 en_out_mux <= '1';                                      |
277 |               when "10101" =>                                           |
278 |                 mux31_out <= reg_in(21);                                |
279 |                 en_out_mux <= '1';                                      |
280 |               when "10110" =>                                           |
281 |                 mux31_out <= reg_in(22);                                |
282 |                 en_out_mux <= '1';                                      |
283 |               when "10111" =>                                           |
284 |                 mux31_out <= reg_in(23);                                |
285 |                 en_out_mux <= '1';                                      |
286 |               when "11000" =>                                           |
287 |                 mux31_out <= reg_in(24);                                |
288 |                 en_out_mux <= '1';                                      |
289 |               when "11001" =>                                           |
290 |                 mux31_out <= reg_in(25);                                |
291 |                 en_out_mux <= '1';                                      |
292 |               when "11010" =>                                           |
293 |                 mux31_out <= reg_in(26);                                |
294 |                 en_out_mux <= '1';                                      |
295 |               when "11011" =>                                           |
296 |                 mux31_out <= reg_in(27);                                |
297 |                 en_out_mux <= '1';                                      |
298 |               when "11100" =>                                           |
299 |                 mux31_out <= reg_in(28);                                |
300 |                 en_out_mux <= '1';                                      |
301 |               when "11101" =>                                           |
302 |                 mux31_out <= reg_in(29);                                |
303 |                 en_out_mux <= '1';                                      |
304 |               when "11110" =>                                           |
305 |                 mux31_out <= reg_in(30);                                |
306 |                 en_out_mux <= '1';                                      |
307 |               when "11111" =>                                           |
308 |                 mux31_out <= reg_in(31);                                |
309 |                 en_out_mux <= '1';                                      |
310 |                                                                         |
311 |               when others =>                                            |
312 |                 en_out_mux <= '0';                                      |
313 |                 mux31_out <= (others => 'Z');                           |
314 |             end case;                                                   |
315 |                                                                         |
316 |             -- writes to the output vector array                        |
317 |             when SYNC =>                                                |
318 |               reg_out(counter_2) <= mux2_out;                           |
319 |               if (counter_2 < 31) then                                  |
320 |                 counter_2 <= counter_2 + 1;                             |
321 |               end if;                                                   |
322 |                                                                         |
323 |             -- push the data in the output vector on output port output_seq
324 |                                                                         |
325 |             when PUSH =>                                                |
326 |               output_seq(4 downto 0) <= reg_out(0);                     |
327 |               output_seq(9 downto 5) <= reg_out(1);                     |
328 |               output_seq(14 downto 10) <= reg_out(2);                   |
329 |               output_seq(19 downto 15) <= reg_out(3);                   |
330 |               output_seq(24 downto 20) <= reg_out(4);                   |
331 |               output_seq(29 downto 25) <= reg_out(5);                   |
```

```
            output_seq(34 downto 30) <= reg_out(6);
            output_seq(39 downto 35) <= reg_out(7);
            output_seq(44 downto 40) <= reg_out(8);
            output_seq(49 downto 45) <= reg_out(9);
            output_seq(54 downto 50) <= reg_out(10);
            output_seq(59 downto 55) <= reg_out(11);
            output_seq(64 downto 60) <= reg_out(12);
            output_seq(69 downto 65) <= reg_out(13);
            output_seq(74 downto 70) <= reg_out(14);
            output_seq(79 downto 75) <= reg_out(15);
            output_seq(84 downto 80) <= reg_out(16);
            output_seq(89 downto 85) <= reg_out(17);
            output_seq(94 downto 90) <= reg_out(18);
            output_seq(99 downto 95) <= reg_out(19);
            output_seq(104 downto 100) <= reg_out(20);
            output_seq(109 downto 105) <= reg_out(21);
            output_seq(114 downto 110) <= reg_out(22);
            output_seq(119 downto 115) <= reg_out(23);
            output_seq(124 downto 120) <= reg_out(24);
            output_seq(129 downto 125) <= reg_out(25);
            output_seq(134 downto 130) <= reg_out(26);
            output_seq(139 downto 135) <= reg_out(27);
            output_seq(144 downto 140) <= reg_out(28);
            output_seq(149 downto 145) <= reg_out(29);
            output_seq(154 downto 150) <= reg_out(30);
            output_seq(159 downto 155) <= reg_out(31);
            en_out <= '1';
        end case;
      end if;
    end if;

  end process;

  -- controlling process for the FSM
  next_state_dec: process(state,counter_1,counter_2,en_out_mar)
  begin
    next_state <= state;
    case (state) is
      when IDLE =>
        case (counter_1) is
          when "00000" =>
            next_state <= MUX2_SEL;

          when others =>
            next_state <= MUX31_SEL;
          end case;
      when MUX2_SEL =>
        next_state <= SYNC;
      when SEND_TO_MASTER =>
        next_state <= RECV_MR;
      when RECV_MR =>
        next_state <= MUX2_SEL;
      when MUX31_SEL =>
        next_state <= SEND_TO_MASTER;

      when SYNC =>
        if (counter_2 = 31) then
          next_state <= PUSH;
        else
          next_state <= MUX31_SEL;
        end if;

      when PUSH =>
        next_state <= IDLE;
    end case;
  end process;


end Behavioral;
```

A.1.5 Master ROM

Listing A.5: Sequencer

```
1  ----------------------------------------------------------------------------------
2  -- Company: NTNU
3  -- Engineer: Stig Fjellskaalnes
4  --
5  -- Create Date:    12:16:51 10/22/2008
6  -- Design Name:
7  -- Module Name:    master_rom - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22 use IEEE.STD_LOGIC_ARITH.ALL;
23 use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25 ---- Uncomment the following library declaration if instantiating
26 ---- any Xilinx primitives in this code.
27 --library UNISIM;
28 --use UNISIM.VComponents.all;
29
30 entity master_rom is
31   port (
32     clk     : in std_logic;
33     reset     : in std_logic;
34     counter   : in integer range 0 to 31;
35     en_in     : in std_logic;
36     address_bus : in std_logic_vector (9 downto 0);
37     data_bus    : out std_logic_vector (4 downto 0);
38     en_out    : out std_logic
39     );
40
41 end master_rom;
42
43 architecture Behavioral of master_rom is
44   type ma_rom is array((2**10)-1 downto 0) of std_logic_vector (4 downto 0);
45   type control_rom is array ((2**5)-1 downto 0) of std_logic_vector (2 downto 0);
46   --signal counter        : integer range 0 to 31;
47   signal from_ctrl      : std_logic_vector (2 downto 0);
48   signal from_rom_1     : std_logic_vector (4 downto 0);
49   signal from_rom_2     : std_logic_vector (4 downto 0);
50   signal from_rom_3     : std_logic_vector (4 downto 0);
51   signal from_rom_4     : std_logic_vector (4 downto 0);
52   signal from_rom_5     : std_logic_vector (4 downto 0);
53   signal from_rom_6     : std_logic_vector (4 downto 0);
54   signal from_rom_7     : std_logic_vector (4 downto 0);
55   signal from_rom_8     : std_logic_vector (4 downto 0);
56   signal enable_out     : std_logic;
57
58
59   --signal en_out_ctrl  : std_logic;
60
61   constant rom_1 : ma_rom := (
62   "11011","01000","00111","01101","10110","11010","01110","00011",
63   "10011","11111","01001","01001","00110","01001","01101","01001",
64     - - - -
65     - - - -
66     - - - -
67   "00110","11111","01111","11110","00100","01010","01110","10100",
68   "00010","10101","00111","01010","00110","11111","01001","11110"
69   );
70     |
71     |
72     |
73     |
74
75   constant rom_8 : ma_rom := (
76   "10110","11010","01110","00011","10011","11111","01001","01001",
77   "00110","01001","01101","01001","10001","11000","00011","11010",
78     - - - -
79     - - - -
```

```
80 |   "00100","01010","01110","10100","00010","10101","00111","01010",
81 |   "00110","11111","01001","11110","00001","00001","00001","00001"
82 |   );
83 |
84 |
85 |   constant cnt_rom : control_rom := (
86 |   "101","111","001","101","011","000","001","011",
87 |   "000","111","100","011","101","011","110","100",
88 |   "101","010","110","001","110","101","011","101",
89 |   "010","100","101","000","001","101","000","010");
90 |
91 | begin
92 |
93 |   --  main process
94 |   --  the selector is controlled by the input port counter, which
95 |   --  is located in the sequencer
96 |
97 |   sync_run: process (clk,en_in,enable_out)
98 |   begin
99 |     if (clk'event and clk = '1') then
100|       if (reset = '1') then
101|         from_ctrl <= "000";
102| --        counter <= 0;
103|         en_out <= '0';
104|
105|       elsif (en_in = '1') then
106|         from_ctrl <= cnt_rom(counter);
107|         if (counter = 31) then
108|           en_out <= '0';
109|           --counter <= 0;
110|         else
111|           --counter <= counter + 1;
112|           en_out <= '1';
113|         end if;
114|       end if;
115|     end if;
116|   end process;
117|
118|
119|   --  sets the from_rom signals dependent on the address
120|
121|   address_calc: process (clk,en_in,reset,address_bus)
122|   begin
123|     if (clk'event and clk = '1') then
124|       if (reset = '1') then
125|         from_rom_1 <= (others => '0');
126|         from_rom_2 <= (others => '0');
127|         from_rom_3 <= (others => '0');
128|         from_rom_4 <= (others => '0');
129|         from_rom_5 <= (others => '0');
130|         from_rom_6 <= (others => '0');
131|         from_rom_7 <= (others => '0');
132|         from_rom_8 <= (others => '0');
133|
134|       elsif (en_in = '1') then
135|         case (address_bus) is
136|           when "0000000000" =>
137|             from_rom_1 <= rom_1(0);
138|             from_rom_2 <= rom_2(0);
139|             from_rom_3 <= rom_3(0);
140|             from_rom_4 <= rom_4(0);
141|             from_rom_5 <= rom_5(0);
142|             from_rom_6 <= rom_6(0);
143|             from_rom_7 <= rom_7(0);
144|             from_rom_8 <= rom_8(0);
145|
146|             |
147|             |
148|             |
149|
150|           when "1111111111" =>
151|             from_rom_1 <= rom_1(1023);
152|             from_rom_2 <= rom_2(1023);
153|             from_rom_3 <= rom_3(1023);
154|             from_rom_4 <= rom_4(1023);
155|             from_rom_5 <= rom_5(1023);
156|             from_rom_6 <= rom_6(1023);
157|             from_rom_7 <= rom_7(1023);
158|             from_rom_8 <= rom_8(1023);
159|
160|           when others =>
161|             from_rom_1 <= (others => 'Z');
162|             from_rom_2 <= (others => 'Z');
163|             from_rom_3 <= (others => 'Z');
```

```
164 |             from_rom_4 <= (others => 'Z');
165 |             from_rom_5 <= (others => 'Z');
166 |             from_rom_6 <= (others => 'Z');
167 |             from_rom_7 <= (others => 'Z');
168 |             from_rom_8 <= (others => 'Z');
169 |
170 |         end case;
171 |
172 |       end if;
173 |     end if;
174 |   end process;
175 |
176 |   -- determines which ROM is to be used as output data
177 |   push_data: process(clk,en_in,from_ctrl)
178 |   begin
179 |     if (clk'event and clk = '1') then
180 |       if (reset = '1') then
181 |         data_bus <= "00000";
182 |         enable_out <= '0';
183 |       elsif (en_in = '1') then
184 |         case (from_ctrl) is
185 |           when "000" =>
186 |             data_bus <= from_rom_1;
187 |             --enable_out <= '1';
188 |
189 |           when "001" =>
190 |             data_bus <= from_rom_2;
191 |             --  enable_out <= '1';
192 |
193 |           when "010" =>
194 |             data_bus <= from_rom_3;
195 |             --  enable_out <= '1';
196 |
197 |           when "011" =>
198 |             data_bus <= from_rom_4;
199 |             --  enable_out <= '1';
200 |
201 |           when "100" =>
202 |             data_bus <= from_rom_5;
203 |             --  enable_out <= '1';
204 |
205 |           when "101" =>
206 |             data_bus <= from_rom_6;
207 |             --  enable_out <= '1';
208 |
209 |           when "110" =>
210 |             data_bus <= from_rom_7;
211 |             --  enable_out <= '1';
212 |
213 |           when "111" =>
214 |             data_bus <= from_rom_8;
215 |             --  enable_out <= '1';
216 |
217 |           when others =>
218 |           --              data_bus <= (others => 'Z');
219 |           --              enable_out <= '0';
220 |         end case;
221 |
222 |       end if;
223 |       enable_out <= '1';
224 |     end if;
225 |
226 |   end process;
227 | end Behavioral;
```

A.2 C/C++ code

To easily generate similar codelines fast, some programs written in C or C++ has been written to quickly generate codelines that are similar in nature.

Listing A.6: Dobbertin ROM file generator (C)

```c
#include <stdio.h>
#include <stdlib.h>



/*
Defines standard values needed for the generator program
 */
int i = 0;
int** rom_array;
// ROM size has 2^13 elements
int colval = 8192;
// Each element has length 13 bits
int rowval = 13;
// File pointer
FILE *rom;


/*
Dynamically allocates memory for the rom_array two-
dimensional array (matrix) using malloc()
*/
void init_rom_array(){
  rom_array = (int**) malloc(sizeof(int*)*colval);
  if (rom_array == NULL){
    printf ("Not_allocated\n");
  }
  for (i = 0; i < colval; i++){
    rom_array[i] = malloc(sizeof(int)*rowval);
    if (rom_array[i] == NULL){
      printf ("Not_allocated\n");
    }
  }
}

/*
Sets random 0's and 1's in the rom_array matrix
 */
int** randomize(int** rom_array){
  int row;
  int col;
  for (col = 0; col < colval; col++){
    for (row = 0; row < rowval; row++){
      rom_array[col][row] = rand() % 2;
    }
  }
  return rom_array;
}

/*
Writes the result to a dob.rom file, which can be imported to the
VHDL source code as a ROM block. The dob.rom is properly formatted.
 */
void write_file(int** rom_array){
  int row;
  int col;
  int i = 0;
  rom = fopen("dob.rom", "w");
  for ( col = 0; col < colval; col++){
    fprintf(rom,"\"");
    for (row = 0; row < rowval; row++){
      fprintf(rom, "%i", rom_array[col][row]);
      if( row == (rowval - 1)){
  fprintf(rom, "\",");
      }
    }
    if(i == 3){
      fprintf(rom, "\n");
      //sets i to -1. this is because i will get value 1 the next round
      //if i is set to 0
      i = -1;
    }
    i++;
  }
  fclose(rom);
```

```
}

int main(void){

  init_rom_array();
  randomize(rom_array);
  write_file(rom_array);

  return 0;

}
```

Listing A.7: Dobbertin ROM code generator (C++)

```
#include <bitset>
#include <iostream>
#include <cstdlib>
#include <fstream>
using namespace std;


void bit(){

  //fp is the file pointer
  ofstream fp;

  // opens the file for writing
  fp.open ("dob_code.txt");

  for (int i = 0; i < 8192; i++){

    // converts the current value of i to a binary value
    // with length 13
    const bitset<13> mask(i);

    /*
      Prints out a valid VHDL syntax for the content in a
      multiplexer with 2^13 inputs
     */
    fp << "when " << "\"" << mask  << "\"" << " " << "=> ";
    fp << " " << "data_bus <= dob_rom(" << i <<");" << endl;
  }

  // closes the output file
  fp.close();
}

int main(void){

  bit();
  return 0;

}
```

Listing A.8: Master ROM Generator (C++)

```
#include <iostream>
#include <bitset>
#include <cstdlib>
#include <fstream>

using namespace std;

int length, size;

int* rom1 = new int[length];
int* rom2 = new int[length];
int* rom3 = new int[length];
int* rom4 = new int[length];
int* rom5 = new int[length];
int* rom6 = new int[length];
int* rom7 = new int[length];
int* rom8 = new int[length];


void init(){
  for (int i = 0; i < length; i++){
    rom1[i] = rand()%size;
    rom2[i] = rom1[i] << 1;
    rom3[i] = rom2[i] | rom1[i];
    rom4[i] = rom2[i] & rom3[i];
```

```
    rom5[i] = rom4[i] >> 1;
    rom6[i] = rom4[i] xor rom5[i];
    rom7[i] = rom5[i] and rom6[i];
    rom8[i] = rom7[i] or rom6[i];
    //cout << rom4[i] << endl;
  }
}

void bit(){
  ofstream fp;
  int line = 0;
  fp.open ("master.rom");
  for (int i = 0; i < length; i++){
    int temp1 = rom7[i];

    const bitset<5> mask(temp1);
    fp << "\"" << mask << "\"" << ",";

    if (line == 7){
      fp << endl;
      line = -1;
    }
    line++;
  }
  fp.close();
}


int main(void){
  cin >> size;
  cin >> length;
  init();
  bit();
  return 0;
}
```

Listing A.9: Master ROM Code Generator (C++)

```
#include <bitset>
#include <iostream>
#include <cstdlib>
#include <fstream>
using namespace std;


void bit(){

  //fp is the file pointer
  ofstream fp;

  // opens the file for writing
  fp.open ("master.code");

  for (int i = 0; i < 1024; i++){

    // converts the current value of i to a binary value
    // with length 10
    const bitset<10> mask(i);

    /*
      Prints out a valid VHDL syntax for the content in a
      multiplexer with 2^10 inputs
     */
    fp << "when " << "\"" << mask  << "\"" << " " << "=> " << endl;
    fp << "from_rom_1 <= rom_1(" << i <<");" << endl;
    fp << "from_rom_2 <= rom_2(" << i <<");" << endl;
    fp << "from_rom_3 <= rom_3(" << i <<");" << endl;
    fp << "from_rom_4 <= rom_4(" << i <<");" << endl;
    fp << "from_rom_5 <= rom_5(" << i <<");" << endl;
    fp << "from_rom_6 <= rom_6(" << i <<");" << endl;
    fp << "from_rom_7 <= rom_7(" << i <<");" << endl;
    fp << "from_rom_8 <= rom_8(" << i <<");" << endl;
    fp << endl;

  }

  // closes the output file
  fp.close();
}

int main(void){

  bit();
```

```
46 |   return 0;
47 |
48 | }
```

Listing A.10: Private Matrix code generator (C++)

```
1  | #include <bitset>
2  | #include <iostream>
3  | #include <cstdlib>
4  | #include <fstream>
5  | using namespace std;
6  |
7  |
8  | void signal_and(){
9  |
10 |   //fp is the file pointer
11 |   ofstream fp;
12 |
13 |   // opens the file for writing
14 |   fp.open ("priv_code_and.txt");
15 |
16 |   // writes VHDL code for signal declaration to file
17 |   for (int i = 1; i < 161; i++){
18 |     fp << "signal_and_rom_in_" << i << "_______:___std_logic_vector_(4_downto_0);" << endl;
19 |   }
20 |
21 |   // closes the output file
22 |   fp.close();
23 | }
24 | void signal_from(){
25 |   ofstream fp2;
26 |   fp2.open ("sig_priv_code_from.txt");
27 |
28 |   for (int i = 1; i < 161; i++){
29 |     fp2 << "signal_from_rom_" << i << "________:_std_logic_vector_(4_downto_0);" << endl;
30 |
31 |   }
32 |   fp2.close();
33 | }
34 |
35 | void signal_tmp(){
36 |   ofstream fp3;
37 |   fp3.open ("sig_priv_code_tmp.txt");
38 |
39 |   for (int i = 1; i < 161; i++){
40 |     fp3 << "signal_tmp_" << i << "_______:_std_logic;" << endl;
41 |   }
42 |   fp3.close();
43 | }
44 |
45 | void signal_xor(){
46 |   ofstream fp4;
47 |   fp4.open("sig_priv_code_xor.txt");
48 |   for (int i = 1; i < 161; i++){
49 |     fp4 << "signal_xor_" << i << "_______:_std_logic;" << endl;
50 |     fp4 << "signal_sync_xor_" << i << "_______:_std_logic;" << endl;
51 |   }
52 |   fp4.close();
53 | }
54 |
55 | void from(){
56 |   ofstream fp5;
57 |   fp5.open("priv_code_from.txt");
58 |   for (int i = 1; i < 161; i++){
59 |     fp5 << "from_rom_" << i << "_<=_" << "rom_" << i << "(cnt);" << endl;
60 |   }
61 |   fp5.close();
62 | }
63 |
64 | void andop(){
65 |   ofstream fp6;
66 |   fp6.open("priv_code_andop.txt");
67 |   for (int i = 1; i < 161; i++){
68 |     fp6 << "and_rom_in_" << i << "_<=_"<< "from_rom_" << i << "_and_" << "input;" << endl;
69 |   }
70 |   fp6.close();
71 | }
72 |
73 | void sync(){
74 |   ofstream fp7;
75 |   fp7.open("priv_code_sync0.txt");
76 |   for (int i = 1; i < 161; i++){
77 |     fp7 << "sync_xor_" << i << "_<=_" << "xor_" << i << ";" << endl;
```

```
78 |   }
79 |   fp7.close();
80 | }
81 |
82 | void sync_else(){
83 |   ofstream fp8;
84 |   fp8.open("priv_code_sync_else.txt");
85 |   for (int i = 1; i < 161; i++){
86 |     fp8 << "sync_xor_" << i << " <= "<< "xor_" << i << "_xor_" << "sync_xor_" << i << ";" << endl;
87 |   }
88 |   fp8.close();
89 | }
90 |
91 | void reg(){
92 |   ofstream fp9;
93 |   fp9.open("priv_code_reg.txt");
94 |   for (int i = 1; i < 161; i++){
95 |     fp9 << "register_x(" << i-1 << ") <= " << "sync_xor_" << i << ";" << endl;
96 |   }
97 |   fp9.close();
98 | }
99 |
100 | void other(){
101 |   ofstream fp10;
102 |   fp10.open("priv_code_other.txt");
103 |   for (int i = 1; i < 161; i++){
104 |     fp10 << "from_rom_" << i << " <= (others => '0');" << endl;
105 |   }
106 |   fp10.close();
107 | }
108 |
109 | void xoring(){
110 |   ofstream fp11;
111 |   fp11.open("priv_code_xor.txt");
112 |   for (int i = 1; i < 161; i++){
113 |     fp11 << "tmp_" << i << " <= " << "and_rom_in_" << i << "(count_xor);" << endl;
114 |     fp11 << "xor_" << i << " <= " << "and_rom_in_" << i << "(count_xor+1) xor_" << "tmp_" << i << ";"
          << endl;
115 |   }
116 |   fp11.close();
117 | }
118 |
119 | int main(void){
120 |
121 |   signal_and();
122 |   signal_from();
123 |   signal_tmp();
124 |   signal_xor();
125 |   from();
126 |   andop();
127 |   sync();
128 |   sync_else();
129 |   reg();
130 |   other();
131 |   xoring();
132 |   return 0;
133 |
134 | }
```

Appendix B

Synthesis report

Listing B.1: Synthesis report (xc5vfx70-1ff1136)

```
Release 10.1.03 - xst K.39 (lin)
Copyright (c) 1995-2008 Xilinx, Inc.  All rights reserved.
-->
Parameter TMPDIR set to /home/stig/Documents/mqq/xst/projnav.tmp


Total REAL time to Xst completion: 0.00 secs
Total CPU time to Xst completion: 0.05 secs

-->
Parameter xsthdpdir set to /home/stig/Documents/mqq/xst


Total REAL time to Xst completion: 0.00 secs
Total CPU time to Xst completion: 0.05 secs

-->
Reading design: decryption.prj

TABLE OF CONTENTS
  1) Synthesis Options Summary
  2) HDL Compilation
  3) Design Hierarchy Analysis
  4) HDL Analysis
  5) HDL Synthesis
     5.1) HDL Synthesis Report
  6) Advanced HDL Synthesis
     6.1) Advanced HDL Synthesis Report
  7) Low Level Synthesis
  8) Partition Report
  9) Final Report
     9.1) Device utilization summary
     9.2) Partition Resource Summary
     9.3) TIMING REPORT


=========================================================================
*                      Synthesis Options Summary                        *
=========================================================================
---- Source Parameters
Input File Name                    : "decryption.prj"
Input Format                       : mixed
Ignore Synthesis Constraint File   : NO

---- Target Parameters
Output File Name                   : "decryption"
Output Format                      : NGC
Target Device                      : xc5vfx70t-1-ff1136

---- Source Options
Top Module Name                    : decryption
Automatic FSM Extraction           : YES
FSM Encoding Algorithm             : Auto
Safe Implementation                : No
FSM Style                          : lut
RAM Extraction                     : Yes
RAM Style                          : Auto
ROM Extraction                     : Yes
Mux Style                          : Auto
Decoder Extraction                 : YES
Priority Encoder Extraction        : YES
Shift Register Extraction          : YES
Logical Shifter Extraction         : YES
XOR Collapsing                     : YES
ROM Style                          : Auto
Mux Extraction                     : YES
Resource Sharing                   : YES
```

```
Asynchronous To Synchronous          : NO
Use DSP Block                        : auto
Automatic Register Balancing         : No

---- Target Options
LUT Combining                        : off
Reduce Control Sets                  : off
Add IO Buffers                       : YES
Global Maximum Fanout                : 100000
Add Generic Clock Buffer(BUFG)       : 32
Register Duplication                 : YES
Slice Packing                        : YES
Optimize Instantiated Primitives     : NO
Use Clock Enable                     : Auto
Use Synchronous Set                  : Auto
Use Synchronous Reset                : Auto
Pack IO Registers into IOBs          : auto
Equivalent register Removal          : YES

---- General Options
Optimization Goal                    : Speed
Optimization Effort                  : 1
Power Reduction                      : NO
Library Search Order                 : decryption.lso
Keep Hierarchy                       : NO
Netlist Hierarchy                    : as_optimized
RTL Output                           : Yes
Global Optimization                  : AllClockNets
Read Cores                           : YES
Write Timing Constraints             : NO
Cross Clock Analysis                 : NO
Hierarchy Separator                  : /
Bus Delimiter                        : <>
Case Specifier                       : maintain
Slice Utilization Ratio              : 100
BRAM Utilization Ratio               : 100
DSP48 Utilization Ratio              : 100
Verilog 2001                         : YES
Auto BRAM Packing                    : NO
Slice Utilization Ratio Delta        : 5

=========================================================================


=========================================================================
*                            HDL Compilation                            *
=========================================================================
Compiling vhdl file "/home/stig/Documents/mqq/VHDL/master_rom.vhd" in Library work.
Architecture behavioral of Entity master_rom is up to date.
Compiling vhdl file "/home/stig/Documents/mqq/VHDL/dobbertin_rom.vhd" in Library work.
Architecture behavioral of Entity dobbertin_rom is up to date.
Compiling vhdl file "/home/stig/Documents/mqq/VHDL/private_matrix_t.vhd" in Library work.
Architecture behavioral of Entity private_matrix_t is up to date.
Compiling vhdl file "/home/stig/Documents/mqq/VHDL/sequencer.vhd" in Library work.
Architecture behavioral of Entity sequencer is up to date.
Compiling vhdl file "/home/stig/Documents/mqq/VHDL/decryption.vhd" in Library work.
Architecture rtl of Entity decryption is up to date.

=========================================================================
*                        Design Hierarchy Analysis                      *
=========================================================================
Analyzing hierarchy for entity <decryption> in library <work> (architecture <rtl>).

Analyzing hierarchy for entity <dobbertin_rom> in library <work> (architecture <behavioral>).

Analyzing hierarchy for entity <private_matrix_t> in library <work> (architecture <behavioral>).

Analyzing hierarchy for entity <sequencer> in library <work> (architecture <behavioral>).

Analyzing hierarchy for entity <master_rom> in library <work> (architecture <behavioral>).


=========================================================================
*                            HDL Analysis                               *
=========================================================================
Analyzing Entity <decryption> in library <work> (Architecture <rtl>).
INFO:Xst:1561 - "/home/stig/Documents/mqq/VHDL/decryption.vhd" line 345: Mux is complete : default of
    case is discarded
INFO:Xst:2679 - Register <en_out> in unit <decryption> has a constant value of 0 during circuit
    operation. The register is replaced by logic.
Entity <decryption> analyzed. Unit <decryption> generated.

Analyzing Entity <dobbertin_rom> in library <work> (Architecture <behavioral>).
Entity <dobbertin_rom> analyzed. Unit <dobbertin_rom> generated.

Analyzing Entity <private_matrix_t> in library <work> (Architecture <behavioral>).
Entity <private_matrix_t> analyzed. Unit <private_matrix_t> generated.

Analyzing Entity <sequencer> in library <work> (Architecture <behavioral>).
INFO:Xst:1561 - "/home/stig/Documents/mqq/VHDL/sequencer.vhd" line 194: Mux is complete : default of
    case is discarded
Entity <sequencer> analyzed. Unit <sequencer> generated.

Analyzing Entity <master_rom> in library <work> (Architecture <behavioral>).
```

```
INFO:Xst:2679 - Register <enable_out> in unit <master_rom> has a constant value of 1 during circuit
     operation. The register is replaced by logic.
Entity <master_rom> analyzed. Unit <master_rom> generated.


=========================================================================
*                            HDL Synthesis                              *
=========================================================================

Performing bidirectional port resolution...

Synthesizing Unit <dobbertin_rom>.
    Related source file is "/home/stig/Documents/mqq/VHDL/dobbertin_rom.vhd".
    Found 8192x13-bit ROM for signal <data_bus$mux0000> created at line 2106.
    Found 1-bit register for signal <en_out>.
    Found 13-bit register for signal <data_bus>.
    Summary:
  inferred   1 ROM(s).
  inferred  14 D-type flip-flop(s).
Unit <dobbertin_rom> synthesized.


Synthesizing Unit <private_matrix_t>.
    Related source file is "/home/stig/Documents/mqq/VHDL/private_matrix_t.vhd".
    Found 32x5-bit ROM for signal <from_rom_1$rom0000> created at line 2175.
    Found 32x5-bit ROM for signal <from_rom_2$rom0000> created at line 2176.
    Found 32x5-bit ROM for signal <from_rom_3$rom0000> created at line 2177.
    Found 32x5-bit ROM for signal <from_rom_4$rom0000> created at line 2178.
    Found 32x5-bit ROM for signal <from_rom_5$rom0000> created at line 2179.
    Found 32x5-bit ROM for signal <from_rom_6$rom0000> created at line 2180.
    Found 32x5-bit ROM for signal <from_rom_7$rom0000> created at line 2181.
    Found 32x5-bit ROM for signal <from_rom_8$rom0000> created at line 2182.
    Found 32x5-bit ROM for signal <from_rom_9$rom0000> created at line 2183.
    Found 32x5-bit ROM for signal <from_rom_10$rom0000> created at line 2184.
    Found 32x5-bit ROM for signal <from_rom_11$rom0000> created at line 2185.
    Found 32x5-bit ROM for signal <from_rom_12$rom0000> created at line 2186.
    Found 32x5-bit ROM for signal <from_rom_13$rom0000> created at line 2187.
    Found 32x5-bit ROM for signal <from_rom_14$rom0000> created at line 2188.
    Found 32x5-bit ROM for signal <from_rom_20$rom0000> created at line 2194.
    Found 32x5-bit ROM for signal <from_rom_15$rom0000> created at line 2189.
    Found 32x5-bit ROM for signal <from_rom_16$rom0000> created at line 2190.
    Found 32x5-bit ROM for signal <from_rom_21$rom0000> created at line 2195.
    Found 32x5-bit ROM for signal <from_rom_22$rom0000> created at line 2196.
    Found 32x5-bit ROM for signal <from_rom_17$rom0000> created at line 2191.
    Found 32x5-bit ROM for signal <from_rom_23$rom0000> created at line 2197.
    Found 32x5-bit ROM for signal <from_rom_18$rom0000> created at line 2192.
    Found 32x5-bit ROM for signal <from_rom_24$rom0000> created at line 2198.
    Found 32x5-bit ROM for signal <from_rom_19$rom0000> created at line 2193.
    Found 32x5-bit ROM for signal <from_rom_25$rom0000> created at line 2199.
    Found 32x5-bit ROM for signal <from_rom_30$rom0000> created at line 2204.
    Found 32x5-bit ROM for signal <from_rom_31$rom0000> created at line 2205.
    Found 32x5-bit ROM for signal <from_rom_26$rom0000> created at line 2200.
    Found 32x5-bit ROM for signal <from_rom_32$rom0000> created at line 2206.
    Found 32x5-bit ROM for signal <from_rom_27$rom0000> created at line 2201.
    Found 32x5-bit ROM for signal <from_rom_28$rom0000> created at line 2202.
    Found 32x5-bit ROM for signal <from_rom_33$rom0000> created at line 2207.
    Found 32x5-bit ROM for signal <from_rom_29$rom0000> created at line 2203.
    Found 32x5-bit ROM for signal <from_rom_34$rom0000> created at line 2208.
    Found 32x5-bit ROM for signal <from_rom_35$rom0000> created at line 2209.
    Found 32x5-bit ROM for signal <from_rom_40$rom0000> created at line 2214.
    Found 32x5-bit ROM for signal <from_rom_41$rom0000> created at line 2215.
    Found 32x5-bit ROM for signal <from_rom_36$rom0000> created at line 2210.
    Found 32x5-bit ROM for signal <from_rom_42$rom0000> created at line 2216.
    Found 32x5-bit ROM for signal <from_rom_37$rom0000> created at line 2211.
    Found 32x5-bit ROM for signal <from_rom_38$rom0000> created at line 2212.
    Found 32x5-bit ROM for signal <from_rom_43$rom0000> created at line 2217.
    Found 32x5-bit ROM for signal <from_rom_39$rom0000> created at line 2213.
    Found 32x5-bit ROM for signal <from_rom_44$rom0000> created at line 2218.
    Found 32x5-bit ROM for signal <from_rom_45$rom0000> created at line 2219.
    Found 32x5-bit ROM for signal <from_rom_50$rom0000> created at line 2224.
    Found 32x5-bit ROM for signal <from_rom_51$rom0000> created at line 2225.
    Found 32x5-bit ROM for signal <from_rom_46$rom0000> created at line 2220.
    Found 32x5-bit ROM for signal <from_rom_52$rom0000> created at line 2226.
    Found 32x5-bit ROM for signal <from_rom_47$rom0000> created at line 2221.
    Found 32x5-bit ROM for signal <from_rom_48$rom0000> created at line 2222.
    Found 32x5-bit ROM for signal <from_rom_53$rom0000> created at line 2227.
    Found 32x5-bit ROM for signal <from_rom_49$rom0000> created at line 2223.
    Found 32x5-bit ROM for signal <from_rom_54$rom0000> created at line 2228.
    Found 32x5-bit ROM for signal <from_rom_55$rom0000> created at line 2229.
    Found 32x5-bit ROM for signal <from_rom_60$rom0000> created at line 2234.
    Found 32x5-bit ROM for signal <from_rom_61$rom0000> created at line 2235.
    Found 32x5-bit ROM for signal <from_rom_56$rom0000> created at line 2230.
    Found 32x5-bit ROM for signal <from_rom_62$rom0000> created at line 2236.
    Found 32x5-bit ROM for signal <from_rom_57$rom0000> created at line 2231.
    Found 32x5-bit ROM for signal <from_rom_58$rom0000> created at line 2232.
    Found 32x5-bit ROM for signal <from_rom_63$rom0000> created at line 2237.
    Found 32x5-bit ROM for signal <from_rom_59$rom0000> created at line 2233.
    Found 32x5-bit ROM for signal <from_rom_64$rom0000> created at line 2238.
    Found 32x5-bit ROM for signal <from_rom_65$rom0000> created at line 2239.
    Found 32x5-bit ROM for signal <from_rom_70$rom0000> created at line 2244.
    Found 32x5-bit ROM for signal <from_rom_66$rom0000> created at line 2240.
    Found 32x5-bit ROM for signal <from_rom_71$rom0000> created at line 2245.
    Found 32x5-bit ROM for signal <from_rom_72$rom0000> created at line 2246.
    Found 32x5-bit ROM for signal <from_rom_67$rom0000> created at line 2241.
    Found 32x5-bit ROM for signal <from_rom_68$rom0000> created at line 2242.
```

```
Found 32x5-bit ROM for signal <from_rom_73$rom0000> created at line 2247.
Found 32x5-bit ROM for signal <from_rom_69$rom0000> created at line 2243.
Found 32x5-bit ROM for signal <from_rom_74$rom0000> created at line 2248.
Found 32x5-bit ROM for signal <from_rom_75$rom0000> created at line 2249.
Found 32x5-bit ROM for signal <from_rom_80$rom0000> created at line 2254.
Found 32x5-bit ROM for signal <from_rom_76$rom0000> created at line 2250.
Found 32x5-bit ROM for signal <from_rom_81$rom0000> created at line 2255.
Found 32x5-bit ROM for signal <from_rom_82$rom0000> created at line 2256.
Found 32x5-bit ROM for signal <from_rom_77$rom0000> created at line 2251.
Found 32x5-bit ROM for signal <from_rom_78$rom0000> created at line 2252.
Found 32x5-bit ROM for signal <from_rom_83$rom0000> created at line 2257.
Found 32x5-bit ROM for signal <from_rom_79$rom0000> created at line 2253.
Found 32x5-bit ROM for signal <from_rom_84$rom0000> created at line 2258.
Found 32x5-bit ROM for signal <from_rom_85$rom0000> created at line 2259.
Found 32x5-bit ROM for signal <from_rom_90$rom0000> created at line 2264.
Found 32x5-bit ROM for signal <from_rom_86$rom0000> created at line 2260.
Found 32x5-bit ROM for signal <from_rom_91$rom0000> created at line 2265.
Found 32x5-bit ROM for signal <from_rom_92$rom0000> created at line 2266.
Found 32x5-bit ROM for signal <from_rom_87$rom0000> created at line 2261.
Found 32x5-bit ROM for signal <from_rom_88$rom0000> created at line 2262.
Found 32x5-bit ROM for signal <from_rom_93$rom0000> created at line 2267.
Found 32x5-bit ROM for signal <from_rom_89$rom0000> created at line 2263.
Found 32x5-bit ROM for signal <from_rom_94$rom0000> created at line 2268.
Found 32x5-bit ROM for signal <from_rom_95$rom0000> created at line 2269.
Found 32x5-bit ROM for signal <from_rom_96$rom0000> created at line 2270.
Found 32x5-bit ROM for signal <from_rom_97$rom0000> created at line 2271.
Found 32x5-bit ROM for signal <from_rom_98$rom0000> created at line 2272.
Found 32x5-bit ROM for signal <from_rom_99$rom0000> created at line 2273.
Found 32x5-bit ROM for signal <from_rom_100$rom0000> created at line 2274.
Found 32x5-bit ROM for signal <next_state$rom0000> created at line 3342.
Found 32x5-bit ROM for signal <next_state$rom0001> created at line 3342.
Found 32x5-bit ROM for signal <from_rom_101$rom0000> created at line 2275.
Found 32x5-bit ROM for signal <from_rom_102$rom0000> created at line 2276.
Found 32x5-bit ROM for signal <from_rom_103$rom0000> created at line 2277.
Found 32x5-bit ROM for signal <from_rom_104$rom0000> created at line 2278.
Found 32x5-bit ROM for signal <from_rom_110$rom0000> created at line 2284.
Found 32x5-bit ROM for signal <from_rom_105$rom0000> created at line 2279.
Found 32x5-bit ROM for signal <from_rom_106$rom0000> created at line 2280.
Found 32x5-bit ROM for signal <from_rom_111$rom0000> created at line 2285.
Found 32x5-bit ROM for signal <from_rom_107$rom0000> created at line 2281.
Found 32x5-bit ROM for signal <from_rom_112$rom0000> created at line 2286.
Found 32x5-bit ROM for signal <from_rom_113$rom0000> created at line 2287.
Found 32x5-bit ROM for signal <from_rom_108$rom0000> created at line 2282.
Found 32x5-bit ROM for signal <from_rom_109$rom0000> created at line 2283.
Found 32x5-bit ROM for signal <from_rom_114$rom0000> created at line 2288.
Found 32x5-bit ROM for signal <from_rom_120$rom0000> created at line 2294.
Found 32x5-bit ROM for signal <from_rom_115$rom0000> created at line 2289.
Found 32x5-bit ROM for signal <from_rom_116$rom0000> created at line 2290.
Found 32x5-bit ROM for signal <from_rom_121$rom0000> created at line 2295.
Found 32x5-bit ROM for signal <from_rom_117$rom0000> created at line 2291.
Found 32x5-bit ROM for signal <from_rom_122$rom0000> created at line 2296.
Found 32x5-bit ROM for signal <from_rom_123$rom0000> created at line 2297.
Found 32x5-bit ROM for signal <from_rom_118$rom0000> created at line 2292.
Found 32x5-bit ROM for signal <from_rom_119$rom0000> created at line 2293.
Found 32x5-bit ROM for signal <from_rom_124$rom0000> created at line 2298.
Found 32x5-bit ROM for signal <from_rom_130$rom0000> created at line 2304.
Found 32x5-bit ROM for signal <from_rom_125$rom0000> created at line 2299.
Found 32x5-bit ROM for signal <from_rom_126$rom0000> created at line 2300.
Found 32x5-bit ROM for signal <from_rom_131$rom0000> created at line 2305.
Found 32x5-bit ROM for signal <from_rom_127$rom0000> created at line 2301.
Found 32x5-bit ROM for signal <from_rom_132$rom0000> created at line 2306.
Found 32x5-bit ROM for signal <from_rom_133$rom0000> created at line 2307.
Found 32x5-bit ROM for signal <from_rom_128$rom0000> created at line 2302.
Found 32x5-bit ROM for signal <from_rom_129$rom0000> created at line 2303.
Found 32x5-bit ROM for signal <from_rom_134$rom0000> created at line 2308.
Found 32x5-bit ROM for signal <from_rom_140$rom0000> created at line 2314.
Found 32x5-bit ROM for signal <from_rom_135$rom0000> created at line 2309.
Found 32x5-bit ROM for signal <from_rom_136$rom0000> created at line 2310.
Found 32x5-bit ROM for signal <from_rom_141$rom0000> created at line 2315.
Found 32x5-bit ROM for signal <from_rom_137$rom0000> created at line 2311.
Found 32x5-bit ROM for signal <from_rom_142$rom0000> created at line 2316.
Found 32x5-bit ROM for signal <from_rom_143$rom0000> created at line 2317.
Found 32x5-bit ROM for signal <from_rom_138$rom0000> created at line 2312.
Found 32x5-bit ROM for signal <from_rom_139$rom0000> created at line 2313.
Found 32x5-bit ROM for signal <from_rom_144$rom0000> created at line 2318.
Found 32x5-bit ROM for signal <from_rom_150$rom0000> created at line 2324.
Found 32x5-bit ROM for signal <from_rom_145$rom0000> created at line 2319.
Found 32x5-bit ROM for signal <from_rom_146$rom0000> created at line 2320.
Found 32x5-bit ROM for signal <from_rom_151$rom0000> created at line 2325.
Found 32x5-bit ROM for signal <from_rom_147$rom0000> created at line 2321.
Found 32x5-bit ROM for signal <from_rom_152$rom0000> created at line 2326.
Found 32x5-bit ROM for signal <from_rom_153$rom0000> created at line 2327.
Found 32x5-bit ROM for signal <from_rom_148$rom0000> created at line 2322.
Found 32x5-bit ROM for signal <from_rom_149$rom0000> created at line 2323.
Found 32x5-bit ROM for signal <from_rom_154$rom0000> created at line 2328.
Found 32x5-bit ROM for signal <from_rom_160$rom0000> created at line 2334.
Found 32x5-bit ROM for signal <from_rom_155$rom0000> created at line 2329.
Found 32x5-bit ROM for signal <from_rom_156$rom0000> created at line 2330.
Found 32x5-bit ROM for signal <from_rom_157$rom0000> created at line 2331.
Found 32x5-bit ROM for signal <from_rom_158$rom0000> created at line 2332.
Found 32x5-bit ROM for signal <from_rom_159$rom0000> created at line 2333.
Found finite state machine <FSM_0> for signal <state>.
-----------------------------------------------------------------------
| States             | 5                                              |
| Transitions        | 8                                              |
```

```
| Inputs              | 4                                                  |
| Outputs             | 8                                                  |
| Clock               | clk (rising_edge)                                  |
| Clock enable        | en_in (positive)                                   |
| Reset               | reset (positive)                                   |
| Reset type          | synchronous                                        |
| Reset State         | idle                                               |
| Power Up State      | idle                                               |
| Encoding            | automatic                                          |
| Implementation      | LUT                                                |
-----------------------------------------------------------------------
Found 1-bit register for signal <en_out>.
Found 160-bit register for signal <output>.
Found 5-bit register for signal <and_rom_in_1>.
Found 5-bit register for signal <and_rom_in_10>.
Found 5-bit register for signal <and_rom_in_100>.
Found 5-bit register for signal <and_rom_in_101>.
Found 5-bit register for signal <and_rom_in_102>.
Found 5-bit register for signal <and_rom_in_103>.
Found 5-bit register for signal <and_rom_in_104>.
Found 5-bit register for signal <and_rom_in_105>.
Found 5-bit register for signal <and_rom_in_106>.
Found 5-bit register for signal <and_rom_in_107>.
Found 5-bit register for signal <and_rom_in_108>.
Found 5-bit register for signal <and_rom_in_109>.
Found 5-bit register for signal <and_rom_in_11>.
Found 5-bit register for signal <and_rom_in_110>.
Found 5-bit register for signal <and_rom_in_111>.
Found 5-bit register for signal <and_rom_in_112>.
Found 5-bit register for signal <and_rom_in_113>.
Found 5-bit register for signal <and_rom_in_114>.
Found 5-bit register for signal <and_rom_in_115>.
Found 5-bit register for signal <and_rom_in_116>.
Found 5-bit register for signal <and_rom_in_117>.
Found 5-bit register for signal <and_rom_in_118>.
Found 5-bit register for signal <and_rom_in_119>.
Found 5-bit register for signal <and_rom_in_12>.
Found 5-bit register for signal <and_rom_in_120>.
Found 5-bit register for signal <and_rom_in_121>.
Found 5-bit register for signal <and_rom_in_122>.
Found 5-bit register for signal <and_rom_in_123>.
Found 5-bit register for signal <and_rom_in_124>.
Found 5-bit register for signal <and_rom_in_125>.
Found 5-bit register for signal <and_rom_in_126>.
Found 5-bit register for signal <and_rom_in_127>.
Found 5-bit register for signal <and_rom_in_128>.
Found 5-bit register for signal <and_rom_in_129>.
Found 5-bit register for signal <and_rom_in_13>.
Found 5-bit register for signal <and_rom_in_130>.
Found 5-bit register for signal <and_rom_in_131>.
Found 5-bit register for signal <and_rom_in_132>.
Found 5-bit register for signal <and_rom_in_133>.
Found 5-bit register for signal <and_rom_in_134>.
Found 5-bit register for signal <and_rom_in_135>.
Found 5-bit register for signal <and_rom_in_136>.
Found 5-bit register for signal <and_rom_in_137>.
Found 5-bit register for signal <and_rom_in_138>.
Found 5-bit register for signal <and_rom_in_139>.
Found 5-bit register for signal <and_rom_in_14>.
Found 5-bit register for signal <and_rom_in_140>.
Found 5-bit register for signal <and_rom_in_141>.
Found 5-bit register for signal <and_rom_in_142>.
Found 5-bit register for signal <and_rom_in_143>.
Found 5-bit register for signal <and_rom_in_144>.
Found 5-bit register for signal <and_rom_in_145>.
Found 5-bit register for signal <and_rom_in_146>.
Found 5-bit register for signal <and_rom_in_147>.
Found 5-bit register for signal <and_rom_in_148>.
Found 5-bit register for signal <and_rom_in_149>.
Found 5-bit register for signal <and_rom_in_15>.
Found 5-bit register for signal <and_rom_in_150>.
Found 5-bit register for signal <and_rom_in_151>.
Found 5-bit register for signal <and_rom_in_152>.
Found 5-bit register for signal <and_rom_in_153>.
Found 5-bit register for signal <and_rom_in_154>.
Found 5-bit register for signal <and_rom_in_155>.
Found 5-bit register for signal <and_rom_in_156>.
Found 5-bit register for signal <and_rom_in_157>.
Found 5-bit register for signal <and_rom_in_158>.
Found 5-bit register for signal <and_rom_in_159>.
Found 5-bit register for signal <and_rom_in_16>.
Found 5-bit register for signal <and_rom_in_160>.
Found 5-bit register for signal <and_rom_in_17>.
Found 5-bit register for signal <and_rom_in_18>.
Found 5-bit register for signal <and_rom_in_19>.
Found 5-bit register for signal <and_rom_in_2>.
Found 5-bit register for signal <and_rom_in_20>.
Found 5-bit register for signal <and_rom_in_21>.
Found 5-bit register for signal <and_rom_in_22>.
Found 5-bit register for signal <and_rom_in_23>.
Found 5-bit register for signal <and_rom_in_24>.
Found 5-bit register for signal <and_rom_in_25>.
Found 5-bit register for signal <and_rom_in_26>.
Found 5-bit register for signal <and_rom_in_27>.
Found 5-bit register for signal <and_rom_in_28>.
```

```
    Found 5-bit register for signal <and_rom_in_29>.
    Found 5-bit register for signal <and_rom_in_3>.
    Found 5-bit register for signal <and_rom_in_30>.
    Found 5-bit register for signal <and_rom_in_31>.
    Found 5-bit register for signal <and_rom_in_32>.
    Found 5-bit register for signal <and_rom_in_33>.
    Found 5-bit register for signal <and_rom_in_34>.
    Found 5-bit register for signal <and_rom_in_35>.
    Found 5-bit register for signal <and_rom_in_36>.
    Found 5-bit register for signal <and_rom_in_37>.
    Found 5-bit register for signal <and_rom_in_38>.
    Found 5-bit register for signal <and_rom_in_39>.
    Found 5-bit register for signal <and_rom_in_4>.
    Found 5-bit register for signal <and_rom_in_40>.
    Found 5-bit register for signal <and_rom_in_41>.
    Found 5-bit register for signal <and_rom_in_42>.
    Found 5-bit register for signal <and_rom_in_43>.
    Found 5-bit register for signal <and_rom_in_44>.
    Found 5-bit register for signal <and_rom_in_45>.
    Found 5-bit register for signal <and_rom_in_46>.
    Found 5-bit register for signal <and_rom_in_47>.
    Found 5-bit register for signal <and_rom_in_48>.
    Found 5-bit register for signal <and_rom_in_49>.
    Found 5-bit register for signal <and_rom_in_5>.
    Found 5-bit register for signal <and_rom_in_50>.
    Found 5-bit register for signal <and_rom_in_51>.
    Found 5-bit register for signal <and_rom_in_52>.
    Found 5-bit register for signal <and_rom_in_53>.
    Found 5-bit register for signal <and_rom_in_54>.
    Found 5-bit register for signal <and_rom_in_55>.
    Found 5-bit register for signal <and_rom_in_56>.
    Found 5-bit register for signal <and_rom_in_57>.
    Found 5-bit register for signal <and_rom_in_58>.
    Found 5-bit register for signal <and_rom_in_59>.
    Found 5-bit register for signal <and_rom_in_6>.
    Found 5-bit register for signal <and_rom_in_60>.
    Found 5-bit register for signal <and_rom_in_61>.
    Found 5-bit register for signal <and_rom_in_62>.
    Found 5-bit register for signal <and_rom_in_63>.
    Found 5-bit register for signal <and_rom_in_64>.
    Found 5-bit register for signal <and_rom_in_65>.
    Found 5-bit register for signal <and_rom_in_66>.
    Found 5-bit register for signal <and_rom_in_67>.
    Found 5-bit register for signal <and_rom_in_68>.
    Found 5-bit register for signal <and_rom_in_69>.
    Found 5-bit register for signal <and_rom_in_7>.
    Found 5-bit register for signal <and_rom_in_70>.
    Found 5-bit register for signal <and_rom_in_71>.
    Found 5-bit register for signal <and_rom_in_72>.
    Found 5-bit register for signal <and_rom_in_73>.
    Found 5-bit register for signal <and_rom_in_74>.
    Found 5-bit register for signal <and_rom_in_75>.
    Found 5-bit register for signal <and_rom_in_76>.
    Found 5-bit register for signal <and_rom_in_77>.
    Found 5-bit register for signal <and_rom_in_78>.
    Found 5-bit register for signal <and_rom_in_79>.
    Found 5-bit register for signal <and_rom_in_8>.
    Found 5-bit register for signal <and_rom_in_80>.
    Found 5-bit register for signal <and_rom_in_81>.
    Found 5-bit register for signal <and_rom_in_82>.
    Found 5-bit register for signal <and_rom_in_83>.
    Found 5-bit register for signal <and_rom_in_84>.
    Found 5-bit register for signal <and_rom_in_85>.
    Found 5-bit register for signal <and_rom_in_86>.
    Found 5-bit register for signal <and_rom_in_87>.
    Found 5-bit register for signal <and_rom_in_88>.
    Found 5-bit register for signal <and_rom_in_89>.
    Found 5-bit register for signal <and_rom_in_9>.
    Found 5-bit register for signal <and_rom_in_90>.
    Found 5-bit register for signal <and_rom_in_91>.
    Found 5-bit register for signal <and_rom_in_92>.
    Found 5-bit register for signal <and_rom_in_93>.
    Found 5-bit register for signal <and_rom_in_94>.
    Found 5-bit register for signal <and_rom_in_95>.
    Found 5-bit register for signal <and_rom_in_96>.
    Found 5-bit register for signal <and_rom_in_97>.
    Found 5-bit register for signal <and_rom_in_98>.
    Found 5-bit register for signal <and_rom_in_99>.
    Found 3-bit register for signal <count_xor>.
    Found 3-bit adder for signal <count_xor$addsub0000> created at line 2825.
    Found 1-bit register for signal <enable_out>.
    Found 5-bit register for signal <from_rom_1>.
    Found 5-bit register for signal <from_rom_10>.
    Found 5-bit register for signal <from_rom_100>.
    Found 5-bit register for signal <from_rom_101>.
    Found 5-bit register for signal <from_rom_102>.
    Found 5-bit register for signal <from_rom_103>.
    Found 5-bit register for signal <from_rom_104>.
    Found 5-bit register for signal <from_rom_105>.
    Found 5-bit register for signal <from_rom_106>.
    Found 5-bit register for signal <from_rom_107>.
    Found 5-bit register for signal <from_rom_108>.
    Found 5-bit register for signal <from_rom_109>.
    Found 5-bit register for signal <from_rom_11>.
    Found 5-bit register for signal <from_rom_110>.
```

```
Found 5-bit register for signal <from_rom_111>.
Found 5-bit register for signal <from_rom_112>.
Found 5-bit register for signal <from_rom_113>.
Found 5-bit register for signal <from_rom_114>.
Found 5-bit register for signal <from_rom_115>.
Found 5-bit register for signal <from_rom_116>.
Found 5-bit register for signal <from_rom_117>.
Found 5-bit register for signal <from_rom_118>.
Found 5-bit register for signal <from_rom_119>.
Found 5-bit register for signal <from_rom_12>.
Found 5-bit register for signal <from_rom_120>.
Found 5-bit register for signal <from_rom_121>.
Found 5-bit register for signal <from_rom_122>.
Found 5-bit register for signal <from_rom_123>.
Found 5-bit register for signal <from_rom_124>.
Found 5-bit register for signal <from_rom_125>.
Found 5-bit register for signal <from_rom_126>.
Found 5-bit register for signal <from_rom_127>.
Found 5-bit register for signal <from_rom_128>.
Found 5-bit register for signal <from_rom_129>.
Found 5-bit register for signal <from_rom_13>.
Found 5-bit register for signal <from_rom_130>.
Found 5-bit register for signal <from_rom_131>.
Found 5-bit register for signal <from_rom_132>.
Found 5-bit register for signal <from_rom_133>.
Found 5-bit register for signal <from_rom_134>.
Found 5-bit register for signal <from_rom_135>.
Found 5-bit register for signal <from_rom_136>.
Found 5-bit register for signal <from_rom_137>.
Found 5-bit register for signal <from_rom_138>.
Found 5-bit register for signal <from_rom_139>.
Found 5-bit register for signal <from_rom_14>.
Found 5-bit register for signal <from_rom_140>.
Found 5-bit register for signal <from_rom_141>.
Found 5-bit register for signal <from_rom_142>.
Found 5-bit register for signal <from_rom_143>.
Found 5-bit register for signal <from_rom_144>.
Found 5-bit register for signal <from_rom_145>.
Found 5-bit register for signal <from_rom_146>.
Found 5-bit register for signal <from_rom_147>.
Found 5-bit register for signal <from_rom_148>.
Found 5-bit register for signal <from_rom_149>.
Found 5-bit register for signal <from_rom_15>.
Found 5-bit register for signal <from_rom_150>.
Found 5-bit register for signal <from_rom_151>.
Found 5-bit register for signal <from_rom_152>.
Found 5-bit register for signal <from_rom_153>.
Found 5-bit register for signal <from_rom_154>.
Found 5-bit register for signal <from_rom_155>.
Found 5-bit register for signal <from_rom_156>.
Found 5-bit register for signal <from_rom_157>.
Found 5-bit register for signal <from_rom_158>.
Found 5-bit register for signal <from_rom_159>.
Found 5-bit register for signal <from_rom_16>.
Found 5-bit register for signal <from_rom_160>.
Found 5-bit register for signal <from_rom_17>.
Found 5-bit register for signal <from_rom_18>.
Found 5-bit register for signal <from_rom_19>.
Found 5-bit register for signal <from_rom_2>.
Found 5-bit register for signal <from_rom_20>.
Found 5-bit register for signal <from_rom_21>.
Found 5-bit register for signal <from_rom_22>.
Found 5-bit register for signal <from_rom_23>.
Found 5-bit register for signal <from_rom_24>.
Found 5-bit register for signal <from_rom_25>.
Found 5-bit register for signal <from_rom_26>.
Found 5-bit register for signal <from_rom_27>.
Found 5-bit register for signal <from_rom_28>.
Found 5-bit register for signal <from_rom_29>.
Found 5-bit register for signal <from_rom_3>.
Found 5-bit register for signal <from_rom_30>.
Found 5-bit register for signal <from_rom_31>.
Found 5-bit register for signal <from_rom_32>.
Found 5-bit register for signal <from_rom_33>.
Found 5-bit register for signal <from_rom_34>.
Found 5-bit register for signal <from_rom_35>.
Found 5-bit register for signal <from_rom_36>.
Found 5-bit register for signal <from_rom_37>.
Found 5-bit register for signal <from_rom_38>.
Found 5-bit register for signal <from_rom_39>.
Found 5-bit register for signal <from_rom_4>.
Found 5-bit register for signal <from_rom_40>.
Found 5-bit register for signal <from_rom_41>.
Found 5-bit register for signal <from_rom_42>.
Found 5-bit register for signal <from_rom_43>.
Found 5-bit register for signal <from_rom_44>.
Found 5-bit register for signal <from_rom_45>.
Found 5-bit register for signal <from_rom_46>.
Found 5-bit register for signal <from_rom_47>.
Found 5-bit register for signal <from_rom_48>.
Found 5-bit register for signal <from_rom_49>.
Found 5-bit register for signal <from_rom_5>.
Found 5-bit register for signal <from_rom_50>.
Found 5-bit register for signal <from_rom_51>.
Found 5-bit register for signal <from_rom_52>.
```

```
Found 5-bit register for signal <from_rom_53>.
Found 5-bit register for signal <from_rom_54>.
Found 5-bit register for signal <from_rom_55>.
Found 5-bit register for signal <from_rom_56>.
Found 5-bit register for signal <from_rom_57>.
Found 5-bit register for signal <from_rom_58>.
Found 5-bit register for signal <from_rom_59>.
Found 5-bit register for signal <from_rom_6>.
Found 5-bit register for signal <from_rom_60>.
Found 5-bit register for signal <from_rom_61>.
Found 5-bit register for signal <from_rom_62>.
Found 5-bit register for signal <from_rom_63>.
Found 5-bit register for signal <from_rom_64>.
Found 5-bit register for signal <from_rom_65>.
Found 5-bit register for signal <from_rom_66>.
Found 5-bit register for signal <from_rom_67>.
Found 5-bit register for signal <from_rom_68>.
Found 5-bit register for signal <from_rom_69>.
Found 5-bit register for signal <from_rom_7>.
Found 5-bit register for signal <from_rom_70>.
Found 5-bit register for signal <from_rom_71>.
Found 5-bit register for signal <from_rom_72>.
Found 5-bit register for signal <from_rom_73>.
Found 5-bit register for signal <from_rom_74>.
Found 5-bit register for signal <from_rom_75>.
Found 5-bit register for signal <from_rom_76>.
Found 5-bit register for signal <from_rom_77>.
Found 5-bit register for signal <from_rom_78>.
Found 5-bit register for signal <from_rom_79>.
Found 5-bit register for signal <from_rom_8>.
Found 5-bit register for signal <from_rom_80>.
Found 5-bit register for signal <from_rom_81>.
Found 5-bit register for signal <from_rom_82>.
Found 5-bit register for signal <from_rom_83>.
Found 5-bit register for signal <from_rom_84>.
Found 5-bit register for signal <from_rom_85>.
Found 5-bit register for signal <from_rom_86>.
Found 5-bit register for signal <from_rom_87>.
Found 5-bit register for signal <from_rom_88>.
Found 5-bit register for signal <from_rom_89>.
Found 5-bit register for signal <from_rom_9>.
Found 5-bit register for signal <from_rom_90>.
Found 5-bit register for signal <from_rom_91>.
Found 5-bit register for signal <from_rom_92>.
Found 5-bit register for signal <from_rom_93>.
Found 5-bit register for signal <from_rom_94>.
Found 5-bit register for signal <from_rom_95>.
Found 5-bit register for signal <from_rom_96>.
Found 5-bit register for signal <from_rom_97>.
Found 5-bit register for signal <from_rom_98>.
Found 5-bit register for signal <from_rom_99>.
Found 160-bit register for signal <register_x>.
Found 5-bit comparator equal for signal <state$cmp_eq0000> created at line 3342.
Found 5-bit comparator equal for signal <state$cmp_eq0001> created at line 3342.
Found 3-bit comparator less for signal <state$cmp_lt0000> created at line 3348.
Found 1-bit register for signal <sync_xor_1>.
Found 1-bit xor2 for signal <sync_xor_1$xor0000> created at line 2998.
Found 1-bit register for signal <sync_xor_10>.
Found 1-bit xor2 for signal <sync_xor_10$xor0000> created at line 3007.
Found 1-bit register for signal <sync_xor_100>.
Found 1-bit xor2 for signal <sync_xor_100$xor0000> created at line 3097.
Found 1-bit register for signal <sync_xor_101>.
Found 1-bit xor2 for signal <sync_xor_101$xor0000> created at line 3098.
Found 1-bit register for signal <sync_xor_102>.
Found 1-bit xor2 for signal <sync_xor_102$xor0000> created at line 3099.
Found 1-bit register for signal <sync_xor_103>.
Found 1-bit xor2 for signal <sync_xor_103$xor0000> created at line 3100.
Found 1-bit register for signal <sync_xor_104>.
Found 1-bit xor2 for signal <sync_xor_104$xor0000> created at line 3101.
Found 1-bit register for signal <sync_xor_105>.
Found 1-bit xor2 for signal <sync_xor_105$xor0000> created at line 3102.
Found 1-bit register for signal <sync_xor_106>.
Found 1-bit xor2 for signal <sync_xor_106$xor0000> created at line 3103.
Found 1-bit register for signal <sync_xor_107>.
Found 1-bit xor2 for signal <sync_xor_107$xor0000> created at line 3104.
Found 1-bit register for signal <sync_xor_108>.
Found 1-bit xor2 for signal <sync_xor_108$xor0000> created at line 3105.
Found 1-bit register for signal <sync_xor_109>.
Found 1-bit xor2 for signal <sync_xor_109$xor0000> created at line 3106.
Found 1-bit register for signal <sync_xor_11>.
Found 1-bit xor2 for signal <sync_xor_11$xor0000> created at line 3008.
Found 1-bit register for signal <sync_xor_110>.
Found 1-bit xor2 for signal <sync_xor_110$xor0000> created at line 3107.
Found 1-bit register for signal <sync_xor_111>.
Found 1-bit xor2 for signal <sync_xor_111$xor0000> created at line 3108.
Found 1-bit register for signal <sync_xor_112>.
Found 1-bit xor2 for signal <sync_xor_112$xor0000> created at line 3109.
Found 1-bit register for signal <sync_xor_113>.
Found 1-bit xor2 for signal <sync_xor_113$xor0000> created at line 3110.
Found 1-bit register for signal <sync_xor_114>.
Found 1-bit xor2 for signal <sync_xor_114$xor0000> created at line 3111.
Found 1-bit register for signal <sync_xor_115>.
Found 1-bit xor2 for signal <sync_xor_115$xor0000> created at line 3112.
Found 1-bit register for signal <sync_xor_116>.
Found 1-bit xor2 for signal <sync_xor_116$xor0000> created at line 3113.
```

```
Found 1-bit register for signal <sync_xor_117>.
Found 1-bit xor2 for signal <sync_xor_117$xor0000> created at line 3114.
Found 1-bit register for signal <sync_xor_118>.
Found 1-bit xor2 for signal <sync_xor_118$xor0000> created at line 3115.
Found 1-bit register for signal <sync_xor_119>.
Found 1-bit xor2 for signal <sync_xor_119$xor0000> created at line 3116.
Found 1-bit register for signal <sync_xor_12>.
Found 1-bit xor2 for signal <sync_xor_12$xor0000> created at line 3009.
Found 1-bit register for signal <sync_xor_120>.
Found 1-bit xor2 for signal <sync_xor_120$xor0000> created at line 3117.
Found 1-bit register for signal <sync_xor_121>.
Found 1-bit xor2 for signal <sync_xor_121$xor0000> created at line 3118.
Found 1-bit register for signal <sync_xor_122>.
Found 1-bit xor2 for signal <sync_xor_122$xor0000> created at line 3119.
Found 1-bit register for signal <sync_xor_123>.
Found 1-bit xor2 for signal <sync_xor_123$xor0000> created at line 3120.
Found 1-bit register for signal <sync_xor_124>.
Found 1-bit xor2 for signal <sync_xor_124$xor0000> created at line 3121.
Found 1-bit register for signal <sync_xor_125>.
Found 1-bit xor2 for signal <sync_xor_125$xor0000> created at line 3122.
Found 1-bit register for signal <sync_xor_126>.
Found 1-bit xor2 for signal <sync_xor_126$xor0000> created at line 3123.
Found 1-bit register for signal <sync_xor_127>.
Found 1-bit xor2 for signal <sync_xor_127$xor0000> created at line 3124.
Found 1-bit register for signal <sync_xor_128>.
Found 1-bit xor2 for signal <sync_xor_128$xor0000> created at line 3125.
Found 1-bit register for signal <sync_xor_129>.
Found 1-bit xor2 for signal <sync_xor_129$xor0000> created at line 3126.
Found 1-bit register for signal <sync_xor_13>.
Found 1-bit xor2 for signal <sync_xor_13$xor0000> created at line 3010.
Found 1-bit register for signal <sync_xor_130>.
Found 1-bit xor2 for signal <sync_xor_130$xor0000> created at line 3127.
Found 1-bit register for signal <sync_xor_131>.
Found 1-bit xor2 for signal <sync_xor_131$xor0000> created at line 3128.
Found 1-bit register for signal <sync_xor_132>.
Found 1-bit xor2 for signal <sync_xor_132$xor0000> created at line 3129.
Found 1-bit register for signal <sync_xor_133>.
Found 1-bit xor2 for signal <sync_xor_133$xor0000> created at line 3130.
Found 1-bit register for signal <sync_xor_134>.
Found 1-bit xor2 for signal <sync_xor_134$xor0000> created at line 3131.
Found 1-bit register for signal <sync_xor_135>.
Found 1-bit xor2 for signal <sync_xor_135$xor0000> created at line 3132.
Found 1-bit register for signal <sync_xor_136>.
Found 1-bit xor2 for signal <sync_xor_136$xor0000> created at line 3133.
Found 1-bit register for signal <sync_xor_137>.
Found 1-bit xor2 for signal <sync_xor_137$xor0000> created at line 3134.
Found 1-bit register for signal <sync_xor_138>.
Found 1-bit xor2 for signal <sync_xor_138$xor0000> created at line 3135.
Found 1-bit register for signal <sync_xor_139>.
Found 1-bit xor2 for signal <sync_xor_139$xor0000> created at line 3136.
Found 1-bit register for signal <sync_xor_14>.
Found 1-bit xor2 for signal <sync_xor_14$xor0000> created at line 3011.
Found 1-bit register for signal <sync_xor_140>.
Found 1-bit xor2 for signal <sync_xor_140$xor0000> created at line 3137.
Found 1-bit register for signal <sync_xor_141>.
Found 1-bit xor2 for signal <sync_xor_141$xor0000> created at line 3138.
Found 1-bit register for signal <sync_xor_142>.
Found 1-bit xor2 for signal <sync_xor_142$xor0000> created at line 3139.
Found 1-bit register for signal <sync_xor_143>.
Found 1-bit xor2 for signal <sync_xor_143$xor0000> created at line 3140.
Found 1-bit register for signal <sync_xor_144>.
Found 1-bit xor2 for signal <sync_xor_144$xor0000> created at line 3141.
Found 1-bit register for signal <sync_xor_145>.
Found 1-bit xor2 for signal <sync_xor_145$xor0000> created at line 3142.
Found 1-bit register for signal <sync_xor_146>.
Found 1-bit xor2 for signal <sync_xor_146$xor0000> created at line 3143.
Found 1-bit register for signal <sync_xor_147>.
Found 1-bit xor2 for signal <sync_xor_147$xor0000> created at line 3144.
Found 1-bit register for signal <sync_xor_148>.
Found 1-bit xor2 for signal <sync_xor_148$xor0000> created at line 3145.
Found 1-bit register for signal <sync_xor_149>.
Found 1-bit xor2 for signal <sync_xor_149$xor0000> created at line 3146.
Found 1-bit register for signal <sync_xor_15>.
Found 1-bit xor2 for signal <sync_xor_15$xor0000> created at line 3012.
Found 1-bit register for signal <sync_xor_150>.
Found 1-bit xor2 for signal <sync_xor_150$xor0000> created at line 3147.
Found 1-bit register for signal <sync_xor_151>.
Found 1-bit xor2 for signal <sync_xor_151$xor0000> created at line 3148.
Found 1-bit register for signal <sync_xor_152>.
Found 1-bit xor2 for signal <sync_xor_152$xor0000> created at line 3149.
Found 1-bit register for signal <sync_xor_153>.
Found 1-bit xor2 for signal <sync_xor_153$xor0000> created at line 3150.
Found 1-bit register for signal <sync_xor_154>.
Found 1-bit xor2 for signal <sync_xor_154$xor0000> created at line 3151.
Found 1-bit register for signal <sync_xor_155>.
Found 1-bit xor2 for signal <sync_xor_155$xor0000> created at line 3152.
Found 1-bit register for signal <sync_xor_156>.
Found 1-bit xor2 for signal <sync_xor_156$xor0000> created at line 3153.
Found 1-bit register for signal <sync_xor_157>.
Found 1-bit xor2 for signal <sync_xor_157$xor0000> created at line 3154.
Found 1-bit register for signal <sync_xor_158>.
Found 1-bit xor2 for signal <sync_xor_158$xor0000> created at line 3155.
Found 1-bit register for signal <sync_xor_159>.
Found 1-bit xor2 for signal <sync_xor_159$xor0000> created at line 3156.
Found 1-bit register for signal <sync_xor_16>.
```

```
Found 1-bit xor2 for signal <sync_xor_16$xor0000> created at line 3013.
Found 1-bit register for signal <sync_xor_160>.
Found 1-bit xor2 for signal <sync_xor_160$xor0000> created at line 3157.
Found 1-bit register for signal <sync_xor_17>.
Found 1-bit xor2 for signal <sync_xor_17$xor0000> created at line 3014.
Found 1-bit register for signal <sync_xor_18>.
Found 1-bit xor2 for signal <sync_xor_18$xor0000> created at line 3015.
Found 1-bit register for signal <sync_xor_19>.
Found 1-bit xor2 for signal <sync_xor_19$xor0000> created at line 3016.
Found 1-bit register for signal <sync_xor_2>.
Found 1-bit xor2 for signal <sync_xor_2$xor0000> created at line 2999.
Found 1-bit register for signal <sync_xor_20>.
Found 1-bit xor2 for signal <sync_xor_20$xor0000> created at line 3017.
Found 1-bit register for signal <sync_xor_21>.
Found 1-bit xor2 for signal <sync_xor_21$xor0000> created at line 3018.
Found 1-bit register for signal <sync_xor_22>.
Found 1-bit xor2 for signal <sync_xor_22$xor0000> created at line 3019.
Found 1-bit register for signal <sync_xor_23>.
Found 1-bit xor2 for signal <sync_xor_23$xor0000> created at line 3020.
Found 1-bit register for signal <sync_xor_24>.
Found 1-bit xor2 for signal <sync_xor_24$xor0000> created at line 3021.
Found 1-bit register for signal <sync_xor_25>.
Found 1-bit xor2 for signal <sync_xor_25$xor0000> created at line 3022.
Found 1-bit register for signal <sync_xor_26>.
Found 1-bit xor2 for signal <sync_xor_26$xor0000> created at line 3023.
Found 1-bit register for signal <sync_xor_27>.
Found 1-bit xor2 for signal <sync_xor_27$xor0000> created at line 3024.
Found 1-bit register for signal <sync_xor_28>.
Found 1-bit xor2 for signal <sync_xor_28$xor0000> created at line 3025.
Found 1-bit register for signal <sync_xor_29>.
Found 1-bit xor2 for signal <sync_xor_29$xor0000> created at line 3026.
Found 1-bit register for signal <sync_xor_3>.
Found 1-bit xor2 for signal <sync_xor_3$xor0000> created at line 3000.
Found 1-bit register for signal <sync_xor_30>.
Found 1-bit xor2 for signal <sync_xor_30$xor0000> created at line 3027.
Found 1-bit register for signal <sync_xor_31>.
Found 1-bit xor2 for signal <sync_xor_31$xor0000> created at line 3028.
Found 1-bit register for signal <sync_xor_32>.
Found 1-bit xor2 for signal <sync_xor_32$xor0000> created at line 3029.
Found 1-bit register for signal <sync_xor_33>.
Found 1-bit xor2 for signal <sync_xor_33$xor0000> created at line 3030.
Found 1-bit register for signal <sync_xor_34>.
Found 1-bit xor2 for signal <sync_xor_34$xor0000> created at line 3031.
Found 1-bit register for signal <sync_xor_35>.
Found 1-bit xor2 for signal <sync_xor_35$xor0000> created at line 3032.
Found 1-bit register for signal <sync_xor_36>.
Found 1-bit xor2 for signal <sync_xor_36$xor0000> created at line 3033.
Found 1-bit register for signal <sync_xor_37>.
Found 1-bit xor2 for signal <sync_xor_37$xor0000> created at line 3034.
Found 1-bit register for signal <sync_xor_38>.
Found 1-bit xor2 for signal <sync_xor_38$xor0000> created at line 3035.
Found 1-bit register for signal <sync_xor_39>.
Found 1-bit xor2 for signal <sync_xor_39$xor0000> created at line 3036.
Found 1-bit register for signal <sync_xor_4>.
Found 1-bit xor2 for signal <sync_xor_4$xor0000> created at line 3001.
Found 1-bit register for signal <sync_xor_40>.
Found 1-bit xor2 for signal <sync_xor_40$xor0000> created at line 3037.
Found 1-bit register for signal <sync_xor_41>.
Found 1-bit xor2 for signal <sync_xor_41$xor0000> created at line 3038.
Found 1-bit register for signal <sync_xor_42>.
Found 1-bit xor2 for signal <sync_xor_42$xor0000> created at line 3039.
Found 1-bit register for signal <sync_xor_43>.
Found 1-bit xor2 for signal <sync_xor_43$xor0000> created at line 3040.
Found 1-bit register for signal <sync_xor_44>.
Found 1-bit xor2 for signal <sync_xor_44$xor0000> created at line 3041.
Found 1-bit register for signal <sync_xor_45>.
Found 1-bit xor2 for signal <sync_xor_45$xor0000> created at line 3042.
Found 1-bit register for signal <sync_xor_46>.
Found 1-bit xor2 for signal <sync_xor_46$xor0000> created at line 3043.
Found 1-bit register for signal <sync_xor_47>.
Found 1-bit xor2 for signal <sync_xor_47$xor0000> created at line 3044.
Found 1-bit register for signal <sync_xor_48>.
Found 1-bit xor2 for signal <sync_xor_48$xor0000> created at line 3045.
Found 1-bit register for signal <sync_xor_49>.
Found 1-bit xor2 for signal <sync_xor_49$xor0000> created at line 3046.
Found 1-bit register for signal <sync_xor_5>.
Found 1-bit xor2 for signal <sync_xor_5$xor0000> created at line 3002.
Found 1-bit register for signal <sync_xor_50>.
Found 1-bit xor2 for signal <sync_xor_50$xor0000> created at line 3047.
Found 1-bit register for signal <sync_xor_51>.
Found 1-bit xor2 for signal <sync_xor_51$xor0000> created at line 3048.
Found 1-bit register for signal <sync_xor_52>.
Found 1-bit xor2 for signal <sync_xor_52$xor0000> created at line 3049.
Found 1-bit register for signal <sync_xor_53>.
Found 1-bit xor2 for signal <sync_xor_53$xor0000> created at line 3050.
Found 1-bit register for signal <sync_xor_54>.
Found 1-bit xor2 for signal <sync_xor_54$xor0000> created at line 3051.
Found 1-bit register for signal <sync_xor_55>.
Found 1-bit xor2 for signal <sync_xor_55$xor0000> created at line 3052.
Found 1-bit register for signal <sync_xor_56>.
Found 1-bit xor2 for signal <sync_xor_56$xor0000> created at line 3053.
Found 1-bit register for signal <sync_xor_57>.
Found 1-bit xor2 for signal <sync_xor_57$xor0000> created at line 3054.
Found 1-bit register for signal <sync_xor_58>.
Found 1-bit xor2 for signal <sync_xor_58$xor0000> created at line 3055.
```

```
Found 1-bit register for signal <sync_xor_59>.
Found 1-bit xor2 for signal <sync_xor_59$xor0000> created at line 3056.
Found 1-bit register for signal <sync_xor_6>.
Found 1-bit xor2 for signal <sync_xor_6$xor0000> created at line 3003.
Found 1-bit register for signal <sync_xor_60>.
Found 1-bit xor2 for signal <sync_xor_60$xor0000> created at line 3057.
Found 1-bit register for signal <sync_xor_61>.
Found 1-bit xor2 for signal <sync_xor_61$xor0000> created at line 3058.
Found 1-bit register for signal <sync_xor_62>.
Found 1-bit xor2 for signal <sync_xor_62$xor0000> created at line 3059.
Found 1-bit register for signal <sync_xor_63>.
Found 1-bit xor2 for signal <sync_xor_63$xor0000> created at line 3060.
Found 1-bit register for signal <sync_xor_64>.
Found 1-bit xor2 for signal <sync_xor_64$xor0000> created at line 3061.
Found 1-bit register for signal <sync_xor_65>.
Found 1-bit xor2 for signal <sync_xor_65$xor0000> created at line 3062.
Found 1-bit register for signal <sync_xor_66>.
Found 1-bit xor2 for signal <sync_xor_66$xor0000> created at line 3063.
Found 1-bit register for signal <sync_xor_67>.
Found 1-bit xor2 for signal <sync_xor_67$xor0000> created at line 3064.
Found 1-bit register for signal <sync_xor_68>.
Found 1-bit xor2 for signal <sync_xor_68$xor0000> created at line 3065.
Found 1-bit register for signal <sync_xor_69>.
Found 1-bit xor2 for signal <sync_xor_69$xor0000> created at line 3066.
Found 1-bit register for signal <sync_xor_7>.
Found 1-bit xor2 for signal <sync_xor_7$xor0000> created at line 3004.
Found 1-bit register for signal <sync_xor_70>.
Found 1-bit xor2 for signal <sync_xor_70$xor0000> created at line 3067.
Found 1-bit register for signal <sync_xor_71>.
Found 1-bit xor2 for signal <sync_xor_71$xor0000> created at line 3068.
Found 1-bit register for signal <sync_xor_72>.
Found 1-bit xor2 for signal <sync_xor_72$xor0000> created at line 3069.
Found 1-bit register for signal <sync_xor_73>.
Found 1-bit xor2 for signal <sync_xor_73$xor0000> created at line 3070.
Found 1-bit register for signal <sync_xor_74>.
Found 1-bit xor2 for signal <sync_xor_74$xor0000> created at line 3071.
Found 1-bit register for signal <sync_xor_75>.
Found 1-bit xor2 for signal <sync_xor_75$xor0000> created at line 3072.
Found 1-bit register for signal <sync_xor_76>.
Found 1-bit xor2 for signal <sync_xor_76$xor0000> created at line 3073.
Found 1-bit register for signal <sync_xor_77>.
Found 1-bit xor2 for signal <sync_xor_77$xor0000> created at line 3074.
Found 1-bit register for signal <sync_xor_78>.
Found 1-bit xor2 for signal <sync_xor_78$xor0000> created at line 3075.
Found 1-bit register for signal <sync_xor_79>.
Found 1-bit xor2 for signal <sync_xor_79$xor0000> created at line 3076.
Found 1-bit register for signal <sync_xor_8>.
Found 1-bit xor2 for signal <sync_xor_8$xor0000> created at line 3005.
Found 1-bit register for signal <sync_xor_80>.
Found 1-bit xor2 for signal <sync_xor_80$xor0000> created at line 3077.
Found 1-bit register for signal <sync_xor_81>.
Found 1-bit xor2 for signal <sync_xor_81$xor0000> created at line 3078.
Found 1-bit register for signal <sync_xor_82>.
Found 1-bit xor2 for signal <sync_xor_82$xor0000> created at line 3079.
Found 1-bit register for signal <sync_xor_83>.
Found 1-bit xor2 for signal <sync_xor_83$xor0000> created at line 3080.
Found 1-bit register for signal <sync_xor_84>.
Found 1-bit xor2 for signal <sync_xor_84$xor0000> created at line 3081.
Found 1-bit register for signal <sync_xor_85>.
Found 1-bit xor2 for signal <sync_xor_85$xor0000> created at line 3082.
Found 1-bit register for signal <sync_xor_86>.
Found 1-bit xor2 for signal <sync_xor_86$xor0000> created at line 3083.
Found 1-bit register for signal <sync_xor_87>.
Found 1-bit xor2 for signal <sync_xor_87$xor0000> created at line 3084.
Found 1-bit register for signal <sync_xor_88>.
Found 1-bit xor2 for signal <sync_xor_88$xor0000> created at line 3085.
Found 1-bit register for signal <sync_xor_89>.
Found 1-bit xor2 for signal <sync_xor_89$xor0000> created at line 3086.
Found 1-bit register for signal <sync_xor_9>.
Found 1-bit xor2 for signal <sync_xor_9$xor0000> created at line 3006.
Found 1-bit register for signal <sync_xor_90>.
Found 1-bit xor2 for signal <sync_xor_90$xor0000> created at line 3087.
Found 1-bit register for signal <sync_xor_91>.
Found 1-bit xor2 for signal <sync_xor_91$xor0000> created at line 3088.
Found 1-bit register for signal <sync_xor_92>.
Found 1-bit xor2 for signal <sync_xor_92$xor0000> created at line 3089.
Found 1-bit register for signal <sync_xor_93>.
Found 1-bit xor2 for signal <sync_xor_93$xor0000> created at line 3090.
Found 1-bit register for signal <sync_xor_94>.
Found 1-bit xor2 for signal <sync_xor_94$xor0000> created at line 3091.
Found 1-bit register for signal <sync_xor_95>.
Found 1-bit xor2 for signal <sync_xor_95$xor0000> created at line 3092.
Found 1-bit register for signal <sync_xor_96>.
Found 1-bit xor2 for signal <sync_xor_96$xor0000> created at line 3093.
Found 1-bit register for signal <sync_xor_97>.
Found 1-bit xor2 for signal <sync_xor_97$xor0000> created at line 3094.
Found 1-bit register for signal <sync_xor_98>.
Found 1-bit xor2 for signal <sync_xor_98$xor0000> created at line 3095.
Found 1-bit register for signal <sync_xor_99>.
Found 1-bit xor2 for signal <sync_xor_99$xor0000> created at line 3096.
Found 1-bit register for signal <tmp_1>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_1$mux0000> created at line 2503.
Found 1-bit register for signal <tmp_10>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_10$mux0000> created at line 2521.
Found 1-bit register for signal <tmp_100>.
```

```
Found 1-bit 5-to-1 multiplexer for signal <tmp_100$mux0000> created at line 2701.
Found 1-bit register for signal <tmp_101>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_101$mux0000> created at line 2703.
Found 1-bit register for signal <tmp_102>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_102$mux0000> created at line 2705.
Found 1-bit register for signal <tmp_103>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_103$mux0000> created at line 2707.
Found 1-bit register for signal <tmp_104>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_104$mux0000> created at line 2709.
Found 1-bit register for signal <tmp_105>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_105$mux0000> created at line 2711.
Found 1-bit register for signal <tmp_106>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_106$mux0000> created at line 2713.
Found 1-bit register for signal <tmp_107>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_107$mux0000> created at line 2715.
Found 1-bit register for signal <tmp_108>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_108$mux0000> created at line 2717.
Found 1-bit register for signal <tmp_109>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_109$mux0000> created at line 2719.
Found 1-bit register for signal <tmp_11>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_11$mux0000> created at line 2523.
Found 1-bit register for signal <tmp_110>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_110$mux0000> created at line 2721.
Found 1-bit register for signal <tmp_111>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_111$mux0000> created at line 2723.
Found 1-bit register for signal <tmp_112>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_112$mux0000> created at line 2725.
Found 1-bit register for signal <tmp_113>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_113$mux0000> created at line 2727.
Found 1-bit register for signal <tmp_114>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_114$mux0000> created at line 2729.
Found 1-bit register for signal <tmp_115>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_115$mux0000> created at line 2731.
Found 1-bit register for signal <tmp_116>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_116$mux0000> created at line 2733.
Found 1-bit register for signal <tmp_117>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_117$mux0000> created at line 2735.
Found 1-bit register for signal <tmp_118>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_118$mux0000> created at line 2737.
Found 1-bit register for signal <tmp_119>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_119$mux0000> created at line 2739.
Found 1-bit register for signal <tmp_12>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_12$mux0000> created at line 2525.
Found 1-bit register for signal <tmp_120>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_120$mux0000> created at line 2741.
Found 1-bit register for signal <tmp_121>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_121$mux0000> created at line 2743.
Found 1-bit register for signal <tmp_122>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_122$mux0000> created at line 2745.
Found 1-bit register for signal <tmp_123>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_123$mux0000> created at line 2747.
Found 1-bit register for signal <tmp_124>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_124$mux0000> created at line 2749.
Found 1-bit register for signal <tmp_125>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_125$mux0000> created at line 2751.
Found 1-bit register for signal <tmp_126>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_126$mux0000> created at line 2753.
Found 1-bit register for signal <tmp_127>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_127$mux0000> created at line 2755.
Found 1-bit register for signal <tmp_128>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_128$mux0000> created at line 2757.
Found 1-bit register for signal <tmp_129>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_129$mux0000> created at line 2759.
Found 1-bit register for signal <tmp_13>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_13$mux0000> created at line 2527.
Found 1-bit register for signal <tmp_130>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_130$mux0000> created at line 2761.
Found 1-bit register for signal <tmp_131>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_131$mux0000> created at line 2763.
Found 1-bit register for signal <tmp_132>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_132$mux0000> created at line 2765.
Found 1-bit register for signal <tmp_133>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_133$mux0000> created at line 2767.
Found 1-bit register for signal <tmp_134>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_134$mux0000> created at line 2769.
Found 1-bit register for signal <tmp_135>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_135$mux0000> created at line 2771.
Found 1-bit register for signal <tmp_136>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_136$mux0000> created at line 2773.
Found 1-bit register for signal <tmp_137>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_137$mux0000> created at line 2775.
Found 1-bit register for signal <tmp_138>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_138$mux0000> created at line 2777.
Found 1-bit register for signal <tmp_139>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_139$mux0000> created at line 2779.
Found 1-bit register for signal <tmp_14>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_14$mux0000> created at line 2529.
Found 1-bit register for signal <tmp_140>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_140$mux0000> created at line 2781.
Found 1-bit register for signal <tmp_141>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_141$mux0000> created at line 2783.
Found 1-bit register for signal <tmp_142>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_142$mux0000> created at line 2785.
Found 1-bit register for signal <tmp_143>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_143$mux0000> created at line 2787.
```

```
Found 1-bit register for signal <tmp_144>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_144$mux0000> created at line 2789.
Found 1-bit register for signal <tmp_145>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_145$mux0000> created at line 2791.
Found 1-bit register for signal <tmp_146>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_146$mux0000> created at line 2793.
Found 1-bit register for signal <tmp_147>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_147$mux0000> created at line 2795.
Found 1-bit register for signal <tmp_148>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_148$mux0000> created at line 2797.
Found 1-bit register for signal <tmp_149>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_149$mux0000> created at line 2799.
Found 1-bit register for signal <tmp_15>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_15$mux0000> created at line 2531.
Found 1-bit register for signal <tmp_150>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_150$mux0000> created at line 2801.
Found 1-bit register for signal <tmp_151>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_151$mux0000> created at line 2803.
Found 1-bit register for signal <tmp_152>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_152$mux0000> created at line 2805.
Found 1-bit register for signal <tmp_153>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_153$mux0000> created at line 2807.
Found 1-bit register for signal <tmp_154>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_154$mux0000> created at line 2809.
Found 1-bit register for signal <tmp_155>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_155$mux0000> created at line 2811.
Found 1-bit register for signal <tmp_156>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_156$mux0000> created at line 2813.
Found 1-bit register for signal <tmp_157>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_157$mux0000> created at line 2815.
Found 1-bit register for signal <tmp_158>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_158$mux0000> created at line 2817.
Found 1-bit register for signal <tmp_159>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_159$mux0000> created at line 2819.
Found 1-bit register for signal <tmp_16>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_16$mux0000> created at line 2533.
Found 1-bit register for signal <tmp_160>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_160$mux0000> created at line 2821.
Found 1-bit register for signal <tmp_17>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_17$mux0000> created at line 2535.
Found 1-bit register for signal <tmp_18>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_18$mux0000> created at line 2537.
Found 1-bit register for signal <tmp_19>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_19$mux0000> created at line 2539.
Found 1-bit register for signal <tmp_2>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_2$mux0000> created at line 2505.
Found 1-bit register for signal <tmp_20>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_20$mux0000> created at line 2541.
Found 1-bit register for signal <tmp_21>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_21$mux0000> created at line 2543.
Found 1-bit register for signal <tmp_22>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_22$mux0000> created at line 2545.
Found 1-bit register for signal <tmp_23>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_23$mux0000> created at line 2547.
Found 1-bit register for signal <tmp_24>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_24$mux0000> created at line 2549.
Found 1-bit register for signal <tmp_25>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_25$mux0000> created at line 2551.
Found 1-bit register for signal <tmp_26>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_26$mux0000> created at line 2553.
Found 1-bit register for signal <tmp_27>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_27$mux0000> created at line 2555.
Found 1-bit register for signal <tmp_28>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_28$mux0000> created at line 2557.
Found 1-bit register for signal <tmp_29>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_29$mux0000> created at line 2559.
Found 1-bit register for signal <tmp_3>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_3$mux0000> created at line 2507.
Found 1-bit register for signal <tmp_30>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_30$mux0000> created at line 2561.
Found 1-bit register for signal <tmp_31>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_31$mux0000> created at line 2563.
Found 1-bit register for signal <tmp_32>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_32$mux0000> created at line 2565.
Found 1-bit register for signal <tmp_33>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_33$mux0000> created at line 2567.
Found 1-bit register for signal <tmp_34>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_34$mux0000> created at line 2569.
Found 1-bit register for signal <tmp_35>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_35$mux0000> created at line 2571.
Found 1-bit register for signal <tmp_36>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_36$mux0000> created at line 2573.
Found 1-bit register for signal <tmp_37>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_37$mux0000> created at line 2575.
Found 1-bit register for signal <tmp_38>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_38$mux0000> created at line 2577.
Found 1-bit register for signal <tmp_39>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_39$mux0000> created at line 2579.
Found 1-bit register for signal <tmp_4>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_4$mux0000> created at line 2509.
Found 1-bit register for signal <tmp_40>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_40$mux0000> created at line 2581.
Found 1-bit register for signal <tmp_41>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_41$mux0000> created at line 2583.
Found 1-bit register for signal <tmp_42>.
```

```
Found 1-bit 5-to-1 multiplexer for signal <tmp_42$mux0000> created at line 2585.
Found 1-bit register for signal <tmp_43>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_43$mux0000> created at line 2587.
Found 1-bit register for signal <tmp_44>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_44$mux0000> created at line 2589.
Found 1-bit register for signal <tmp_45>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_45$mux0000> created at line 2591.
Found 1-bit register for signal <tmp_46>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_46$mux0000> created at line 2593.
Found 1-bit register for signal <tmp_47>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_47$mux0000> created at line 2595.
Found 1-bit register for signal <tmp_48>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_48$mux0000> created at line 2597.
Found 1-bit register for signal <tmp_49>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_49$mux0000> created at line 2599.
Found 1-bit register for signal <tmp_5>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_5$mux0000> created at line 2511.
Found 1-bit register for signal <tmp_50>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_50$mux0000> created at line 2601.
Found 1-bit register for signal <tmp_51>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_51$mux0000> created at line 2603.
Found 1-bit register for signal <tmp_52>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_52$mux0000> created at line 2605.
Found 1-bit register for signal <tmp_53>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_53$mux0000> created at line 2607.
Found 1-bit register for signal <tmp_54>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_54$mux0000> created at line 2609.
Found 1-bit register for signal <tmp_55>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_55$mux0000> created at line 2611.
Found 1-bit register for signal <tmp_56>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_56$mux0000> created at line 2613.
Found 1-bit register for signal <tmp_57>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_57$mux0000> created at line 2615.
Found 1-bit register for signal <tmp_58>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_58$mux0000> created at line 2617.
Found 1-bit register for signal <tmp_59>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_59$mux0000> created at line 2619.
Found 1-bit register for signal <tmp_6>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_6$mux0000> created at line 2513.
Found 1-bit register for signal <tmp_60>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_60$mux0000> created at line 2621.
Found 1-bit register for signal <tmp_61>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_61$mux0000> created at line 2623.
Found 1-bit register for signal <tmp_62>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_62$mux0000> created at line 2625.
Found 1-bit register for signal <tmp_63>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_63$mux0000> created at line 2627.
Found 1-bit register for signal <tmp_64>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_64$mux0000> created at line 2629.
Found 1-bit register for signal <tmp_65>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_65$mux0000> created at line 2631.
Found 1-bit register for signal <tmp_66>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_66$mux0000> created at line 2633.
Found 1-bit register for signal <tmp_67>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_67$mux0000> created at line 2635.
Found 1-bit register for signal <tmp_68>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_68$mux0000> created at line 2637.
Found 1-bit register for signal <tmp_69>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_69$mux0000> created at line 2639.
Found 1-bit register for signal <tmp_7>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_7$mux0000> created at line 2515.
Found 1-bit register for signal <tmp_70>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_70$mux0000> created at line 2641.
Found 1-bit register for signal <tmp_71>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_71$mux0000> created at line 2643.
Found 1-bit register for signal <tmp_72>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_72$mux0000> created at line 2645.
Found 1-bit register for signal <tmp_73>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_73$mux0000> created at line 2647.
Found 1-bit register for signal <tmp_74>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_74$mux0000> created at line 2649.
Found 1-bit register for signal <tmp_75>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_75$mux0000> created at line 2651.
Found 1-bit register for signal <tmp_76>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_76$mux0000> created at line 2653.
Found 1-bit register for signal <tmp_77>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_77$mux0000> created at line 2655.
Found 1-bit register for signal <tmp_78>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_78$mux0000> created at line 2657.
Found 1-bit register for signal <tmp_79>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_79$mux0000> created at line 2659.
Found 1-bit register for signal <tmp_8>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_8$mux0000> created at line 2517.
Found 1-bit register for signal <tmp_80>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_80$mux0000> created at line 2661.
Found 1-bit register for signal <tmp_81>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_81$mux0000> created at line 2663.
Found 1-bit register for signal <tmp_82>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_82$mux0000> created at line 2665.
Found 1-bit register for signal <tmp_83>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_83$mux0000> created at line 2667.
Found 1-bit register for signal <tmp_84>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_84$mux0000> created at line 2669.
Found 1-bit register for signal <tmp_85>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_85$mux0000> created at line 2671.
```

```
Found 1-bit register for signal <tmp_86>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_86$mux0000> created at line 2673.
Found 1-bit register for signal <tmp_87>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_87$mux0000> created at line 2675.
Found 1-bit register for signal <tmp_88>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_88$mux0000> created at line 2677.
Found 1-bit register for signal <tmp_89>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_89$mux0000> created at line 2679.
Found 1-bit register for signal <tmp_9>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_9$mux0000> created at line 2519.
Found 1-bit register for signal <tmp_90>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_90$mux0000> created at line 2681.
Found 1-bit register for signal <tmp_91>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_91$mux0000> created at line 2683.
Found 1-bit register for signal <tmp_92>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_92$mux0000> created at line 2685.
Found 1-bit register for signal <tmp_93>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_93$mux0000> created at line 2687.
Found 1-bit register for signal <tmp_94>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_94$mux0000> created at line 2689.
Found 1-bit register for signal <tmp_95>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_95$mux0000> created at line 2691.
Found 1-bit register for signal <tmp_96>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_96$mux0000> created at line 2693.
Found 1-bit register for signal <tmp_97>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_97$mux0000> created at line 2695.
Found 1-bit register for signal <tmp_98>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_98$mux0000> created at line 2697.
Found 1-bit register for signal <tmp_99>.
Found 1-bit 5-to-1 multiplexer for signal <tmp_99$mux0000> created at line 2699.
Found 1-bit register for signal <xor_1>.
Found 1-bit xor2 for signal <xor_1$xor0000> created at line 2504.
Found 1-bit register for signal <xor_10>.
Found 1-bit xor2 for signal <xor_10$xor0000> created at line 2522.
Found 1-bit register for signal <xor_100>.
Found 1-bit xor2 for signal <xor_100$xor0000> created at line 2702.
Found 1-bit register for signal <xor_101>.
Found 1-bit xor2 for signal <xor_101$xor0000> created at line 2704.
Found 1-bit register for signal <xor_102>.
Found 1-bit xor2 for signal <xor_102$xor0000> created at line 2706.
Found 1-bit register for signal <xor_103>.
Found 1-bit xor2 for signal <xor_103$xor0000> created at line 2708.
Found 1-bit register for signal <xor_104>.
Found 1-bit xor2 for signal <xor_104$xor0000> created at line 2710.
Found 1-bit register for signal <xor_105>.
Found 1-bit xor2 for signal <xor_105$xor0000> created at line 2712.
Found 1-bit register for signal <xor_106>.
Found 1-bit xor2 for signal <xor_106$xor0000> created at line 2714.
Found 1-bit register for signal <xor_107>.
Found 1-bit xor2 for signal <xor_107$xor0000> created at line 2716.
Found 1-bit register for signal <xor_108>.
Found 1-bit xor2 for signal <xor_108$xor0000> created at line 2718.
Found 1-bit register for signal <xor_109>.
Found 1-bit xor2 for signal <xor_109$xor0000> created at line 2720.
Found 1-bit register for signal <xor_11>.
Found 1-bit xor2 for signal <xor_11$xor0000> created at line 2524.
Found 1-bit register for signal <xor_110>.
Found 1-bit xor2 for signal <xor_110$xor0000> created at line 2722.
Found 1-bit register for signal <xor_111>.
Found 1-bit xor2 for signal <xor_111$xor0000> created at line 2724.
Found 1-bit register for signal <xor_112>.
Found 1-bit xor2 for signal <xor_112$xor0000> created at line 2726.
Found 1-bit register for signal <xor_113>.
Found 1-bit xor2 for signal <xor_113$xor0000> created at line 2728.
Found 1-bit register for signal <xor_114>.
Found 1-bit xor2 for signal <xor_114$xor0000> created at line 2730.
Found 1-bit register for signal <xor_115>.
Found 1-bit xor2 for signal <xor_115$xor0000> created at line 2732.
Found 1-bit register for signal <xor_116>.
Found 1-bit xor2 for signal <xor_116$xor0000> created at line 2734.
Found 1-bit register for signal <xor_117>.
Found 1-bit xor2 for signal <xor_117$xor0000> created at line 2736.
Found 1-bit register for signal <xor_118>.
Found 1-bit xor2 for signal <xor_118$xor0000> created at line 2738.
Found 1-bit register for signal <xor_119>.
Found 1-bit xor2 for signal <xor_119$xor0000> created at line 2740.
Found 1-bit register for signal <xor_12>.
Found 1-bit xor2 for signal <xor_12$xor0000> created at line 2526.
Found 1-bit register for signal <xor_120>.
Found 1-bit xor2 for signal <xor_120$xor0000> created at line 2742.
Found 1-bit register for signal <xor_121>.
Found 1-bit xor2 for signal <xor_121$xor0000> created at line 2744.
Found 1-bit register for signal <xor_122>.
Found 1-bit xor2 for signal <xor_122$xor0000> created at line 2746.
Found 1-bit register for signal <xor_123>.
Found 1-bit xor2 for signal <xor_123$xor0000> created at line 2748.
Found 1-bit register for signal <xor_124>.
Found 1-bit xor2 for signal <xor_124$xor0000> created at line 2750.
Found 1-bit register for signal <xor_125>.
Found 1-bit xor2 for signal <xor_125$xor0000> created at line 2752.
Found 1-bit register for signal <xor_126>.
Found 1-bit xor2 for signal <xor_126$xor0000> created at line 2754.
Found 1-bit register for signal <xor_127>.
Found 1-bit xor2 for signal <xor_127$xor0000> created at line 2756.
Found 1-bit register for signal <xor_128>.
```

```
Found 1-bit xor2 for signal <xor_128$xor0000> created at line 2758.
Found 1-bit register for signal <xor_129>.
Found 1-bit xor2 for signal <xor_129$xor0000> created at line 2760.
Found 1-bit register for signal <xor_13>.
Found 1-bit xor2 for signal <xor_13$xor0000> created at line 2528.
Found 1-bit register for signal <xor_130>.
Found 1-bit xor2 for signal <xor_130$xor0000> created at line 2762.
Found 1-bit register for signal <xor_131>.
Found 1-bit xor2 for signal <xor_131$xor0000> created at line 2764.
Found 1-bit register for signal <xor_132>.
Found 1-bit xor2 for signal <xor_132$xor0000> created at line 2766.
Found 1-bit register for signal <xor_133>.
Found 1-bit xor2 for signal <xor_133$xor0000> created at line 2768.
Found 1-bit register for signal <xor_134>.
Found 1-bit xor2 for signal <xor_134$xor0000> created at line 2770.
Found 1-bit register for signal <xor_135>.
Found 1-bit xor2 for signal <xor_135$xor0000> created at line 2772.
Found 1-bit register for signal <xor_136>.
Found 1-bit xor2 for signal <xor_136$xor0000> created at line 2774.
Found 1-bit register for signal <xor_137>.
Found 1-bit xor2 for signal <xor_137$xor0000> created at line 2776.
Found 1-bit register for signal <xor_138>.
Found 1-bit xor2 for signal <xor_138$xor0000> created at line 2778.
Found 1-bit register for signal <xor_139>.
Found 1-bit xor2 for signal <xor_139$xor0000> created at line 2780.
Found 1-bit register for signal <xor_14>.
Found 1-bit xor2 for signal <xor_14$xor0000> created at line 2530.
Found 1-bit register for signal <xor_140>.
Found 1-bit xor2 for signal <xor_140$xor0000> created at line 2782.
Found 1-bit register for signal <xor_141>.
Found 1-bit xor2 for signal <xor_141$xor0000> created at line 2784.
Found 1-bit register for signal <xor_142>.
Found 1-bit xor2 for signal <xor_142$xor0000> created at line 2786.
Found 1-bit register for signal <xor_143>.
Found 1-bit xor2 for signal <xor_143$xor0000> created at line 2788.
Found 1-bit register for signal <xor_144>.
Found 1-bit xor2 for signal <xor_144$xor0000> created at line 2790.
Found 1-bit register for signal <xor_145>.
Found 1-bit xor2 for signal <xor_145$xor0000> created at line 2792.
Found 1-bit register for signal <xor_146>.
Found 1-bit xor2 for signal <xor_146$xor0000> created at line 2794.
Found 1-bit register for signal <xor_147>.
Found 1-bit xor2 for signal <xor_147$xor0000> created at line 2796.
Found 1-bit register for signal <xor_148>.
Found 1-bit xor2 for signal <xor_148$xor0000> created at line 2798.
Found 1-bit register for signal <xor_149>.
Found 1-bit xor2 for signal <xor_149$xor0000> created at line 2800.
Found 1-bit register for signal <xor_15>.
Found 1-bit xor2 for signal <xor_15$xor0000> created at line 2532.
Found 1-bit register for signal <xor_150>.
Found 1-bit xor2 for signal <xor_150$xor0000> created at line 2802.
Found 1-bit register for signal <xor_151>.
Found 1-bit xor2 for signal <xor_151$xor0000> created at line 2804.
Found 1-bit register for signal <xor_152>.
Found 1-bit xor2 for signal <xor_152$xor0000> created at line 2806.
Found 1-bit register for signal <xor_153>.
Found 1-bit xor2 for signal <xor_153$xor0000> created at line 2808.
Found 1-bit register for signal <xor_154>.
Found 1-bit xor2 for signal <xor_154$xor0000> created at line 2810.
Found 1-bit register for signal <xor_155>.
Found 1-bit xor2 for signal <xor_155$xor0000> created at line 2812.
Found 1-bit register for signal <xor_156>.
Found 1-bit xor2 for signal <xor_156$xor0000> created at line 2814.
Found 1-bit register for signal <xor_157>.
Found 1-bit xor2 for signal <xor_157$xor0000> created at line 2816.
Found 1-bit register for signal <xor_158>.
Found 1-bit xor2 for signal <xor_158$xor0000> created at line 2818.
Found 1-bit register for signal <xor_159>.
Found 1-bit xor2 for signal <xor_159$xor0000> created at line 2820.
Found 1-bit register for signal <xor_16>.
Found 1-bit xor2 for signal <xor_16$xor0000> created at line 2534.
Found 1-bit register for signal <xor_160>.
Found 1-bit xor2 for signal <xor_160$xor0000> created at line 2822.
Found 1-bit register for signal <xor_17>.
Found 1-bit xor2 for signal <xor_17$xor0000> created at line 2536.
Found 1-bit register for signal <xor_18>.
Found 1-bit xor2 for signal <xor_18$xor0000> created at line 2538.
Found 1-bit register for signal <xor_19>.
Found 1-bit xor2 for signal <xor_19$xor0000> created at line 2540.
Found 1-bit register for signal <xor_2>.
Found 1-bit xor2 for signal <xor_2$xor0000> created at line 2506.
Found 1-bit register for signal <xor_20>.
Found 1-bit xor2 for signal <xor_20$xor0000> created at line 2542.
Found 1-bit register for signal <xor_21>.
Found 1-bit xor2 for signal <xor_21$xor0000> created at line 2544.
Found 1-bit register for signal <xor_22>.
Found 1-bit xor2 for signal <xor_22$xor0000> created at line 2546.
Found 1-bit register for signal <xor_23>.
Found 1-bit xor2 for signal <xor_23$xor0000> created at line 2548.
Found 1-bit register for signal <xor_24>.
Found 1-bit xor2 for signal <xor_24$xor0000> created at line 2550.
Found 1-bit register for signal <xor_25>.
Found 1-bit xor2 for signal <xor_25$xor0000> created at line 2552.
Found 1-bit register for signal <xor_26>.
Found 1-bit xor2 for signal <xor_26$xor0000> created at line 2554.
```

```
Found 1-bit register for signal <xor_27>.
Found 1-bit xor2 for signal <xor_27$xor0000> created at line 2556.
Found 1-bit register for signal <xor_28>.
Found 1-bit xor2 for signal <xor_28$xor0000> created at line 2558.
Found 1-bit register for signal <xor_29>.
Found 1-bit xor2 for signal <xor_29$xor0000> created at line 2560.
Found 1-bit register for signal <xor_3>.
Found 1-bit xor2 for signal <xor_3$xor0000> created at line 2508.
Found 1-bit register for signal <xor_30>.
Found 1-bit xor2 for signal <xor_30$xor0000> created at line 2562.
Found 1-bit register for signal <xor_31>.
Found 1-bit xor2 for signal <xor_31$xor0000> created at line 2564.
Found 1-bit register for signal <xor_32>.
Found 1-bit xor2 for signal <xor_32$xor0000> created at line 2566.
Found 1-bit register for signal <xor_33>.
Found 1-bit xor2 for signal <xor_33$xor0000> created at line 2568.
Found 1-bit register for signal <xor_34>.
Found 1-bit xor2 for signal <xor_34$xor0000> created at line 2570.
Found 1-bit register for signal <xor_35>.
Found 1-bit xor2 for signal <xor_35$xor0000> created at line 2572.
Found 1-bit register for signal <xor_36>.
Found 1-bit xor2 for signal <xor_36$xor0000> created at line 2574.
Found 1-bit register for signal <xor_37>.
Found 1-bit xor2 for signal <xor_37$xor0000> created at line 2576.
Found 1-bit register for signal <xor_38>.
Found 1-bit xor2 for signal <xor_38$xor0000> created at line 2578.
Found 1-bit register for signal <xor_39>.
Found 1-bit xor2 for signal <xor_39$xor0000> created at line 2580.
Found 1-bit register for signal <xor_4>.
Found 1-bit xor2 for signal <xor_4$xor0000> created at line 2510.
Found 1-bit register for signal <xor_40>.
Found 1-bit xor2 for signal <xor_40$xor0000> created at line 2582.
Found 1-bit register for signal <xor_41>.
Found 1-bit xor2 for signal <xor_41$xor0000> created at line 2584.
Found 1-bit register for signal <xor_42>.
Found 1-bit xor2 for signal <xor_42$xor0000> created at line 2586.
Found 1-bit register for signal <xor_43>.
Found 1-bit xor2 for signal <xor_43$xor0000> created at line 2588.
Found 1-bit register for signal <xor_44>.
Found 1-bit xor2 for signal <xor_44$xor0000> created at line 2590.
Found 1-bit register for signal <xor_45>.
Found 1-bit xor2 for signal <xor_45$xor0000> created at line 2592.
Found 1-bit register for signal <xor_46>.
Found 1-bit xor2 for signal <xor_46$xor0000> created at line 2594.
Found 1-bit register for signal <xor_47>.
Found 1-bit xor2 for signal <xor_47$xor0000> created at line 2596.
Found 1-bit register for signal <xor_48>.
Found 1-bit xor2 for signal <xor_48$xor0000> created at line 2598.
Found 1-bit register for signal <xor_49>.
Found 1-bit xor2 for signal <xor_49$xor0000> created at line 2600.
Found 1-bit register for signal <xor_5>.
Found 1-bit xor2 for signal <xor_5$xor0000> created at line 2512.
Found 1-bit register for signal <xor_50>.
Found 1-bit xor2 for signal <xor_50$xor0000> created at line 2602.
Found 1-bit register for signal <xor_51>.
Found 1-bit xor2 for signal <xor_51$xor0000> created at line 2604.
Found 1-bit register for signal <xor_52>.
Found 1-bit xor2 for signal <xor_52$xor0000> created at line 2606.
Found 1-bit register for signal <xor_53>.
Found 1-bit xor2 for signal <xor_53$xor0000> created at line 2608.
Found 1-bit register for signal <xor_54>.
Found 1-bit xor2 for signal <xor_54$xor0000> created at line 2610.
Found 1-bit register for signal <xor_55>.
Found 1-bit xor2 for signal <xor_55$xor0000> created at line 2612.
Found 1-bit register for signal <xor_56>.
Found 1-bit xor2 for signal <xor_56$xor0000> created at line 2614.
Found 1-bit register for signal <xor_57>.
Found 1-bit xor2 for signal <xor_57$xor0000> created at line 2616.
Found 1-bit register for signal <xor_58>.
Found 1-bit xor2 for signal <xor_58$xor0000> created at line 2618.
Found 1-bit register for signal <xor_59>.
Found 1-bit xor2 for signal <xor_59$xor0000> created at line 2620.
Found 1-bit register for signal <xor_6>.
Found 1-bit xor2 for signal <xor_6$xor0000> created at line 2514.
Found 1-bit register for signal <xor_60>.
Found 1-bit xor2 for signal <xor_60$xor0000> created at line 2622.
Found 1-bit register for signal <xor_61>.
Found 1-bit xor2 for signal <xor_61$xor0000> created at line 2624.
Found 1-bit register for signal <xor_62>.
Found 1-bit xor2 for signal <xor_62$xor0000> created at line 2626.
Found 1-bit register for signal <xor_63>.
Found 1-bit xor2 for signal <xor_63$xor0000> created at line 2628.
Found 1-bit register for signal <xor_64>.
Found 1-bit xor2 for signal <xor_64$xor0000> created at line 2630.
Found 1-bit register for signal <xor_65>.
Found 1-bit xor2 for signal <xor_65$xor0000> created at line 2632.
Found 1-bit register for signal <xor_66>.
Found 1-bit xor2 for signal <xor_66$xor0000> created at line 2634.
Found 1-bit register for signal <xor_67>.
Found 1-bit xor2 for signal <xor_67$xor0000> created at line 2636.
Found 1-bit register for signal <xor_68>.
Found 1-bit xor2 for signal <xor_68$xor0000> created at line 2638.
Found 1-bit register for signal <xor_69>.
Found 1-bit xor2 for signal <xor_69$xor0000> created at line 2640.
Found 1-bit register for signal <xor_7>.
```

```
    Found 1-bit xor2 for signal <xor_7$xor0000> created at line 2516.
    Found 1-bit register for signal <xor_70>.
    Found 1-bit xor2 for signal <xor_70$xor0000> created at line 2642.
    Found 1-bit register for signal <xor_71>.
    Found 1-bit xor2 for signal <xor_71$xor0000> created at line 2644.
    Found 1-bit register for signal <xor_72>.
    Found 1-bit xor2 for signal <xor_72$xor0000> created at line 2646.
    Found 1-bit register for signal <xor_73>.
    Found 1-bit xor2 for signal <xor_73$xor0000> created at line 2648.
    Found 1-bit register for signal <xor_74>.
    Found 1-bit xor2 for signal <xor_74$xor0000> created at line 2650.
    Found 1-bit register for signal <xor_75>.
    Found 1-bit xor2 for signal <xor_75$xor0000> created at line 2652.
    Found 1-bit register for signal <xor_76>.
    Found 1-bit xor2 for signal <xor_76$xor0000> created at line 2654.
    Found 1-bit register for signal <xor_77>.
    Found 1-bit xor2 for signal <xor_77$xor0000> created at line 2656.
    Found 1-bit register for signal <xor_78>.
    Found 1-bit xor2 for signal <xor_78$xor0000> created at line 2658.
    Found 1-bit register for signal <xor_79>.
    Found 1-bit xor2 for signal <xor_79$xor0000> created at line 2660.
    Found 1-bit register for signal <xor_8>.
    Found 1-bit xor2 for signal <xor_8$xor0000> created at line 2518.
    Found 1-bit register for signal <xor_80>.
    Found 1-bit xor2 for signal <xor_80$xor0000> created at line 2662.
    Found 1-bit register for signal <xor_81>.
    Found 1-bit xor2 for signal <xor_81$xor0000> created at line 2664.
    Found 1-bit register for signal <xor_82>.
    Found 1-bit xor2 for signal <xor_82$xor0000> created at line 2666.
    Found 1-bit register for signal <xor_83>.
    Found 1-bit xor2 for signal <xor_83$xor0000> created at line 2668.
    Found 1-bit register for signal <xor_84>.
    Found 1-bit xor2 for signal <xor_84$xor0000> created at line 2670.
    Found 1-bit register for signal <xor_85>.
    Found 1-bit xor2 for signal <xor_85$xor0000> created at line 2672.
    Found 1-bit register for signal <xor_86>.
    Found 1-bit xor2 for signal <xor_86$xor0000> created at line 2674.
    Found 1-bit register for signal <xor_87>.
    Found 1-bit xor2 for signal <xor_87$xor0000> created at line 2676.
    Found 1-bit register for signal <xor_88>.
    Found 1-bit xor2 for signal <xor_88$xor0000> created at line 2678.
    Found 1-bit register for signal <xor_89>.
    Found 1-bit xor2 for signal <xor_89$xor0000> created at line 2680.
    Found 1-bit register for signal <xor_9>.
    Found 1-bit xor2 for signal <xor_9$xor0000> created at line 2520.
    Found 1-bit register for signal <xor_90>.
    Found 1-bit xor2 for signal <xor_90$xor0000> created at line 2682.
    Found 1-bit register for signal <xor_91>.
    Found 1-bit xor2 for signal <xor_91$xor0000> created at line 2684.
    Found 1-bit register for signal <xor_92>.
    Found 1-bit xor2 for signal <xor_92$xor0000> created at line 2686.
    Found 1-bit register for signal <xor_93>.
    Found 1-bit xor2 for signal <xor_93$xor0000> created at line 2688.
    Found 1-bit register for signal <xor_94>.
    Found 1-bit xor2 for signal <xor_94$xor0000> created at line 2690.
    Found 1-bit register for signal <xor_95>.
    Found 1-bit xor2 for signal <xor_95$xor0000> created at line 2692.
    Found 1-bit register for signal <xor_96>.
    Found 1-bit xor2 for signal <xor_96$xor0000> created at line 2694.
    Found 1-bit register for signal <xor_97>.
    Found 1-bit xor2 for signal <xor_97$xor0000> created at line 2696.
    Found 1-bit register for signal <xor_98>.
    Found 1-bit xor2 for signal <xor_98$xor0000> created at line 2698.
    Found 1-bit register for signal <xor_99>.
    Found 1-bit xor2 for signal <xor_99$xor0000> created at line 2700.
    Summary:
  inferred   1 Finite State Machine(s).
  inferred 162 ROM(s).
  inferred 2405 D-type flip-flop(s).
  inferred   1 Adder/Subtractor(s).
  inferred   3 Comparator(s).
  inferred 160 Multiplexer(s).
Unit <private_matrix_t> synthesized.


Synthesizing Unit <master_rom>.
    Related source file is "/home/stig/Documents/mqq/VHDL/master_rom.vhd".
WARNING:Xst:646 - Signal <enable_out> is assigned but never used. This unconnected signal will be
    trimmed during the optimization process.
    Found 32x3-bit ROM for signal <from_ctrl$rom0000> created at line 1130.
    Register <from_rom_6> equivalent to <from_rom_1> has been removed
    Register <from_rom_8> equivalent to <from_rom_7> has been removed
    Found 1024x30-bit ROM for signal <address_bus$rom0000>.
    Found 1-bit register for signal <en_out>.
    Found 5-bit register for signal <data_bus>.
    Found 5-bit 8-to-1 multiplexer for signal <data_bus$mux0001> created at line 11422.
    Found 3-bit register for signal <from_ctrl>.
    Found 5-bit register for signal <from_rom_1>.
    Found 5-bit register for signal <from_rom_2>.
    Found 5-bit register for signal <from_rom_3>.
    Found 5-bit register for signal <from_rom_4>.
    Found 5-bit register for signal <from_rom_5>.
    Found 5-bit register for signal <from_rom_7>.
    Summary:
  inferred   2 ROM(s).
```

```
    inferred  39 D-type flip-flop(s).
    inferred   5 Multiplexer(s).
Unit <master_rom> synthesized.


Synthesizing Unit <sequencer>.
    Related source file is "/home/stig/Documents/mqq/VHDL/sequencer.vhd".
WARNING:Xst:646 - Signal <en_out_mux> is assigned but never used. This unconnected signal will be
     trimmed during the optimization process.
WARNING:Xst:646 - Signal <en_out_mar> is assigned but never used. This unconnected signal will be
     trimmed during the optimization process.
    Found finite state machine <FSM_1> for signal <state>.
    -----------------------------------------------------------------------
    | States             | 7                                              |
    | Transitions        | 9                                              |
    | Inputs             | 2                                              |
    | Outputs            | 14                                             |
    | Clock              | clk (rising_edge)                              |
    | Clock enable       | en_in (positive)                               |
    | Reset              | reset (positive)                               |
    | Reset type         | synchronous                                    |
    | Reset State        | idle                                           |
    | Power Up State     | idle                                           |
    | Encoding           | automatic                                      |
    | Implementation     | LUT                                            |
    -----------------------------------------------------------------------
    Found 1-bit register for signal <en_out>.
    Found 160-bit register for signal <output_seq>.
    Found 5-bit up counter for signal <counter_1>.
    Found 5-bit up counter for signal <counter_2>.
    Found 5-bit comparator greatequal for signal <counter_2$cmp_ge0000> created at line 310.
    Found 5-bit register for signal <Mtridata_mux31_out> created at line 144.
    Found 1-bit register for signal <Mtrien_mux31_out> created at line 144.
    Found 5-bit register for signal <mux2_out>.
    Found 5-bit tristate buffer for signal <mux31_out>.
    Found 160-bit register for signal <reg_in>.
    Found 160-bit register for signal <reg_out>.
    Found 1-bit register for signal <sel>.
    Found 5-bit register for signal <sync_mr>.
    Found 10-bit register for signal <to_master>.
    Summary:
  inferred   1 Finite State Machine(s).
  inferred   2 Counter(s).
  inferred 508 D-type flip-flop(s).
  inferred   1 Comparator(s).
  inferred   5 Tristate(s).
Unit <sequencer> synthesized.


Synthesizing Unit <decryption>.
    Related source file is "/home/stig/Documents/mqq/VHDL/decryption.vhd".
WARNING:Xst:647 - Input <inputs> is never used. This port will be preserved and left unconnected if it
     belongs to a top-level block or it belongs to a sub-block and the hierarchy of this sub-block is
     preserved.
    Found finite state machine <FSM_2> for signal <state>.
    -----------------------------------------------------------------------
    | States             | 10                                             |
    | Transitions        | 15                                             |
    | Inputs             | 5                                              |
    | Outputs            | 15                                             |
    | Clock              | clk (rising_edge)                              |
    | Clock enable       | en_in (positive)                               |
    | Reset              | reset (positive)                               |
    | Reset type         | synchronous                                    |
    | Reset State        | idle                                           |
    | Power Up State     | idle                                           |
    | Encoding           | automatic                                      |
    | Implementation     | LUT                                            |
    -----------------------------------------------------------------------
    Found 160-bit register for signal <outputs>.
    Found 5-bit up counter for signal <count_s>.
    Found 5-bit register for signal <count_t>.
    Found 5-bit adder for signal <count_t$addsub0000> created at line 225.
    Found 160-bit register for signal <dec_input>.
    Found 160-bit register for signal <dec_output>.
    Found 13-bit register for signal <dob_vector_in>.
    Found 1-bit register for signal <en_in_dob>.
    Found 1-bit register for signal <en_in_pm_s>.
    Found 1-bit register for signal <en_in_pm_t>.
    Found 1-bit register for signal <en_in_seq>.
    Found 160-bit register for signal <seq_in>.
    Found 5-bit register for signal <shift_pms>.
    Found 5x3-bit multiplier for signal <shift_pms$mult0000> created at line 280.
    Found 5-bit register for signal <shift_pmt>.
    Found 5x3-bit multiplier for signal <shift_pmt$mult0000> created at line 223.
    Summary:
  inferred   1 Finite State Machine(s).
  inferred   1 Counter(s).
  inferred 672 D-type flip-flop(s).
  inferred   1 Adder/Subtractor(s).
  inferred   2 Multiplier(s).
Unit <decryption> synthesized.


=========================================================================
```

```
HDL Synthesis Report

Macro Statistics
# ROMs                                                 : 327
 1024x30-bit ROM                                       : 1
 32x3-bit ROM                                          : 1
 32x5-bit ROM                                          : 324
 8192x13-bit ROM                                       : 1
# Multipliers                                          : 2
 5x3-bit multiplier                                    : 2
# Adders/Subtractors                                   : 3
 3-bit adder                                           : 2
 5-bit adder                                           : 1
# Counters                                             : 3
 5-bit up counter                                      : 3
# Registers                                            : 2362
 1-bit register                                        : 1636
 13-bit register                                       : 1
 160-bit register                                      : 5
 3-bit register                                        : 3
 5-bit register                                        : 717
# Comparators                                          : 7
 3-bit comparator less                                 : 2
 5-bit comparator equal                                : 4
 5-bit comparator greatequal                           : 1
# Multiplexers                                         : 321
 1-bit 5-to-1 multiplexer                              : 320
 5-bit 8-to-1 multiplexer                              : 1
# Tristates                                            : 1
 5-bit tristate buffer                                 : 1
# Xors                                                 : 640
 1-bit xor2                                            : 640

=========================================================================

=========================================================================
*                            Advanced HDL Synthesis                     *
=========================================================================

Analyzing FSM <FSM_2> for best encoding.
Optimizing FSM <state/FSM> on signal <state[1:4]> with sequential encoding.
-------------------------------
 State           | Encoding
-------------------------------
 idle            | 0000
 send_to_pm_t    | 0001
 recv_from_pm_t  | 0010
 send_to_dob     | 0011
 recv_from_dob   | 0100
 send_to_seq     | 0101
 recv_from_seq   | 0110
 send_to_pm_s    | 0111
 recv_from_pm_s  | 1000
 send            | 1001
-------------------------------
Analyzing FSM <FSM_1> for best encoding.
Optimizing FSM <SEQ/state/FSM> on signal <state[1:3]> with gray encoding.
-------------------------------
 State           | Encoding
-------------------------------
 idle            | 000
 mux2_sel        | 001
 send_to_master  | 111
 recv_mr         | 110
 mux31_sel       | 011
 sync            | 010
 push            | 101
-------------------------------
Analyzing FSM <FSM_0> for best encoding.
Optimizing FSM <PM_T/state/FSM> on signal <state[1:3]> with gray encoding.
Optimizing FSM <PM_S/state/FSM> on signal <state[1:3]> with gray encoding.
---------------------
 State     | Encoding
---------------------
 idle      | 000
 andop     | 001
 xoring    | 011
 sync      | 010
 push_out  | 110
---------------------
Loading device for application Rf_Device from file '5vfx70t.nph' in environment /opt/Xilinx/10.1/ISE.

Synthesizing (advanced) Unit <decryption>.
  Found pipelined multiplier on signal <shift_pmt_mult0000>:
    - 1 pipeline level(s) found in a register on signal <count_t>.
    Pushing register(s) into the multiplier macro.
INFO:Xst:2385 - HDL ADVISOR - You can improve the performance of the multiplier
     Mmult_shift_pms_mult0000 by adding 2 register level(s).
INFO:Xst:2385 - HDL ADVISOR - You can improve the performance of the multiplier
     Mmult_shift_pmt_mult0000 by adding 2 register level(s).
INFO:Xst:2385 - HDL ADVISOR - You can improve the performance of the multiplier
     Mmult_shift_pms_mult0000 by adding 2 register level(s).
Unit <decryption> synthesized (advanced).

Synthesizing (advanced) Unit <dobbertin_rom>.
```

```
INFO:Xst - Currently unable to implement the ROM <Mrom_data_bus_mux0000> as a read-only block RAM.
     Please look for coming software updates.
Unit <dobbertin_rom> synthesized (advanced).

Synthesizing (advanced) Unit <master_rom>.
INFO:Xst - In order to maximize performance and save block RAM resources, the small ROM <
     Mrom_from_ctrl_rom0000> will be implemented on LUT. If you want to force its implementation on
     block, use option/constraint rom_style.
INFO:Xst - Currently unable to implement the ROM <Mrom_address_bus_rom0000> as a read-only block RAM.
     Please look for coming software updates.
Unit <master_rom> synthesized (advanced).
WARNING:Xst:1710 - FF/Latch <dec_input_117> (without init value) has a constant value of 0 in block <
     decryption>. This FF/Latch will be trimmed during the optimization process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_116> (without init value) has a
      constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_111> (without init value) has a
      constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_110> (without init value) has a
      constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_107> (without init value) has a
      constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_106> (without init value) has a
      constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_104> (without init value) has a
      constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_101> (without init value) has a
      constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_100> (without init value) has a
      constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_97> (without init value) has a
     constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_94> (without init value) has a
     constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_93> (without init value) has a
     constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_90> (without init value) has a
     constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_87> (without init value) has a
     constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_85> (without init value) has a
     constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_84> (without init value) has a
     constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_79> (without init value) has a
     constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_158> (without init value) has a
      constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_157> (without init value) has a
      constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_154> (without init value) has a
      constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_151> (without init value) has a
      constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_149> (without init value) has a
      constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_148> (without init value) has a
      constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_143> (without init value) has a
      constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_142> (without init value) has a
      constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_139> (without init value) has a
      constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_138> (without init value) has a
      constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_136> (without init value) has a
      constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
```

```
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_133> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_132> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_129> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_126> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_125> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_122> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_119> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_37> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_36> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_33> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_30> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_29> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_26> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_23> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_21> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_20> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_15> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_14> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_11> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_10> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_8> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_5> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_4> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_1> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_78> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_75> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_74> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_72> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_69> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_68> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_65> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
    process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_62> (without init value) has a
    constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
```

```
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_61> (without init value) has a
     constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_58> (without init value) has a
     constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_55> (without init value) has a
     constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_53> (without init value) has a
     constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_52> (without init value) has a
     constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_47> (without init value) has a
     constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_46> (without init value) has a
     constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_43> (without init value) has a
     constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_42> (without init value) has a
     constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
WARNING:Xst:1895 - Due to other FF/Latch trimming, FF/Latch <dec_input_40> (without init value) has a
     constant value of 0 in block <decryption>. This FF/Latch will be trimmed during the optimization
     process.
INFO:Xst:2261 - The FF/Latch <dec_input_0> in Unit <decryption> is equivalent to the following 89 FFs/
     Latches, which will be removed : <dec_input_2> <dec_input_3> <dec_input_6> <dec_input_7> <
     dec_input_9> <dec_input_12> <dec_input_13> <dec_input_16> <dec_input_17> <dec_input_18> <
     dec_input_19> <dec_input_22> <dec_input_24> <dec_input_25> <dec_input_27> <dec_input_28> <
     dec_input_31> <dec_input_32> <dec_input_34> <dec_input_35> <dec_input_38> <dec_input_39> <
     dec_input_41> <dec_input_44> <dec_input_45> <dec_input_48> <dec_input_49> <dec_input_50> <
     dec_input_51> <dec_input_54> <dec_input_56> <dec_input_57> <dec_input_59> <dec_input_60> <
     dec_input_63> <dec_input_64> <dec_input_66> <dec_input_67> <dec_input_70> <dec_input_71> <
     dec_input_73> <dec_input_76> <dec_input_77> <dec_input_80> <dec_input_81> <dec_input_82> <
     dec_input_83> <dec_input_86> <dec_input_88> <dec_input_89> <dec_input_91> <dec_input_92> <
     dec_input_95> <dec_input_96> <dec_input_98> <dec_input_99> <dec_input_102> <dec_input_103> <
     dec_input_105>
    <dec_input_108> <dec_input_109> <dec_input_112> <dec_input_113> <dec_input_114> <dec_input_115> <
         dec_input_118> <dec_input_120> <dec_input_121> <dec_input_123> <dec_input_124> <dec_input_127>
         <dec_input_128> <dec_input_130> <dec_input_131> <dec_input_134> <dec_input_135> <dec_input_137>
          <dec_input_140> <dec_input_141> <dec_input_144> <dec_input_145> <dec_input_146> <dec_input_147
         > <dec_input_150> <dec_input_152> <dec_input_153> <dec_input_155> <dec_input_156> <
         dec_input_159>
WARNING:Xst:2677 - Node <en_out> of sequential type is unconnected in block <MA_R>.

=========================================================================
Advanced HDL Synthesis Report

Macro Statistics
# ROMs                                          : 327
 1024x30-bit ROM                                : 1
 32x3-bit ROM                                   : 1
 32x5-bit ROM                                   : 324
 8192x13-bit ROM                                : 1
# Multipliers                                   : 2
 5x3-bit multiplier                             : 1
 5x3-bit registered multiplier                  : 1
# Adders/Subtractors                            : 3
 3-bit adder                                    : 2
 5-bit adder                                    : 1
# Counters                                      : 3
 5-bit up counter                               : 3
# Registers                                     : 5897
 Flip-Flops                                     : 5897
# Comparators                                   : 7
 3-bit comparator less                          : 2
 5-bit comparator equal                         : 4
 5-bit comparator greatequal                    : 1
# Multiplexers                                  : 325
 1-bit 5-to-1 multiplexer                       : 320
 1-bit 8-to-1 multiplexer                       : 5
# Xors                                          : 640
 1-bit xor2                                     : 640

=========================================================================

=========================================================================
*                            Low Level Synthesis                        *
=========================================================================
INFO:Xst:2146 - In block <private_matrix_t>, ROM <Mrom_from_rom_3_rom0000> <Mrom_from_rom_1_rom0000> <
     Mrom_from_rom_2_rom0000> <Mrom_from_rom_6_rom0000> <Mrom_from_rom_4_rom0000> <
     Mrom_from_rom_5_rom0000> <Mrom_from_rom_9_rom0000> <Mrom_from_rom_7_rom0000> <
     Mrom_from_rom_8_rom0000> <Mrom_from_rom_10_rom0000> <Mrom_from_rom_11_rom0000> <
     Mrom_from_rom_14_rom0000> <Mrom_from_rom_12_rom0000> <Mrom_from_rom_13_rom0000> <
     Mrom_from_rom_20_rom0000> <Mrom_from_rom_15_rom0000> <Mrom_from_rom_22_rom0000> <
     Mrom_from_rom_16_rom0000> <Mrom_from_rom_21_rom0000> <Mrom_from_rom_17_rom0000> <
     Mrom_from_rom_23_rom0000> <Mrom_from_rom_19_rom0000> <Mrom_from_rom_18_rom0000> <
     Mrom_from_rom_24_rom0000> <Mrom_from_rom_25_rom0000> <Mrom_from_rom_30_rom0000> <
     Mrom_from_rom_32_rom0000> <Mrom_from_rom_31_rom0000> <Mrom_from_rom_26_rom0000> <
```

```
     Mrom_from_rom_27_rom0000> <Mrom_from_rom_28_rom0000> <Mrom_from_rom_34_rom0000> <
     Mrom_from_rom_33_rom0000> <Mrom_from_rom_29_rom0000> <Mrom_from_rom_35_rom0000> <
     Mrom_from_rom_40_rom0000> <Mrom_from_rom_42_rom0000> <Mrom_from_rom_41_rom0000> <
     Mrom_from_rom_36_rom0000> <Mrom_from_rom_37_rom0000> <Mrom_from_rom_38_rom0000> <
     Mrom_from_rom_44_rom0000> <Mrom_from_rom_43_rom0000> <Mrom_from_rom_39_rom0000> <
     Mrom_from_rom_45_rom0000> <Mrom_from_rom_50_rom0000> <Mrom_from_rom_52_rom0000> <
     Mrom_from_rom_51_rom0000> <Mrom_from_rom_46_rom0000> <Mrom_from_rom_47_rom0000> <
     Mrom_from_rom_48_rom0000> <Mrom_from_rom_54_rom0000> <Mrom_from_rom_53_rom0000> <
     Mrom_from_rom_49_rom0000> <Mrom_from_rom_55_rom0000> <Mrom_from_rom_60_rom0000> <
     Mrom_from_rom_62_rom0000> <Mrom_from_rom_61_rom0000> <Mrom_from_rom_56_rom0000> <
     Mrom_from_rom_57_rom0000> <Mrom_from_rom_58_rom0000> <Mrom_from_rom_64_rom0000> <
     Mrom_from_rom_63_rom0000> <Mrom_from_rom_59_rom0000> <Mrom_from_rom_65_rom0000> <
     Mrom_from_rom_70_rom0000> <Mrom_from_rom_72_rom0000> <Mrom_from_rom_66_rom0000> <
     Mrom_from_rom_71_rom0000> <Mrom_from_rom_67_rom0000> <Mrom_from_rom_68_rom0000> <
     Mrom_from_rom_74_rom0000> <Mrom_from_rom_73_rom0000> <Mrom_from_rom_69_rom0000> <
     Mrom_from_rom_75_rom0000> <Mrom_from_rom_80_rom0000> <Mrom_from_rom_82_rom0000> <
     Mrom_from_rom_76_rom0000> <Mrom_from_rom_81_rom0000> <Mrom_from_rom_77_rom0000> <
     Mrom_from_rom_78_rom0000> <Mrom_from_rom_84_rom0000> <Mrom_from_rom_83_rom0000> <
     Mrom_from_rom_79_rom0000> <Mrom_from_rom_86_rom0000> <Mrom_from_rom_85_rom0000> <
     Mrom_from_rom_90_rom0000> <Mrom_from_rom_87_rom0000> <Mrom_from_rom_91_rom0000> <
     Mrom_from_rom_92_rom0000> <Mrom_from_rom_88_rom0000> <Mrom_from_rom_93_rom0000> <
     Mrom_from_rom_95_rom0000> <Mrom_from_rom_89_rom0000> <Mrom_from_rom_94_rom0000> <
     Mrom_from_rom_96_rom0000> <Mrom_from_rom_97_rom0000> <Mrom_from_rom_100_rom0000> <
     Mrom_from_rom_98_rom0000> <Mrom_from_rom_99_rom0000> <Mrom_next_state_rom0000> <
     Mrom_from_rom_101_rom0000> <Mrom_from_rom_104_rom0000> <Mrom_from_rom_102_rom0000> <
     Mrom_from_rom_103_rom0000> <Mrom_from_rom_110_rom0000> <Mrom_from_rom_105_rom0000> <
     Mrom_from_rom_107_rom0000> <Mrom_from_rom_106_rom0000> <Mrom_from_rom_111_rom0000> <
     Mrom_from_rom_112_rom0000> <Mrom_from_rom_113_rom0000> <Mrom_from_rom_114_rom0000> <
     Mrom_from_rom_108_rom0000> <Mrom_from_rom_109_rom0000> <Mrom_from_rom_120_rom0000> <
     Mrom_from_rom_115_rom0000> <Mrom_from_rom_117_rom0000> <Mrom_from_rom_116_rom0000> <
     Mrom_from_rom_121_rom0000> <Mrom_from_rom_122_rom0000> <Mrom_from_rom_123_rom0000> <
     Mrom_from_rom_124_rom0000> <Mrom_from_rom_118_rom0000> <Mrom_from_rom_119_rom0000> <
     Mrom_from_rom_130_rom0000> <Mrom_from_rom_125_rom0000> <Mrom_from_rom_127_rom0000> <
     Mrom_from_rom_126_rom0000> <Mrom_from_rom_131_rom0000> <Mrom_from_rom_132_rom0000> <
     Mrom_from_rom_133_rom0000> <Mrom_from_rom_134_rom0000> <Mrom_from_rom_128_rom0000> <
     Mrom_from_rom_129_rom0000> <Mrom_from_rom_140_rom0000> <Mrom_from_rom_135_rom0000> <
     Mrom_from_rom_137_rom0000> <Mrom_from_rom_136_rom0000> <Mrom_from_rom_141_rom0000> <
     Mrom_from_rom_142_rom0000> <Mrom_from_rom_143_rom0000> <Mrom_from_rom_144_rom0000> <
     Mrom_from_rom_138_rom0000> <Mrom_from_rom_139_rom0000> <Mrom_from_rom_150_rom0000> <
     Mrom_from_rom_145_rom0000> <Mrom_from_rom_147_rom0000> <Mrom_from_rom_146_rom0000> <
     Mrom_from_rom_151_rom0000> <Mrom_from_rom_152_rom0000> <Mrom_from_rom_153_rom0000> <
     Mrom_from_rom_154_rom0000> <Mrom_from_rom_148_rom0000> <Mrom_from_rom_149_rom0000> <
     Mrom_from_rom_160_rom0000> <Mrom_from_rom_155_rom0000> <Mrom_from_rom_158_rom0000> <
     Mrom_from_rom_156_rom0000> <Mrom_from_rom_157_rom0000> <Mrom_from_rom_159_rom0000> are equivalent,
      XST will keep only <Mrom_from_rom_3_rom0000>.
INFO:Xst:2261 - The FF/Latch <count_t_3> in Unit <decryption> is equivalent to the following FF/Latch,
      which will be removed : <Mmult_shift_pmt_mult0000_1>
INFO:Xst:2261 - The FF/Latch <count_t_4> in Unit <decryption> is equivalent to the following FF/Latch,
      which will be removed : <Mmult_shift_pmt_mult0000_0>
INFO:Xst:2261 - The FF/Latch <count_t_0> in Unit <decryption> is equivalent to the following FF/Latch,
      which will be removed : <Mmult_shift_pmt_mult0000_4>
INFO:Xst:2261 - The FF/Latch <count_t_1> in Unit <decryption> is equivalent to the following FF/Latch,
      which will be removed : <Mmult_shift_pmt_mult0000_3>
INFO:Xst:2261 - The FF/Latch <count_t_2> in Unit <decryption> is equivalent to the following FF/Latch,
      which will be removed : <Mmult_shift_pmt_mult0000_2>
WARNING:Xst:2042 - Unit sequencer: 5 internal tristates are replaced by logic (pull-up yes): mux31_out
     <0>, mux31_out<1>, mux31_out<2>, mux31_out<3>, mux31_out<4>.

Optimizing unit <decryption> ...

Optimizing unit <private_matrix_t> ...
WARNING:Xst:1293 - FF/Latch <count_xor_2> has a constant value of 0 in block <private_matrix_t>. This
     FF/Latch will be trimmed during the optimization process.
WARNING:Xst:1293 - FF/Latch <count_xor_2> has a constant value of 0 in block <private_matrix_t>. This
     FF/Latch will be trimmed during the optimization process.

Optimizing unit <master_rom> ...

Optimizing unit <sequencer> ...
WARNING:Xst:2677 - Node <SEQ/MA_R/en_out> of sequential type is unconnected in block <decryption>.

Mapping all equations...
Building and optimizing final netlist ...
Found area constraint ratio of 100 (+ 5) on block decryption, actual ratio is 21.
FlipFlop count_s_1 has been replicated 1 time(s)

Final Macro Processing ...

=========================================================================
Final Register Report

Macro Statistics
# Registers                                          : 5910
 Flip-Flops                                          : 5910

=========================================================================

=========================================================================
*                            Partition Report                           *
=========================================================================

Partition Implementation Status
-------------------------------

  No Partitions were found in this design.
```

```
=========================================================================
*                            Final Report                               *
=========================================================================
Final Results
RTL Top Level Output File Name     : decryption.ngr
Top Level Output File Name         : decryption
Output Format                      : NGC
Optimization Goal                  : Speed
Keep Hierarchy                     : NO

Design Statistics
# IOs                              : 324

Cell Usage :
# BELS                             : 9732
#      GND                         : 1
#      INV                         : 3
#      LUT1                        : 4
#      LUT2                        : 1678
#      LUT3                        : 529
#      LUT4                        : 40
#      LUT5                        : 1991
#      LUT6                        : 3457
#      MUXCY                       : 10
#      MUXF7                       : 1472
#      MUXF8                       : 536
#      VCC                         : 1
#      XORCY                       : 10
# FlipFlops/Latches                : 5910
#      FD                          : 1
#      FDE                         : 4330
#      FDRE                        : 1579
# Clock Buffers                    : 1
#      BUFGP                       : 1
# IO Buffers                       : 163
#      IBUF                        : 2
#      OBUF                        : 161
=========================================================================

Device utilization summary:
---------------------------

Selected Device : 5vfx70tff1136-1


Slice Logic Utilization:
 Number of Slice Registers:             5910  out of  44800     13%
 Number of Slice LUTs:                  7702  out of  44800     17%
    Number used as Logic:               7702  out of  44800     17%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:    8839
   Number with an unused Flip Flop:     2929  out of   8839     33%
   Number with an unused LUT:           1137  out of   8839     12%
   Number of fully used LUT-FF pairs:   4773  out of   8839     53%
   Number of unique control sets:         64

IO Utilization:
 Number of IOs:                          324
 Number of bonded IOBs:                  164  out of    640     25%

Specific Feature Utilization:
 Number of BUFG/BUFGCTRLs:                 1  out of     32      3%

---------------------------
Partition Resource Summary:
---------------------------

  No Partitions were found in this design.

---------------------------


=========================================================================
TIMING REPORT

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.
      FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT
      GENERATED AFTER PLACE-and-ROUTE.

Clock Information:
------------------
+------------------------------------+------------------------+-------+
| Clock Signal                       | Clock buffer(FF name)  | Load  |
+------------------------------------+------------------------+-------+
| clk                                | BUFGP                  | 5910  |
+------------------------------------+------------------------+-------+

Asynchronous Control Signals Information:
-----------------------------------------
No asynchronous control signals found in this design
```

```
Timing Summary:
---------------
Speed Grade: -1

   Minimum period: 4.673ns (Maximum Frequency: 214.000MHz)
   Minimum input arrival time before clock: 2.766ns
   Maximum output required time after clock: 3.259ns
   Maximum combinational path delay: No path found

Timing Detail:
--------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default period analysis for Clock 'clk'
  Clock period: 4.673ns (frequency: 214.000MHz)
  Total number of paths / destination ports: 68641 / 11805
-------------------------------------------------------------------------
Delay:               4.673ns (Levels of Logic = 10)
  Source:            count_s_2 (FF)
  Destination:       shift_pms_0 (FF)
  Source Clock:      clk rising
  Destination Clock: clk rising

  Data Path: count_s_2 to shift_pms_0
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDRE:C->Q            13   0.471   0.642  count_s_2 (count_s_2)
     LUT2:I0->O            1   0.094   0.000  Mmult_shift_pms_mult0000_Madd_lut<2> (
          Mmult_shift_pms_mult0000_Madd_lut<2>)
     MUXCY:S->O            1   0.372   0.000  Mmult_shift_pms_mult0000_Madd_cy<2> (
          Mmult_shift_pms_mult0000_Madd_cy<2>)
     MUXCY:CI->O           1   0.026   0.000  Mmult_shift_pms_mult0000_Madd_cy<3> (
          Mmult_shift_pms_mult0000_Madd_cy<3>)
     MUXCY:CI->O           1   0.026   0.000  Mmult_shift_pms_mult0000_Madd_cy<4> (
          Mmult_shift_pms_mult0000_Madd_cy<4>)
     MUXCY:CI->O           1   0.026   0.000  Mmult_shift_pms_mult0000_Madd_cy<5> (
          Mmult_shift_pms_mult0000_Madd_cy<5>)
     MUXCY:CI->O         100   0.254   0.620  Mmult_shift_pms_mult0000_Madd_cy<6> (
          Mmult_shift_pms_mult0000_Madd_cy<6>)
     LUT3:I2->O            1   0.094   0.710  shift_pms_mux0001<4>1938_SW2 (N92)
     LUT6:I3->O            1   0.094   0.576  shift_pms_mux0001<4>1938 (shift_pms_mux0001<4>1938)
     LUT4:I2->O            1   0.094   0.480  shift_pms_mux0001<4>1984 (shift_pms_mux0001<4>1984)
     LUT6:I5->O            1   0.094   0.000  shift_pms_mux0001<4>11355 (shift_pms_mux0001<4>)
     FDRE:D                   -0.018          shift_pms_4
    ----------------------------------------
    Total                      4.673ns (1.645ns logic, 3.028ns route)
                                       (35.2% logic, 64.8% route)

=========================================================================
Timing constraint: Default OFFSET IN BEFORE for Clock 'clk'
  Total number of paths / destination ports: 6593 / 6593
-------------------------------------------------------------------------
Offset:              2.766ns (Levels of Logic = 2)
  Source:            reset (PAD)
  Destination:       SEQ/counter_2_4 (FF)
  Destination Clock: clk rising

  Data Path: reset to SEQ/counter_2_4
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O           1782   0.818   1.283  reset_IBUF (reset_IBUF)
     LUT6:I0->O             5   0.094   0.358  SEQ/counter_2_not000111 (SEQ/counter_2_not0001)
     FDE:CE                     0.213          SEQ/counter_2_0
    ----------------------------------------
    Total                      2.766ns (1.125ns logic, 1.641ns route)
                                       (40.7% logic, 59.3% route)

=========================================================================
Timing constraint: Default OFFSET OUT AFTER for Clock 'clk'
  Total number of paths / destination ports: 160 / 160
-------------------------------------------------------------------------
Offset:              3.259ns (Levels of Logic = 1)
  Source:            outputs_159 (FF)
  Destination:       outputs<159> (PAD)
  Source Clock:      clk rising

  Data Path: outputs_159 to outputs<159>
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDRE:C->Q              1   0.471   0.336  outputs_159 (outputs_159)
     OBUF:I->O                  2.452          outputs_159_OBUF (outputs<159>)
    ----------------------------------------
    Total                      3.259ns (2.923ns logic, 0.336ns route)
                                       (89.7% logic, 10.3% route)

=========================================================================


Total REAL time to Xst completion: 693.00 secs
Total CPU time to Xst completion: 670.57 secs
```

```
-->


Total memory usage is 423868 kilobytes

Number of errors   :     0 (    0 filtered)
Number of warnings :    80 (    0 filtered)
Number of infos    :    17 (    0 filtered)
```