Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

Markus Rud

# Power and energy consumption in hardware implemented SPI master devices

Master's thesis in Electronics Systems Design and Innovation
Supervisor: Snorre Aunet
Co-supervisor: Øystein Moldsvor
June 2021

NTNU
Norwegian University of Science and Technology

Markus Rud

# Power and energy consumption in hardware implemented SPI master devices

Master's thesis in Electronics Systems Design and Innovation
Supervisor: Snorre Aunet
Co-supervisor: Øystein Moldsvor
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

NTNU
Norwegian University of
Science and Technology

## Abstract

This thesis presents an analysis of two different VHDL designs of the SPI master device implemented onto a model of a FPGA. The analysis is focused at power and energy consumption in the devices compared with provided functionality. The two devices differ in their design strategy where one is created as a simple design where only the required logic to conduct a SPI transmission are implemented. The second one is a more complex design where it is possible to adjust transmission parameters such as setup and hold time after implementation and a more complex interface to the controlling logic which controls the SPI masters. The complex implementation also implement two FIFO registers to store multiple messages during transmission and reception.

The conducted analysis is based upon different tests in order to give an understanding of which elements of a SPI master who impact the energy consumption in the device. These tests look into the impact of operating frequency, communication frequency, operation mode and alternation to the utilized logic. The designs are implemented onto a model of a FPGA using the development tool Vivado. The two designs are also power optimized using the build in power optimizer in Vivado.

The results from the analysis show that when implementing a SPI master, it is necessary with a trade of between functionality and energy consumption. The different implementations are analysed over a frequency span of 1 MHz to 15 MHz where it is seen that the complex master requires 27.2% more energy than the simple master on average. It is therefore seen that a higher complexity in the design requires more energy. The complex master utilize more than twice as much logic, but not twice as much energy, so the energy cost of added functionality is therefore heavily dependent on the switching activity in the added logic. The results also show that it is preferable to operate the tested SPI masters at the highest frequency possible within the tested frequencies since this gives the lowest energy consumption. This result is to some extend limited by the implementation method as the implementation of the SPI master onto a FPGA removes some potential benefits of operating the design at a low frequency such as smaller transistor sizes and lower operating voltage. It is also seen that the two SPI designs react relatively similar to adjustments to transmission parameters such as communication frequency and operating mode since their percentage energy difference with adjustments are approximately similar for both designs.

The analysis consist of some limitations. The SPI masters are analysed as standalone devices not connected to any controlling device which limits the energy analysis due to missing timing delays. The SPI masters are also relatively small designs so when implemented on a large FPGA compared to the designs, a large static power overhead is added which can hide the actual static power consumption for the designs themselves.

## Sammendrag

Denne oppgaven tar for seg en analyse av to ulike VHDL-design av master enheten på en SPI-buss implementert på en FPGA modell. Analysen er i hovedsak fokusert rundt effekt og energi forbruk sammenliknet med enhetenes funksjonalitet. De to enhetene er designet basert på ulike strategier hvor den første er designet som et simpelt design hvor kun den nødvendige logikken for å gjennomføre en SPI-overføring er implementert. Det andre designet er et mer komplekst design hvor det er mulig å modifisere overførings parametere etter implementasjon samt et mer komplekst grensesnitt inn mot kontrollogikken som styrer SPI masterne. Den komplekse implementasjonen implementerer også to FIFO registre for å mellomlagre meldinger under overføring og mottagelse.

Analysen er basert på ulike tester som er gjennomført med formål om å skape en forståelse av hvilke elementer i en SPI master som påvirker energiforbruket. Disse testene tar for seg påvirkningen fra driftsfrekvens, kommunikasjonsfrekvens, driftsmodus og modifikasjon av implementert logikk. Disse designene er implementert på en modell av en FPGA ved bruk av verktøyet Vivado. Designene er også optimalisert med tanke på effektforbruk ved bruk av den innebygde effekt-optimalisereren i Vivado.

Resultatene viser at ved implementasjon av en SPI master, er det nødvendig å vurdere behovet for funksjonalitet opp mot energi forbruk. De ulike implementasjonene er analysert over et frekvensområde fra 1 MHz til 15 MHz hvor man kan se at den komplekse masteren trenger gjennomsnittlig 27.2% mer energi enn den simple masteren. Dette viser at en høyere kompleksitet gir et høyere energiforbruk. Den komplekse masteren trenger mer enn dobbelt så mye logikk, men ikke dobbelt så mye energi. Dette viser at energikostnaden er svært avhengig av svitsje-aktiviteten i den ekstra logikken. Resultatene viser også at det er gunstig å drifte de to SPI-masterne på høyest mulig frekvens innenfor det testede frekvensområdet siden dette gir det laveste energiforbruket. Dette resultatet er til en viss grad begrenset av hvordan designene er implementert siden man ved å benytte en FPGA mister noen fordeler ved lav driftsfrekvens som mindre transistorstørrelser og lavere driftsspenning. De to designene responderer også relativt likt til modifikasjoner i overføringsparametere som ulik kommunikasjon frekvens og driftsmodus siden den prosentvise endringen etter modifikasjon er tilnærmet lik for begge designene.

Analysen består av enkelte begrensninger. SPI masterne er analysert som selvstendige enheter som ikke er koblet til en kontrollenhet. Dette begrenser energianalysen på grunn av manglende tidsforsinkelser i systemet. SPI masterne er også relativt små design sammenliknet med størrelsen på FPGAen så et stort statisk effektforbruk blir lagt til og kan skjule det faktiske statiske forbruket i selve designene.

## Acknowledgement

This master thesis is written as a finalization of the 5 year master degree program Electronics Systems Design and Innovation at the Norwegian University of Science and Technology (NTNU) in Trondheim. It has been 5 very interesting years where I have learned and experienced a lot both on and off campus.

I would like to thank my supervisors professor Snorre Aunet from NTNU and Øystein Moldsvor from Disruptive Technologies for their guidance throughout this project. They have helped me figuring out how to proceed with the research and pointed out when results and methods have seem weird.

Trondheim June 18, 2021

Markus Rud

# Contents

# List of Figures

# List of Tables

## Acronyms

**ASIC** Application-Specific Integrated Circuit. 8, 45, 54

**CPHA** Clock Phase. 11, 22, 35, 49

**CPOL** Clock Polarity. 11, 22, 35, 49

**FIFO** First In, First Out. i, ii, 17, 25, 37, 49

**FPGA** Field-Programmable Gate Array. i, ii, 2, 6, 19, 30, 44, 52, 54

**HDL** Hardware Description Language. 7, 14

**I/O** Input/Output. 2, 7, 19, 30, 44

**I2C** Inter-Integrated Circuit. 1, 54

**IoT** Internet of Things. 1

**IP** Intellectual Property. 2, 44

**LSB** Least Significant Bit. 16

**LUT** Look-Up Table. 7, 19, 31, 44

**MISO** Master In Slave Out. 10, 21

**MOSI** Master Out Slave In. 10, 21

**MSB** Most Significant Bit. 16

**RTL** Register Transfer Level. 7, 19, 30, 51

**SCLK** Serial Clock. 10, 20, 26, 46, 52

**SLOC** Source Lines Of Code. 30, 47

**SoC** System on Chip. 19

**SPI** Serial Peripheral Interface. i, ii, 1, 5, 10, 19, 26, 44, 52, 54

**SS** Slave Select. 10

**VHDL** VHSIC Hardware Description Language. i, ii, 7, 21

# 1 Introduction

One of the largest concerns when designing an electrical systems in the modern era is power and energy consumption. The requirements and expectations for electrical systems grow rapidly with the growing Internet of Things (IoT) era where an important element is a long life time for a device from a limited energy source such as a battery[1]. One estimate predicts that more than 41 billion IoT devices can exist by 2027[2] which gives large market opportunities for devices with a low energy consumption. The importance of energy management is not only limited to the IoT area, but also for all other electrical systems as well since energy consumption has a large environmental impact[3]. This creates a demand for devices with a low energy consumption which can have a long battery life time or low power draw.

An electrical system such as an IoT-device often consists of multiple subsystems where each subsystem has its own consumption of power and energy. Naturally it is the total energy consumption for the combined system who is of importance when trying to lower the energy consumption, but due to the combination of subsystems it can be beneficial to focus the effort in energy reduction to a subsystem-level. The different subsystems will often communicate between each other to exchange data and depending on the system, the energy consumption from such internal communication could be of a significant amount and an effort in energy reduction could be beneficial.

Multiple communication methods between subsystems are already defined as communication protocols and made publicly available. These differ with a large variety of communication concepts such as digital or analog, and wired or wireless where the different methods often includes different functionality and area of usage. This thesis is focused on digital wired communication mainly for internal use within an embedded system such as between a sensor and a microprocessor, but it still exists a variety of different protocols with different advantages and disadvantages within this area of usage. Examples of such protocols are SPI, I2C, MBus[1][4] and more. A new specialized communication method could naturally be created for each new system, but the utilization of a commonly used communication protocol could shorten design time and make it easier to include already created devices such as a sensor made by a different company into the new system.

When implementing a communication protocol into a system, the designer must choose whether the communication functionality should be provided by software or by dedicated hardware. For instance could the communication protocol be implemented in a general purpose processor as a part of the software, or it might be designed as specific modules with dedicated hardware just for communication. There are naturally advan-

[1]Some considerations regarding MBus are described in subsection 7.2 for readers especially interested in the development of this protocol

tages and disadvantages for each of the implementation strategies where one advantage of the hardware implementation is a tendency of a lower energy consumption than a software solution[5], but at the cost of extra area requirements.

With a large amount and variety in established communication protocols, it could be a difficult task to choose the proper one for a specific system. As presented in [6] the proper communication protocol heavily depends on system specifications and requirements such as energy consumption, available I/O, available devices and more. The communication protocol therefore has to be chosen on a case-to-case basis based upon the requirements. The specifications of a communication protocol often only specifies the external interface for how the subsystems should be connected and interact. This gives a large amount of freedom to the designer regarding the implementation of a communication module since as long as the module fulfills the external specifications, the implementation of the modules can be designed freely. This gives the possibility to implement communication modules with different complexity levels such as additional data handling or additional internal communication to a subsystem controlling the communication module. A simple implementation of the communication module where only the bare minimum in order to fulfill the external requirements is implemented might therefore use a lower amount of energy than a complex implementation where added functionality such as temporary storage or other features are implemented due to a simpler implementation.

During the design of a larger system, a designer could typically utilize a premade Intellectual Property (IP) to add functionality or speed up the design process where a communication module is an example of a typical IP to add to the system. Often it is not desired to largely modify such IPs or it might not be possible since they can be encrypted[7]. This creates a need to have an understanding of how the complexity level of the design impact different aspects such as energy consumption and area requirement in order to choose the proper IP or design to include in the system.

This thesis presents an analysis of the energy consumption for two different premade designs of the master device for the SPI protocol. One where only the minimum amount of required functionality in order to operate the external interface is implemented which can be considered a simple design and one with more internal functionality such as more feedback to the controlling subsystem and adjustment of transmission parameters and can therefore be considered a more complex design. These designs are implemented in hardware on a model of a FPGA using the development tool Vivado and their power and energy consumption are estimated and analysed. In order to understand how the differences between the implementations affect the power and energy consumption, multiple tests are conducted. These tests include alternating the operating and communication frequency, different operating modes, modifying the

internal hardware and power optimization. A look at already conducted studies on wired communication in embedded systems show that multiple studies compare the different communication protocols towards each other. However, the research done on how different implementations of the same protocol differ within the specifications are limited and is the reason for this study to be conducted.

The different tests are conducted over different operating frequencies ranging from 1 MHz to 15 MHz. The results from these tests show that the complex implementation requires an average of 27.2% more energy than the simple master at the different frequencies. The results also show that the two implementations respond relatively similarly to alternations in communication frequency and operating mode as the percentage energy change are relatively similar throughout the tests. However, due to the larger overall energy consumption for the complex master, the consequence of increase for the complex master is larger even if the percentage increase is similar. Lastly it is seen that performing power optimization on such a small design as a SPI master might have the opposite effect where the energy consumption increased with 5.2% and 1.3% for the simple and complex master respectively after power optimization when it would be more reasonable for the energy consumption to decrease. These results however has certain uncertainties based on the implementation method on the FPGA. The SPI masters are implemented as standalone devices on the FPGA which not gives a completely realistic environment for the designs due to missing control logic. The use of a FPGA is also seen to be unfortunate when the energy consumption of the SPI masters themselves are of interest since the FPGA has an additional energy consumption due to the re-configurable functionality of a FPGA.

This thesis is organized by first presenting some required background knowledge, information about the external interface of the SPI protocol and a description of the differences in the two SPI master implementations in section 2. In section 3 are the Vivado tool and the method for development and analysis presented together with details regarding the applied tests. The results from the different tests are presented in section 4 and discussed in section 5 together with an evaluation of the applied analysis method. Lastly the conclusion is presented in section 6 and some recommendation for future work and some information regarding the MBus protocol are given in section 7

# 2 Theoretical background

Some basic knowledge about electrical concepts and systems are expected from the reader, but specific details are presented in this section.

## 2.1 Energy and power consumption in electrical systems

The total energy $E$ of an electrical system or design-implementation is equal to the integral of the instantaneous power $P(t)$ over some time interval $T$ as shown in equation 2.1[8]. This equation leads to the following two options for reducing the total energy consumption in a system; either make the design use less power, and/or power the design for a shorter time interval. For instance can a design with a high power consumption still have a low energy consumption as long at the time period is short enough.

$$E = \int_0^T P(t)dt \tag{2.1}$$

The total power consumption $P_{total}$ for a system is a combination of two types of power dissipation as shown in equation 2.2. These are the dynamic power $P_{dynamic}$ and the static power $P_{static}$ where the total power is the sum of these.

$$P_{total} = P_{dynamic} + P_{static} \tag{2.2}$$

The average power consumption $P_{avg}$ in a system is shown in equation 2.3 and is given as the total energy consumption divided by the total time interval.

$$P_{avg} = \frac{E}{T} \tag{2.3}$$

### 2.1.1 Dynamic power consumption

The dynamic power consumption in a design is caused by changing signals and consist of both power from switching and from "short-circuits" as shown in equation 2.4. The short circuit power is generated in the brief moments when both the pMOS and nMOS stacks are partially on. This happens when the transistors in the design switches and a path is created directly between the supply voltage $V_{DD}$ and ground $GND$[8].

$$P_{dynamic} = P_{switching} + P_{shortcircuit} \tag{2.4}$$

The main contributor to the dynamic power consumption is the switching power as it normally contributes to more than 90% of the total dynamic power[8]. The switching

power is again affected by a variety of factors as shown in equation 2.5 where $\alpha$ is the activity factor, $C$ is the capacitance and $f$ is the switching frequency. The activity factor is seen as the probability for a node in the circuit switches from 0 to 1.

$$P_{switching} = \alpha C V_{DD}^2 f \tag{2.5}$$

### 2.1.2 Static power consumption

As mentioned the other element contributing to the total power dissipation in a design is the static power consumption. The equation for this consumption is shown in equation 2.6 where $I_{leakage}$ is the total leakage current in the design. The different elements contributing to the total $I_{leakage}$ is further described in [8], but since the details are not necessary for the understanding of this thesis, they are not presented. In opposite to the dynamic power consumption only being consumed when it is switching activity in the circuit, the static power consumption is a passive consumption and is therefore consumed as long as the system is powered.

$$P_{static} = I_{leakage} V_{DD} \tag{2.6}$$

## 2.2 Low power techniques

As seen from the equations in subsection 2.1 the total energy consumption is based on several factors. When designing a low power system, these factors should be optimized as much as possible with a focus at reducing the power consumption and it exists several design methodologies for such an optimization. One of these are voltage scaling by lowering the operating voltage $V_{DD}$ which has a quadratic effect on the dynamic power in equation 2.5 as well as affecting the static power in 2.6 and is therefore often considered the a key element to optimize. However having multiple voltage domains in a system may add different challenges and since the SPI master always will be a part of a larger system and therefore often have to adjust to the operating voltage of the rest of the system, this option is not further explored. Some other more easily adaptable low power techniques for the SPI master itself are described below, but it exists multiple more who could be further explored[8].

### 2.2.1 Frequency scaling

As seen from equation 2.5 the dynamic switching power consumption is proportional to the operating frequency $f$ where a larger frequency gives a larger consumption and a system should therefore not run faster than necessary. A reduction in frequency also gives the possibility to use downsized transistors or a lower supply voltage[8]. A system can have different frequency domains in different parts of the system where for

instance a bus interface can run on a lower operating frequency than the operating clock for the entire system.

A lower frequency gives a larger on-time for the system which could give a larger total energy consumption as seen in equation 2.1 due to an increased time period $T$. Even if the dynamic power consumption is reduced at a lower frequency, the static power consumption is constant and with a larger time interval, the static contribution to the total energy consumption increases. Therefore the designer must find the best balance between static and dynamical consumption by finding the optimal operating frequency with the lowest total energy consumption.

### 2.2.2 Clock gating

Clock gating is a technique where some enable logic are added to the clock in order to stop it propagating to certain elements of the hardware. Such clock gating logic can be added in a variety of ways, but the concept is based upon adding either logic or a specific signal who can turn on and off a gate and by such stop the switching clock signal to reach a section of idle blocks of registers[8]. This method prevents switching in the registers and stop the activity in downstream logic. Since the clock has a large activity factor the possible power reduction using clock gating may be large depending on the amount of gated elements. The clock gating comes with an overhead where extra logic, interconnects and switching activity are added to the design in order to provide the gating functionality. This overhead gives an extra power consumption to the design, and gating should therefore only be used in cases where the power savings are higher than the clock gating power overhead[9, 10].

### 2.2.3 Capacitance reduction

Switching capacitance in a digital design comes from wires and transistors in the circuit. Many circuits are dominated by the wire capacitance and the importance of minimizing wiring through good floorplanning and placement is high. The switching energy required by a wire is set by its capacitance where the longer the wire is, the more capacitance it has[8]. In order to reduce the power required by all the capacitance in the circuit, it is desirable to reduce the amount of wiring and logic.

## 2.3 FPGA

A Field-Programmable Gate Array (FPGA) is a re-configurable device made up of a combination of configurable logic blocks and configurable routing fabric. These elements can implement the logic part of a system-design into logic blocks and route these together. This gives the possibility to re-configure the logic in the FPGA to implement different digital hardware designs. In order to make the FPGA re-configurable, the

logic blocks are implemented as Look-Up Table (LUT)s and it is required to use a large number of multiplexers in order to make the routing flexible[11]. A FPGA device also consist of flip-flops used as registers to store data values between clock pulses and I/O-ports to transmit or receive signals from other systems.

### 2.3.1 FPGA design process

When creating a system-design for a FPGA, the designprocess goes through different steps which can affect the systems performance in different ways. The design can for instance start as a RTL specification of the design specified as code in for instance the HDL-languages Verilog or VHDL. This design specification is then synthesized into an actual circuit consisting of gates, flip-flops and different types of logical elements. The result of this process is a design netlist where all the required logical elements needed for the different parts of the design are listed. The final part of the process is the implementation of the design. In this step the synthesized netlist is mapped into the FPGA. This process places the required logic onto the device based upon available resources in the FPGA and routes these together[12, 13].

### 2.3.2 Power consumption in a FPGA

The power consumption in a FPGA can be divided into three components: device static $P_{devStat}$, design static $P_{desStat}$ and design dynamic $P_{desDyn}$[14]. The total power draw $P_{total}$ from the voltage supplies of the FPGA is given by the sum of these components as seen in equation 2.7 and the total static power $P_{static}$ draw is given by the sum of the device static and design static as seen in equation 2.8.

$$P_{total} = P_{devStat} + P_{desStat} + P_{desDyn} \tag{2.7}$$

$$P_{static} = P_{devStat} + P_{desStat} \tag{2.8}$$

The device static power represents the power required to make the FPGA available for programming where a large portion is due to leakage in transistors used for holding the device configuration. This power consumption is mostly dependent on manufacturing, process properties, applied voltage and the device junction temperature and is independent of the implemented design. The design static power is the static power consumption when the FPGA is configured with the created design, but there is no activity. This is mainly due to I/O terminations, clock managers and other circuits who consumes power without any design activity. These blocks are enabled depending on the requirements of the design and has a set amount of static power consumption. Lastly the design dynamic power is generated from the design activity and depends

on capacitance and activity of utilized resources and scales with the applied voltage level[14].

## 2.4 FPGA vs ASIC

A different option to implement the design on a FPGA is to use an Application-Specific Integrated Circuit (ASIC). It gives the same possibilities for designing a digital design in hardware, but with some major differences. In opposite of the re-configurable hardware in the FPGA, the ASIC is not re-configurable and each chip is customized and produced for its specific usage. When designing a digital system it is therefore important to choose whether the design should be implemented to a FPGA or an ASIC. All the pros and cons for the different devices are not presented here since this is not the main focus of this thesis. Due to the possibility to re-configure the logic in a FPGA, it is chosen to use this device due to the nature of the thesis where different designs are tested out at the same device.

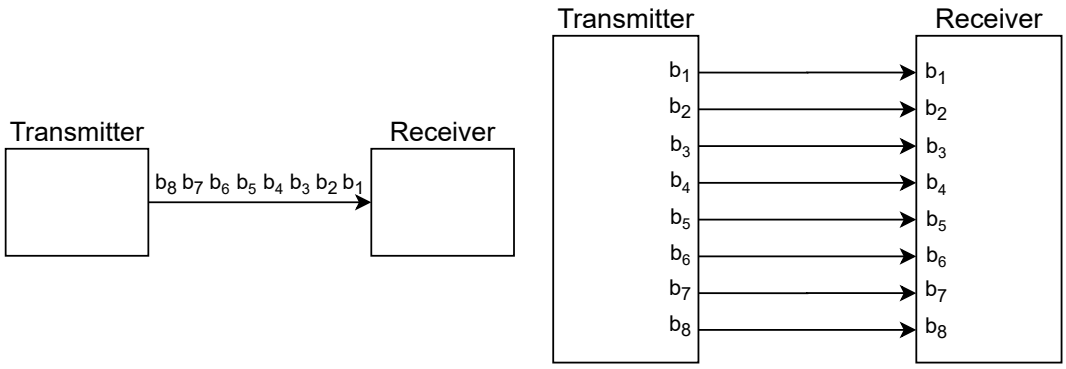
Based on research a FPGA can be 7-14 times less energy efficient than an ASIC and the implemented design can be 5-35 times larger in area[15]. This is because the ASIC can be fully optimized to the implemented design while the FPGA has additional circuitry and transistors in order to make them re-configurable and able to implement a variety of different designs. As the focus of this thesis is to compare different SPI master designs, the gap between FPGAs and ASICs is not necessary of a great importance, but it can affect the size of the power and energy consumption in the results and should therefore be taken into consideration when looking at the individual numbers and will be further discussed later in this thesis.

### 2.4.1 Clock networks

A clock network has a large impact on the power consumption in a digital design. This is due to the high activity factor $\alpha$ and a large fanout since it is connected to a large amount of logic blocks which gives a large capacitance $C$. These two factors can give a large dynamic power consumption which can range from 25% to 50% of the total dynamic power depending of the implementation of the design[16]. The importance of optimizing the clock networks in order to save energy is therefore important in all digital implementation, but due to the larger energy consumption in a FPGA compared to an ASIC, the clock optimization is of even greater importance in a FPGA[17].

## 2.5 Communication in digital wired systems

As mentioned in the introduction there exists a variety of methods for communication and this thesis is focused at digital wired communication in embedded systems. As also mentioned there still exists a large variety of communication methods and protocols within these limitation. These can be sorted into two categories as shown in figure 2.1 where the difference is whether the bits are transferred in serial or in parallel. In the serial case, the bits $b_{1-8}$ are sent one by one on the same line after each other, while in the parallel case all the bits $b_{1-8}$ are sent at the same time, but over multiple lines. This lead to a longer transmitting time for the serial case, but with a lower area footprint since the parallel design requires more space due to all the required connections.




(a) Serial communication (b) Parallel communication

Figure 2.1: Alternative ways of communicating in digital systems, from [6]

Another important aspect of communication is to ensure all the communicating devices are able to interpret the message correctly. A part of this is to have a common understanding between the devices of when to read of the different bits in the message. Again it is a split into two different communication styles, synchronous communication and asynchronous communication. In the synchronous style a clock signal is transferred on a separate line at the same time as a message is transferred. The sender and receiver will have a predefined or chosen understanding on when on the clock flank the data bit should be read or sent. In the asynchronous case there is no such common clock signal. Since the devices still need to know when to read the bits, the devices often has a common preset communication speed. The data message will then typically begin with a specified start bit and the data bits will then be read or sent at the specified time intervals.

The communication lines in the design are also an elements of flexibility. As shown in figure 2.2 it mainly exist three options for the different usage. The first is the simplex design where the communication only goes from the transmitter to the receiver over

one line. This gives a low area and pin footprint, but it is not possible for the receiver to transmit data back to the transmitter. In the half duplex design the data is transmitted both to and from both of the devices over the same data line. This still gives a low area and pin footprint, but gives a lower communication speed since the data can only be transmitted one way at the time. In the full duplex design data can still go both to and from both the devices, but on different lines. This gives the possibility to transmit data both ways at the same time and could therefore be more efficient than the half duplex design, but at the cost of more required area and I/O-pins due to the required two lines.

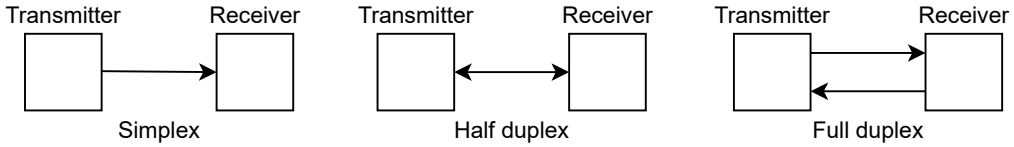



Figure 2.2: Setup of bus directions in communication, from [6]

## 2.6 External interface of SPI

The Serial Peripheral Interface (SPI) is one of the most common and used communication protocols in low level communication in embedded systems[6]. It was created by Motorola in the mid 1980s and has since then been developed into different variations, but the original and most common implementation is the protocol explored in this thesis. The design of a SPI device can be divided into two parts. Firstly it is the external side which specify the external interface from required connections and interactions between different devices on the bus. The second side is the internal side which generates the functionality required to operate the external interface. The external side can therefore be considered more of a written set of specifications of what a SPI bus requires while the internal side is the actual hardware or software implementation to fulfill the external specifications. The conventional design of the external interface is a full duplex, synchronous and serial communication bus between a single master[2] device and one or more peripheral devices[18]. An example of a possible setup of the external interface is presented i figure 2.3 where its seen one master device connected to two peripherals named $Slave1$ and $Slave2$. The conventional setup of the SPI bus consists of the signal wires Serial Clock (SCLK), Master Out Slave In (MOSI), Master In Slave Out (MISO) and Slave Select (SS). The signals SCLK, MOSI and MISO are shared between all the devices while each slave has a separate SS signal as seen in the figure.

[2]Some sources states that the SPI can be a multi-master bus while other states that it is a single-master bus. A multi-master functionality can be achieved, but creates challenges to the slave select procedure and is therefore not common[18].

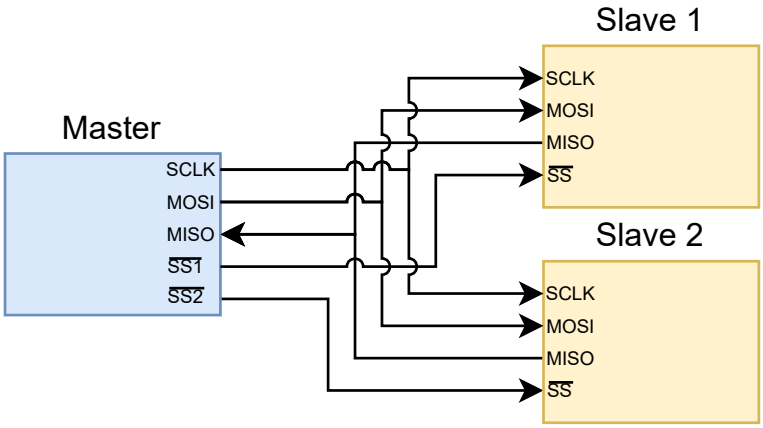



Figure 2.3: Example of external SPI interface, from [6]

A SPI transmission is initialized by the master device by first lowering the SS-signal for the desired slave. Then the clock signal SCLK is initialized and the data is transmitted onto the data-lines MOSI and MISO depending on the desired operation. Due to the full duplex connection in the bus, a transmission can occur in both directions at the same time. When the transmission is done, the SS-signal is raised and the bus is ready for another transmission. From the specifications there is no specified communication frequencies at the clock signal, and the communication speed can therefore be chosen freely be the designer. However the frequency is limited by the timing limitations for the hardware so some conditions must be met. Often the SCLK signal is created as a division from the system clock. For the SPI protocol the only overhead for transmission is the lowering of the SS-line. As this signal is applied on a separate connection-line and the switching on the SCLK-line is initialized slightly after, the SPI device is able to transmit data at all cycles of the SCLK without any overhead in SCLK cycles which means that for the SPI transmission, the SCLK frequency equals the bitrate during transmission.

From the SPI specifications there exist different operation modes as presented in figure 2.4 where the modes are defined by the adjustment of Clock Polarity (CPOL) and Clock Phase (CPHA). The SPI devices might be able to freely change the operating mode between transmissions by adjusting the settings for CPOL and CPHA or the devices might be locked into certain predefined operating modes. This is dependent on the implementation of the SPI device. Both the master and the slave must operate in the same mode in order to properly communicate together. The CPOL determines if the clock is considered active high or low, where 0 at CPOL specifies an active clock on a rising edge and a 1 at CPOL specifies an active clock on a falling clock edge. As seen from the figure, the CPHA bit determines when the data is sampled. When CPHA is set to 1, the data is sampled at the second edge of the clock pulse and when CPHA is set to 0, the data is sampled at the first edge of the clock pulse. The different operation

modes are named 0, 1, 2 and 3 based upon the bits combined for CPOL and CPHA as 00, 01, 10, 11.

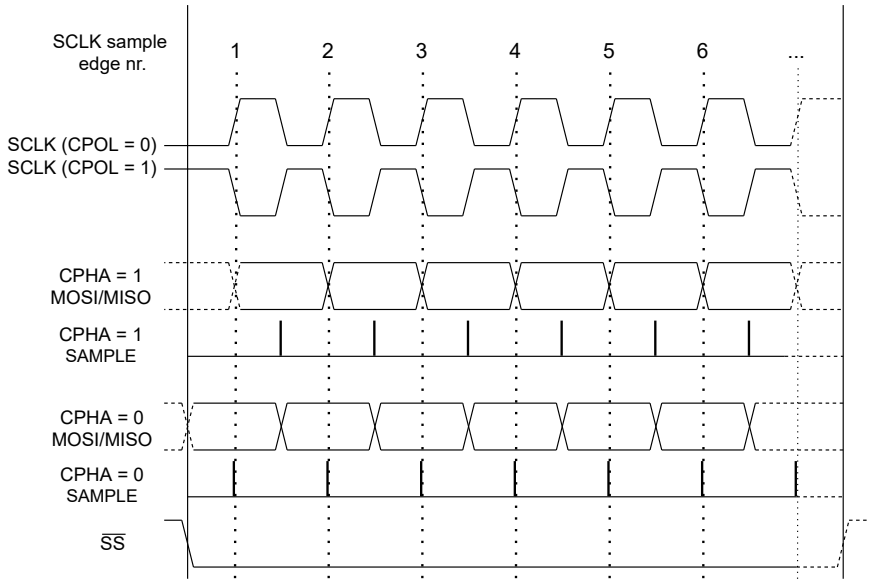



Figure 2.4: Different operating modes in SPI transmission with alternations to CPOL and CPHA, modified from [18]

## 2.7 Internal design of SPI masters

The external SPI interface presented in the previous section can be considered a minimum for what a SPI device should be able to do, but it does not specifies how the internal hardware should be designed or connections to other internal subsystems of the complete system. These loose specifications for the internal requirements can give a large variety of possible implementations where the SPI device could be highly specialized for the overall systems requirements where examples of additional functionality are presented in the following subsections. Two different implementations of the SPI master device are presented below with different levels of complexity[3]. These SPI masters are selected among a large variety of publicly available SPI masters so they only represents a small selection of possible implementations. These two implementations are selected due to their different implementation strategies where the first one has a simple design where only the minimum of requirements to operate the external interface are implemented while the second one includes additional features who will be described in subsection 2.7.2.

[3]The full RTL descriptions of the implemented SPI masters are not presented in this thesis due to copyright agreements, but are available in the presented references

### 2.7.1 Simple implementation

The first explored SPI master is an implementation modified from [19] with the I/O-connection to internal logic and external devices as shown in figure 2.5. The internal logic in the figure represents the internal logic or subsystems controlling the SPI master device and could for instance be a microprocessor or some other control logic. Since the SPI master often is a module purely used for communication, a device only consisting of a SPI master will have a very limited functionality and the SPI master is therefore in all cases connected to some other control logic or component. The external device will typically be a SPI slave device. A description of the different input and output ports of the connection between the internal logic and the SPI master is presented in table 2.1. The connections to the external device is as described in section 2.6, but in this case four slaves are connected to the SPI master as seen from the four SS lines. This implementation of the SPI master can be considered a simple implementation based upon a review of the RTL, functionality and connections to internal logic. This is because the design does not add any additional functionality other than the one required to fully operate the external interface. The functionality and connections to internal logic could however be further reduced by specifying a fixed operation mode and clock division in the module and thereby remove some connection ports and logic, but since this functionality is required for further tests, this is not removed.

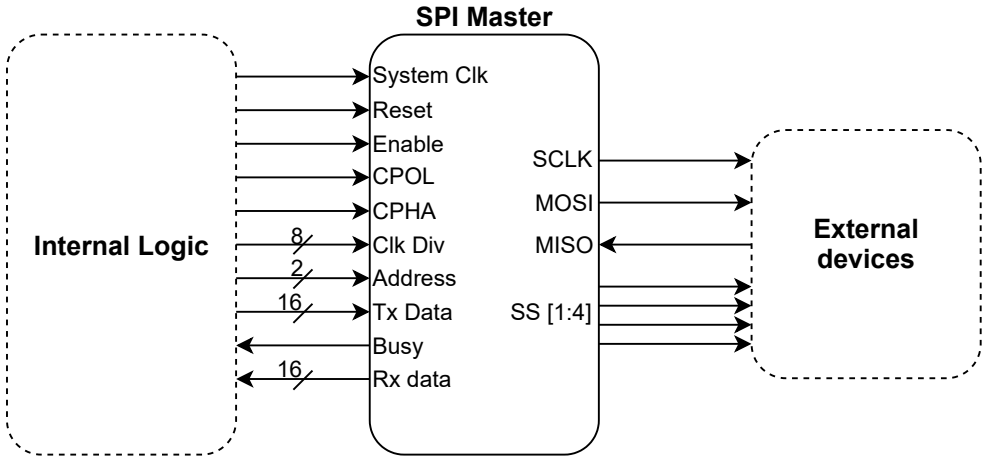



Figure 2.5: I/O-connections for simple implementation of SPI master

As seen in table 2.1 the different connection ports have different directions and widths based on the utilization of the port. The `Tx data` and `Rx data` port have a bit-width of 16. This allows the internal logic to load or read 2 bytes of data into the transmission buffer or from the receive buffer at the time. As seen the width of the address port is 2 bits. This allows for a total of 4 connected slaves which corresponds to the connected devices as shown in figure 2.5.

Table 2.1: Description of I/O-connections for simple implementation of SPI master

| Port | Direction | Width | Description |
|---|---|---|---|
| **System Clk** | Input | 1 | Operating clock from controlling system |
| **Reset** | Input | 1 | Asynchronous active low reset |
| **Enable** | Input | 1 | Start transmission |
| **CPOL** | Input | 1 | SCLK polarity |
| **CPHA** | Input | 1 | SCLK phase |
| **Clk div** | Input | 8 | SCLK division from system clock |
| **Address** | Input | 2 | Address to target slave |
| **Tx data** | Input | 16 | Data to transmit |
| **Busy** | Output | 1 | Busy signal, set to 1 during transmission |
| **Rx data** | Output | 16 | Received data from slave |

The implementation of this SPI master is done using the Hardware Description Language (HDL) VHDL. However due to some limitations in the simulation tool who is further described in section 3.2.2, the design is synthesized to a Verilog netlist and some modifications to the original design are therefore applied as described in the list below due to the crossover between languages. These alternations does not affects the functionality of the design, but might add some extra components compared to the original implementation.

- The input ports for `Clk div` and `Address` were originally of VHDL integer type, but this datatype is not allowed in a VHDL instantiation in Verilog in the utilized tool. These input ports were therefore changed to `STD_LOGIC_VECTOR` type and converted to integers as a part of the SPI master design[20].
- The output ports `SCLK` and `Slave` were originally implemented as buffer directives, but this is changed due to the same reason as the previous point as this directive is unsupported by the tool in the mixed language implementation. Instead dummy registers are implemented as buffers and the output ports are connected to these registers[20].
- Originally the SPI master had functionality for continuous transmission of data where the last transmitted data were repeated until new data arrived. This is removed in the analysed design in order to simplify the functionality as much as practically possible.

The internal functionality of the implementation is based on synchronous operation on a rising clock edge of the system clock. The internal processes are implemented using a single state-machine handling both the data transmission and reception onto the MOSI and MISO lines in addition to generation of the SCLK. A simplified overview of the state-machine is shown in figure 2.6. In this figure it is seen that the design is set to the `READY` state upon reset and stays in this state until a transmission is initialized.

The downscaling of the system clock frequency down to the specified SCLK frequency is done using a counter who counts each system clock pulse. The end number of the counter is specified by the `Clk div` port and halfway during the count and at the end before counter reset, the SCLK signal is toggled and a output communication clock signal is generated. The counter also controls the proper time for writing to the MOSI line and reading from the MISO line.

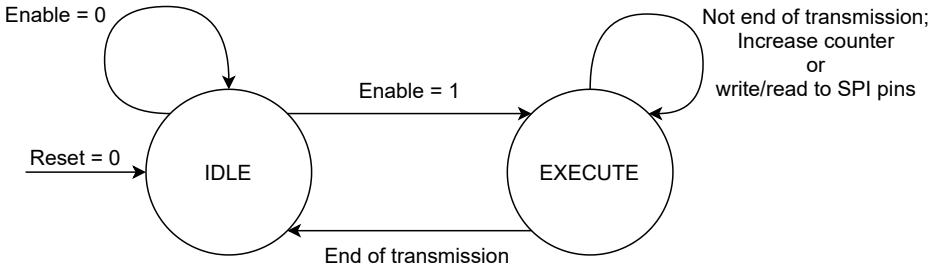



Figure 2.6: State machine for simple implementation of SPI master

This implementation of this SPI master module does not contain any form of storage and depends on the controlling logic to send the next data for transmission after the previous data transmission is finished.

### 2.7.2 Complex implementation

In opposite of the more simple SPI master implementation presented in the previous subsection, the implementation presented by [21] includes more internal functionality and possible adjustments with the same external interface. The connections to the design are presented in figure 2.7 with a further description in table 2.2. The internal logic and external devices will be of the same device-type as for the simple implementation where the internal logic for instance can be a microcontroller and the external devices can be different SPI slaves. As seen in the figure the external interface is similar as for the simple SPI master, but the internal interface has more connections.

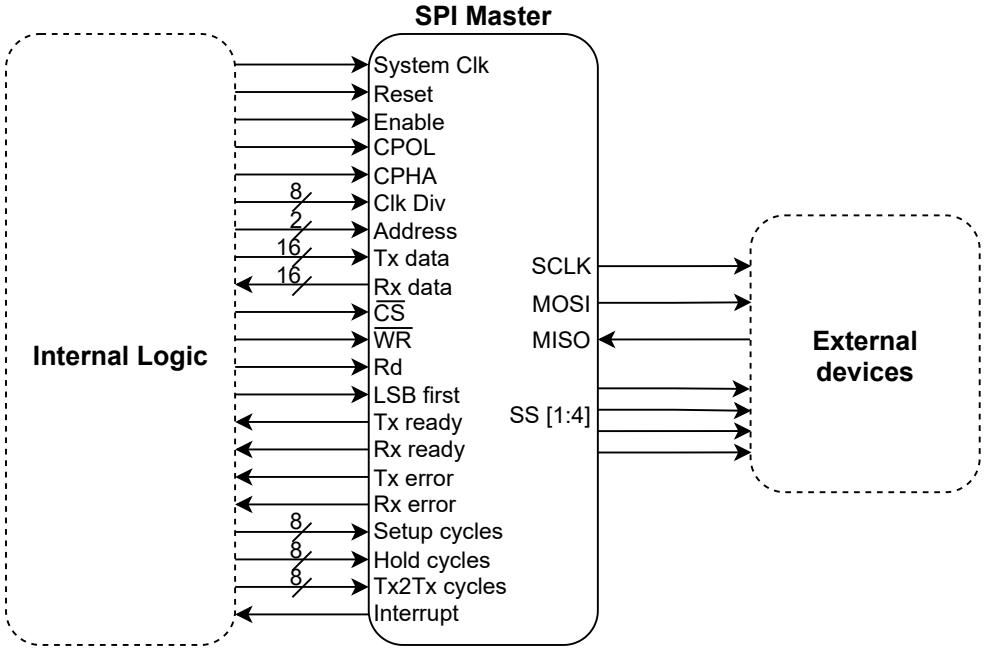



Figure 2.7: I/O connections for complex implementation of SPI master

The additional connection ports are required as a result of the additional functionality in the implementation. As mentioned table 2.2 gives a further description of the functionality of the ports. Compared to the connections for the simple implementations presented in table 2.1 it can be seen that the port `Busy` is removed and all the ports from $\overline{\texttt{CS}}$ and down are added. The ports $\overline{\texttt{CS}}$, $\overline{\texttt{WR}}$ and `Rd` add the possibility to select this SPI master in the same way as a SPI slave is selected with the SS-signal and choosing whether the controlling logic want to write or read transmission data into or from the SPI master. The `LSB first` port gives the possibility to choose if the Least Significant Bit (LSB) or the Most Significant Bit (MSB) of the data should be transmitted first onto the MOSI line. The `Tx/Rx ready/error` ports are used to report different system statuses regarding transmission and reception back to the controlling logic in case an error occurs or the SPI master is ready for a new transmission or has received some data. The ports `Setup cycles`, `Hold cycles` and `Tx2Tx cycles`

are used for adjustment to the transmission behavior and is further described later in this section. Lastly it is implemented an interrupt functionality at port `Interrupt` to signal the controlling logic that one of the status ports are set or one of the internal FIFO registers, who will be described later, are full.

Table 2.2: Description of I/O-connections for complex implementation of SPI master

| Port | Direction | Width | Description |
|---|---|---|---|
| **System Clk** | Input | 1 | Operating clock from controlling system |
| **Reset** | Input | 1 | Asynchronous active high reset |
| **Enable** | Input | 1 | Start transmission |
| **CPOL** | Input | 1 | SCLK polarity |
| **CPHA** | Input | 1 | SCLK phase |
| **Clk div** | Input | 8 | SCLK division from system clock |
| **Address** | Input | 2 | Address to target slave |
| **Tx data** | Input | 16 | Data to transmit |
| **Rx data** | Output | 16 | Received data from slave |
| $\overline{\textbf{CS}}$ | Input | 1 | Active low chip select |
| $\overline{\textbf{WR}}$ | Input | 1 | Active low write enable |
| **Rd** | Input | 1 | Active high read enable |
| **LSB first** | Input | 1 | Choose if LSB or MSB is transmitted first |
| **Tx ready** | Output | 1 | Transmitter ready |
| **Rx ready** | Output | 1 | Receiver ready |
| **Tx error** | Output | 1 | Transmission error |
| **Rx error** | Output | 1 | Receive error |
| **Setup cycles** | Input | 8 | SPI setup time |
| **Hold cycles** | Input | 8 | SPI hold time |
| **Tx2Tx cycles** | Input | 8 | Interval between transmissions |
| **Interrupt** | Output | 1 | Interrupt from SPI master |

The design is based upon the HDL VHDL, but with a Verilog top-module so it is fully compatible with the simulation tool without any modifications. It is also based on a synchronous operation on the rising edge of the clock. The internal processes are divided up into two paths, one for the SCLK generation and one for the data handling, however the SCLK signal is used for shifting the data onto and from the MOSI and MISO lines, so the logic paths are not completely separated.

The implementation is based around a state machine which is seen in a simplified version in figure 2.8. This state machine consists of 5 states and controls both the generation of the SCLK in addition to the data handling. As seen in the figure the machine has the states `IDLE`, `SETUP`, `DATA TX/RX`, `HOLD` and `WAIT`. The master stays in state `IDLE` until a transmission is started and then goes through all the states unless a reset is invoked. In the states `SETUP`, `HOLD` and `WAIT`, the designer can specify an amount of clock cycles the design should delay from a transmission is initialized to the bits are transmitted onto the SPI lines, how long time the final bit should be held at the

data lines and lastly the time interval between each transmission of messages. These settings can be changes during operation through the `Setup cycles`, `Hold cycles` and `Tx2Tx cycles` ports shown in figure 2.7. The data at the MOSI and MISO lines are transmitted and received in the `DATA TX/RX` state.

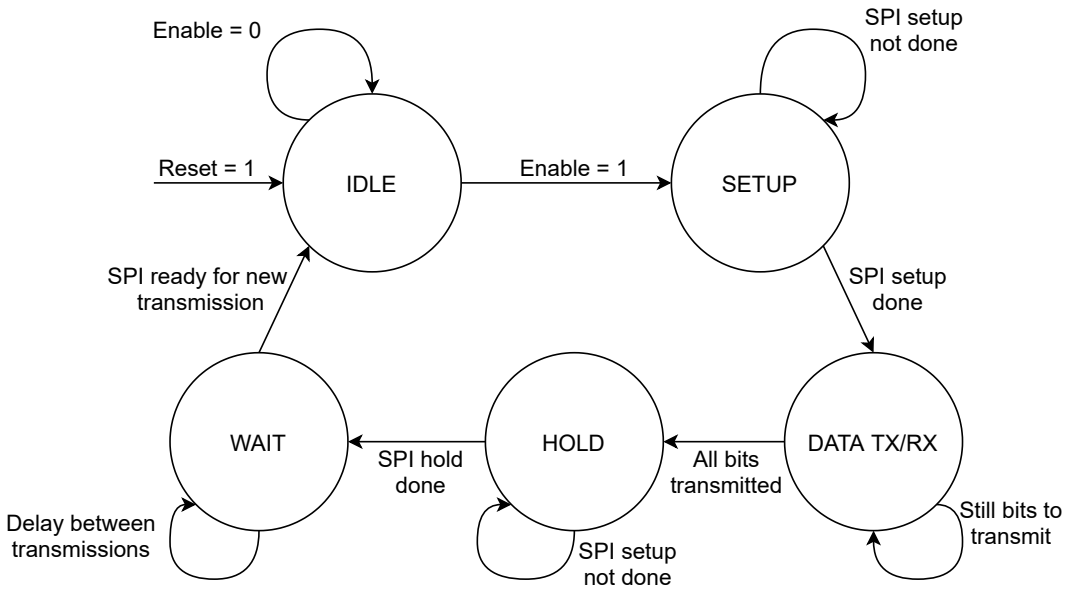



Figure 2.8: State machine for complex implementation of SPI master

For this implementation of the SPI master, two 16x16 bits deep FIFO registers are implemented. One for data to be transmitted, and one for the received data. This allow the controlling device in the internal logic to send multiple bytes of data to the SPI master in one operation and the SPI master will then handle all the transmissions based upon the specified settings as mentioned above. It also allows the SPI master to receive multiple bytes of data without the need for the controlling logic to read it out from the SPI master between each reception.

# 3 Method

This section presents the methodology for analysing the two SPI designs. Firstly the development and simulation tool is presented before the different applied tests are described.

## 3.1 Implementation of the SPI masters

In order to analyse the SPI master designs and evaluate them, the designs are synthesized and implemented onto a virtual model of the Zynq-7000 SoC using device xc7z020 with package clg400 and speed grade -1[22]. This device provides 28 nm Xilinx programmable logic equivalent to an Artix-7 FPGA and has the available programmable logic as shown in table 3.1. The SoC consists of a variety of different subsystems, but only the programmable logic is utilized in this analysis and will be refereed to as the FPGA. The designs are synthesized and implemented using the Vivado Design Suite[23]. This tool gives the possibility to instantiate a virtual model of the mentioned FPGA and can handle the entire design process presented in section 2.3.1 from RTL development to implementation of the design onto the virtual FPGA. The synthesis and implementation process are timing driven in Vivado and in order to meet the required timing requirements for the implemented designs together with accurate power estimations, a set of timing constraints must be gives to the tool[24]. The applied timing constraints and further details are described in subsection 3.2.1.

Table 3.1: Available programmable logic in the applied SoC

| Resource | Amount |
|---|---|
| **LUTs** | 53200 |
| **Flip-flops** | 106400 |
| **I/O ports** | 125 |

## 3.2 Simulation and estimation methodology

As mentioned the Vivado Design Suite is used for synthesis and implementation of the designs. The tool also gives the possibility to simulate the behavior of the design and estimate different parameters such as power consumption. Since the designs are implemented on a model of the FPGA device, the results are estimates of the actual consumption and could therefore be slightly different than measurements on a physical FPGA device depending on the fidelity of the model. This is further discussed in section 5.1.1.

### 3.2.1 Design constraints

When implementing a RTL description into a FPGA device both physical and timing constraints can be applied in the implementation process to give the tool different guidelines on how the design should be implemented. The physical constraints are applied to for instance specify I/O location, cell locations or routing limitations while timing constrains are applied to specify which clock frequency the system run at, I/O delays or other clock and delay elements who affects the design. Based on the specified constraints, the tool might do alternations when synthesising or implementing the design in order to meet the required constraints. For instance might different logic blocks be implemented closer together if their timing constraints are violated in order to reduce the distance between the logic blocks and thereby the required time it takes for a signal to travel between the them[24]. Since there are no requirements for the physical implementation of the designs in this analysis, no physical constraints are applied, but some timing constraints are applied to make sure the design meets the typical timing requirements for a SPI design.

The timing constraints are not just important for the implementation step of the process, but also important for an accurate power estimation[25]. The timing constraints are a way to tell the tool about timing requirement outside of what can be seen in the design itself. For instance may this include system and communication frequencies and I/O-delays as previously mentioned which can have an impact on the dynamic power consumption as presented in equation 2.5. For the SPI master two clock constraints are provided. Firstly a constraint based upon the system clock frequency from the internal control logic. Secondly the frequency of the SCLK who also needs to be constrained since it contributes to the power dissipation. The SCLK is for both of the designs implemented as a downscaled version of the system clock and the constraint is therefore given as a scaling of the system clock. Since the required clock constraints will change based on the applied system clock, no common clock constrains are applied to all tests since these span over different frequencies as presented later in this section.

The I/O-constraints of the design are dependent on the up- and downstream devices of the design, i.e. the internal logic controlling the SPI master as the upstream device and the connected SPI slave as the downstream device. The SPI masters are analysed as stand alone devices with a testbench directly connected to the design and are therefore not connected to a specific upstream device. This leads to difficulties when specifying the delays on the internal side of the module since these delays will depend on the implementation strategy and hardware mapping of the combined system of internal logic and SPI master. The exact mapping is not determined until the combined system is implemented onto a FPGA and the exact I/O-delays are therefore also unknown until the complete implementation. Therefore it is not specified any I/O-constraints for the

internal connections of the SPI masters. For the external connections the constraints depend on the requirements of the applied SPI slave device. The applied constraints on the MISO and MOSI ports are shown in table 3.2 and are based upon the requirements of a representative SPI slave device[26]. Since the MISO port is an input to the SPI master device, an input constraint is specified for this port. For the MOSI port it is opposite and an output constraint has been specified since it is an output from the designs. All the constraints are specified relative to the SCLK since this clock is used for clocking of the communication, and the requirements from the SPI slave are relative to this clock. The setup and hold times then specify when the data should be ready relative to the specified clock where a negative time on the output constraint means that the data is sent before the clock edge[27]. As seen in the table the data is transmitted on the MOSI line 2 ns before the SCLK flank due to the negative value, and held stable 2 ns after the clock flank. For the MISO line the data arrives 55 ns before the SCLK flank and is held 55 ns after the SCLK flank.

Table 3.2: Port constraints in SPI implementation

| Port | Setup time | Hold time | Unit |
|---|---|---|---|
| **MISO** | 55 | 55 | ns |
| **MOSI** | -2 | 2 | ns |

### 3.2.2 Simulation testbench

Another important part of a confident energy estimation is to have a representative simulation of the internal switching activity in the module. This switching activity is naturally dependent on the circuits design and can be monitored by running a simulation using representative stimuli on the input ports. For an increased accuracy in the estimations, it is preferable to run the simulation after the design is implemented and routed onto the model of the FPGA[25]. This makes it possible for the simulation to use the actual timing delays of the design. By running a post-implementation timing simulation, the simulation is the closest emulation of the switching activity in the design compared to actually download and run the design on a physical FPGA[28]. The switching activity should represent either typical or worst case stimuli for an increased accuracy in the simulation and not consist of invalid data or commands as this will give an inaccurate power estimation for normal operation[25]. The switching activity is extracted into a SAIF-file which is back annotated into the power estimation. The post-implementation timing simulation in Vivado is limited to only use Verilog sources. Some elements of the designs and the applied testbench are implemented in VHDL, so in order to run the proper simulation, the simulation netlist is synthesized in Verilog even if the design source is VHDL[29].

The applied testbench works by sending a total of 16 messages each consisting of 16

bits. The transmitted messages are displayed in table A.1 in appendix A and consists of a selection of different representative messages and no invalid data. As presented in subsection 2.6 the SPI transmission can operate in four different modes based on the value of CPOL and CPHA. As a default most of the tests except the one analyzing different operating modes are analysed using mode 00 since this is the most common one. The testbench is modified to match the functionality of each of the SPI masters in order to give a similar testing behavior. The tests are therefore conducted over the same amount of time with the same amount of time between each transmission. For the complex master the possible flexibility regarding setup, hold and transmission to transmission cycles are modified to behave as closely to the simple master as possible. The testbench controls the designs so the specified messages are transmitted to the MOSI line, but the testbench also provides stimuli to the MISO line so the modules receive the same 16 messages in order to stimulate the receive part of the designs.

### 3.2.3 Power estimation

The end result and goal is to get a power estimation for the different designs. Vivado is also used for this purpose in order to report the power consumption of the designs. The power consumption in a design may differ over time due to different operations being conducted at different times. For instance may the SPI master have a larger power consumption during transmission, but a lower consumption between transmission. Vivado reports the average power consumption as given by equation 2.3 separated into a static contribution and a dynamic contribution. The time variations in power consumption are therefore removed. Each power estimation is conducted using the environmental parameters and operating voltages as specified in table A.2 and A.3 in appendix A.

*Static power estimation*

As mentioned in section 2.3.2 static power consumption in a FPGA is due to both device and design static power. The reported static power consumption in Vivado during power estimation is the total static power $P_{static}$ which is the sum of device static power and design static power as seen in equation 2.8. The design static power could therefore be hidden by a larger device static power. For the analysis of the two SPI masters the main interest is the design static power since the device static power is independent of the implemented design. In order isolate the design static power consumption, the device static consumption for the FPGA with only a single gate that never toggles is estimated and subtracted from the total static power consumption[25, 30]. The device static power estimations for only the single gate is extracted using the Xilinx Power Estimator tool[31] which is a tool provided to determine power consumption for SoCs and FPGAs even before the logic is designed. The static consumption of the design with only one gate is estimated using the same environmental and voltage

parameters as in the Vivado analysis.

*Dynamic power estimation*

The dynamic power estimation for the different implementations is directly extracted using Vivado and a vector based power estimation[25]. The vector based approach utilize the actual switching activity in the design with the use of the information in the SAIF-files generated during simulation and the use of probabilistic estimation of the activity is therefore reduced. The reported dynamic power is separated into the consumption of different elements such as clocks, logic, I/O and signals.

## 3.3 Conducted tests

When analysing the power and energy consumption of the two different SPI implementations it is of interest to figure out how they differ and the reason for their potential differences in consumption. It is also of interest to get an understanding of how the different implementations reacts when modifications and adjustments are done to the designs and compare them to each other and themselves. A series of different tests are therefore set up to get an understanding of the consequences of choosing one implementation over another when implementing a SPI master device.

Most of the tests are conducted over a set of different system clock frequencies ranging from 1 MHz to 15 MHz with a power estimation at each integer frequency. Most of the tests are also conducted with a SCLK division of 4 from the system clock giving SCLK frequencies in the range of 250 kHz to 3.75 MHz. Since the SPI protocol does not specify any communication clock frequency, it is possible to operate a SPI transmission outside of this interval as well if the communicating devices support other frequencies, but this interval can be considered a representative window of frequencies[18]. As stated is most of the tests conducted over this frequency span and SCLK division, but some of the tests are conducted using other parameters. For the test regarding SCLK alternation is different SCLK divisions applied and for the operation mode tests were only a selection of the frequencies between 1 MHz and 15 MHz analysed.

### 3.3.1 Frequency

As shown in equation 2.5 and presented in section 2.4.1 the frequency is of great importance to the power consumption in both in the SPI masters design and the FPGA implementation itself. Based on equation 2.5 the dynamic power consumption will decrease when the operating frequency is lowered. However, a lower operating frequency gives a longer on time for the system which based on equation 2.1 might increase the total energy consumption even with a lower dynamic power. It is therefore desirable to find the optimal operating frequency with the lowest energy consumption.

For a SPI master the design consists of two types of clocks. First it is the system clock based upon the operating frequency in the internal logic. The other one is the communication clock SCLK. Both of these clocks cause a large amount of switching activity in the design and are therefore of great importance to optimize with regards to power consumption.

*System frequency*

The system clock given to the SPI module is often based upon the operating frequency the rest of the system is running at, but it can still be interesting to know the optimal operating frequency for the SPI module alone since modern systems often consists of multiple frequency domains or it might be other possibilities of frequency adjustments. This test is conducted with the previously described frequency span of 1 MHz to 15 MHz where each of the two SPI master implementations are uploaded separately onto the FPGA and estimated.

*SCLK division testing*

The system clock frequency for operation of a digital system is as mentioned of great importance for the energy consumption. However the energy consumption of the SPI master is also dependent on the frequency of outputted clock SCLK since also a fair amount of logic are clocked by this clock. Since the impact to the energy consumption generated from the SCLK is dependent on how the module is implemented and how much this signal is integrated into the design, it is interesting to see how the different designs respond to different divisions of the system clock.

This test is as already described conducted using different divisions of the system clock. The divisions 4, 8, 16 and 32 are applied as these are a selection of common divisions[18]. This gives the SCLK frequency intervals of 250 kHz to 3.75 MHz, 125 kHz to 1.875 MHz, 62.5 kHz to 937.5 kHz and 31.25 kHz to 468.75 kHz respectively.

### 3.3.2 Alternation of operating mode

As shown in figure 2.4 the SPI transition can operate in different operating modes with different sampling-edges and dataalignment relative to the SCLK depending on the settings for CPOL and CPHA. Which operating mode to use might largely depend on the requirements from the used SPI slave device, but since both the explored SPI master designs give the possibility to choose the desired operating mode, it can be interesting to see how the choice of operating mode affects the power and energy consumption. As seen in figure 2.4 the operational behavior is quite similar with the same amount of switching in the external interface, so it could therefore be reasonable to think that the power consumption also would be similar between the operating modes. The different operating modes may however invoke different switching activity

in the internal designs due to different sampling behavior so some differences between the modes may occur.

The power estimation for the different operation modes are conducted over a selection of five different frequencies in the range of 1 MHz to 15 MHz since this is enough to give a good understanding of how the modules are affected by the differences.

### 3.3.3 Internal changes to the complex master

The two implementations of the SPI master present two different thoughts for implementation where one either could simplify the design to a minimum implementation or add additional functionality and status reporting. However it can also be done internal modifications to the designs themselves. Due to the simple and minimum implementation presented in the simple master, no alternations other than the ones presented in section 2.7.1 are done. For the complex master a wider span of alternations are possible due to the more complex nature of the design. One of these alternations is to not implement the FIFO registers. This will reduce the functionality of the design to give a more similar functionality as the simple design and give an understanding of where in the complex master the power consumption is created and the cost of added functionality.

The removal of the FIFO registers alternates the original functionality of the SPI design presented in subsection 2.7.2 where the SPI master no longer can handle multiple messages for transmission and reception so the internal logic must write and read each message between transmission and reception.

### 3.3.4 Power optimization

As a feature during the implementation phase in Vivado where the synthesized netlist is placed and routed onto the model of the FPGA, the tool is able to analyse the behavior of the design and add clock gating to turn off the system clock for idle logic[32]. As a part of the process some LUTs are added to control the gating, but it does not adds levels of logic to the original logic paths[9]. As presented in section 2.2.2 the effect of the gating may vary depending on the amount of gated logic and its power savings compared to the overhead in power by including additional clock gating logic. This test is not necessary a test who presents a correct representations of the differences between the SPI masters as it heavily depends on the tools ability to optimize the designs. However since these designs are implemented on a FPGA their ability to map efficiently down to the logic on the FPGA are of importance and in this case it might be differences in the designs.

# 4 Results

This section presents the results from the tests described in subsection 3.3. Be aware that in some of the presented figures the complex and simple master are plotted together, while in some they are plotted separately. The difference is described in the figure title where each SPI master is explicitly written when they are plotted separately.

The results are presented showing plots of dynamic power together with dynamic and static energy. The plots for power consumption is plotted based on the reported power from the power estimation in Vivado. The presented energy is the energy required for one iteration of the SCLK which consists of multiple system clock pulses. For instance for the SCLK division of 4 does one SCLK pulse consists of 4 system clock pulses. The energy consumption is then calculated by multiplying the specific average power consumption with the period for one SCLK pulse as seen in equation 2.3 where the average power consumption is $P_{avg}$ and the period of the SCLK pulse is $T$. Since one iteration of the SCLK transmits one bit at the datalines, the energy per SCLK iteration is the same as energy per bit for the SPI transmission.

## 4.1 System frequency

As described the power and energy consumption for the two implementations are analysed over a frequency span from 1 MHz to 15 MHz at every integer frequency in between. The results of this analysis is presented as a variety of different plot presenting both power consumption and energy consumption with different details.

As seen in figure 4.1 the dynamic power consumption of the different implementations follow an approximately linearly increase in power consumption with the increase in system clock frequency. This corresponds to the theory presented in equation 2.5 for the switching power where it is seen that the dynamic power has a linear relationship with the frequency. For the system frequencies 10 MHz and 11 MHz the power estimations for the complex master has a small deviation from the linear behavior. The figure also shows that the dynamic power consumption for the complex master is larger than for the simple master with an increased gap with an increase in frequency. The percentage increase for the two implementations are further presented in the energy analysis later in this subsection.

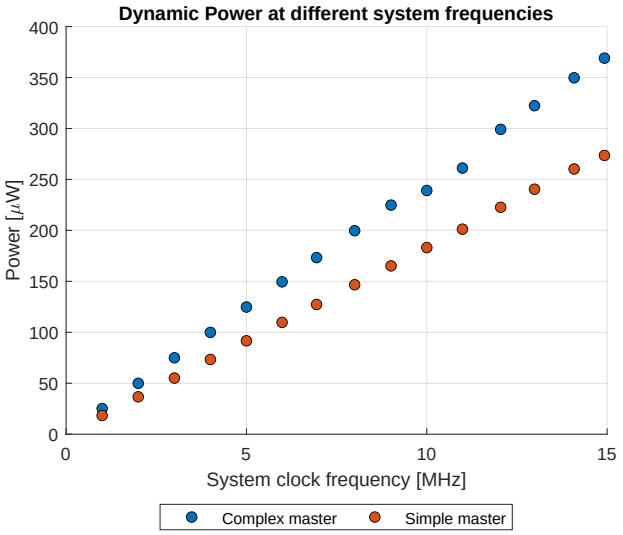
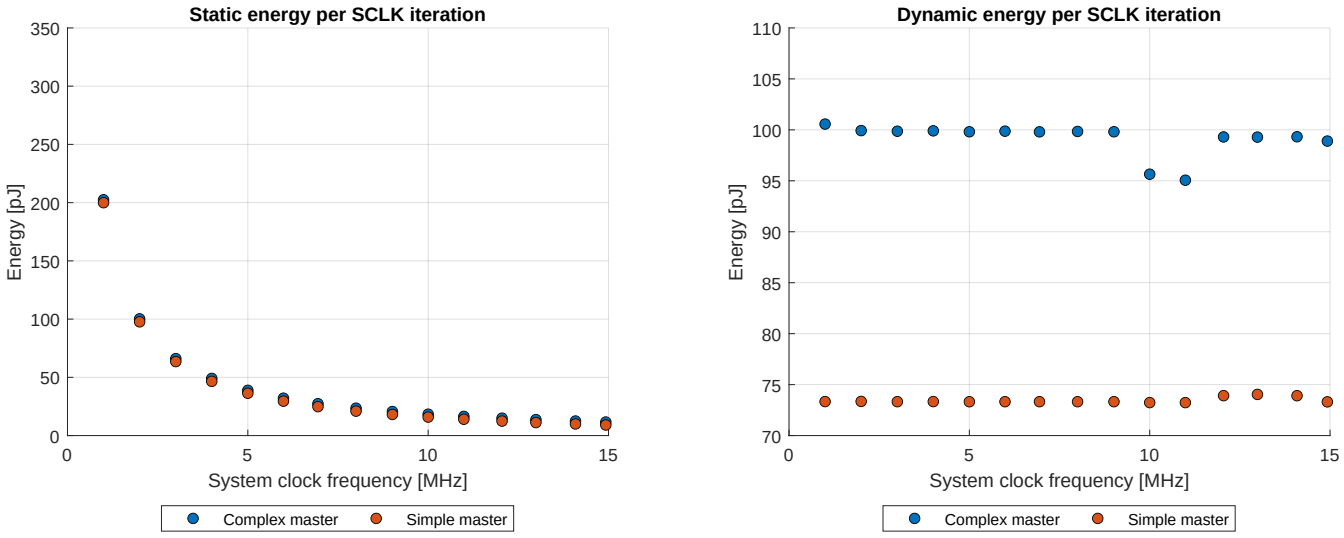



Figure 4.1: Average dynamic power consumption in both implementations of the SPI master applying different system clock frequencies.

The calculated energy estimations for the two implementations are presented in figure 4.2 and are presented as the static contribution and the dynamic contribution to the energy. As seen in figure 4.2a the static energy consumption decreases with an increase in system frequency. With a larger frequency the time period of each of the SCLK pulses is shortened from the relationship between frequency and period. From equation 2.1 it is seen that with a shorter time interval the energy is decreased even if the static

power consumption of the implementation is frequency independent. It is seen in the figure that the energy decreases rapidly between low frequencies, but decreases slower between larger frequencies. This can be explained by the nature of the relationship between frequency and period where the time-difference between 1 MHz to 2 MHz is 500 ns, but for at difference between 14 MHz and 15 MHz it is only approximately 4.7 ns. As also seen in the figure the complex master has a slightly higher static energy consumption than the simple master.




(a) Static energy

(b) Dynamic energy

Figure 4.2: Energy per SCLK iteration in the different implementations of the SPI master divided into static and dynamic consumption.

In figure 4.2b it can be seen that the dynamic energy consumption has a relatively stable consumption independent of frequency. The small deviation in the dynamic power for the complex master at 10 MHz and 11 MHz naturally also gives a deviation from the stable energy consumption, but for the dynamic energy consumption the deviation can be seen more clearly. The dynamic energy consumption for the complex master can be seen to be larger than for the simple master which is reasonable since the energy consumption is directly related to the power consumption.

Figure 4.3 show the total energy consumption with the static and dynamic contribution added together. It can be seen in the figure that the complex master has a larger total energy consumption than the simple master and the dots in the figure follow the same curve as for the static energy consumption just shifted up with the dynamic energy consumption. The energy consumption per transmitted bit for the analysed SPI masters is seen to be of the same magnitude as other conventional SPI master

implementations and communication protocols[4, 33].

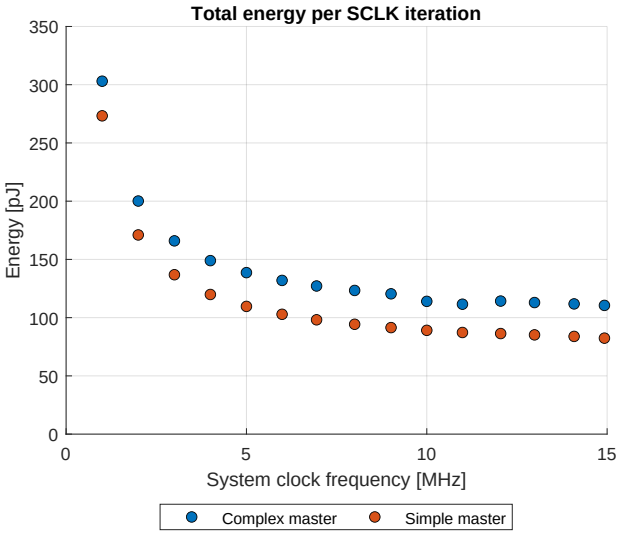



Figure 4.3: Total consumption of energy per clock iteration in both implementations of the SPI master

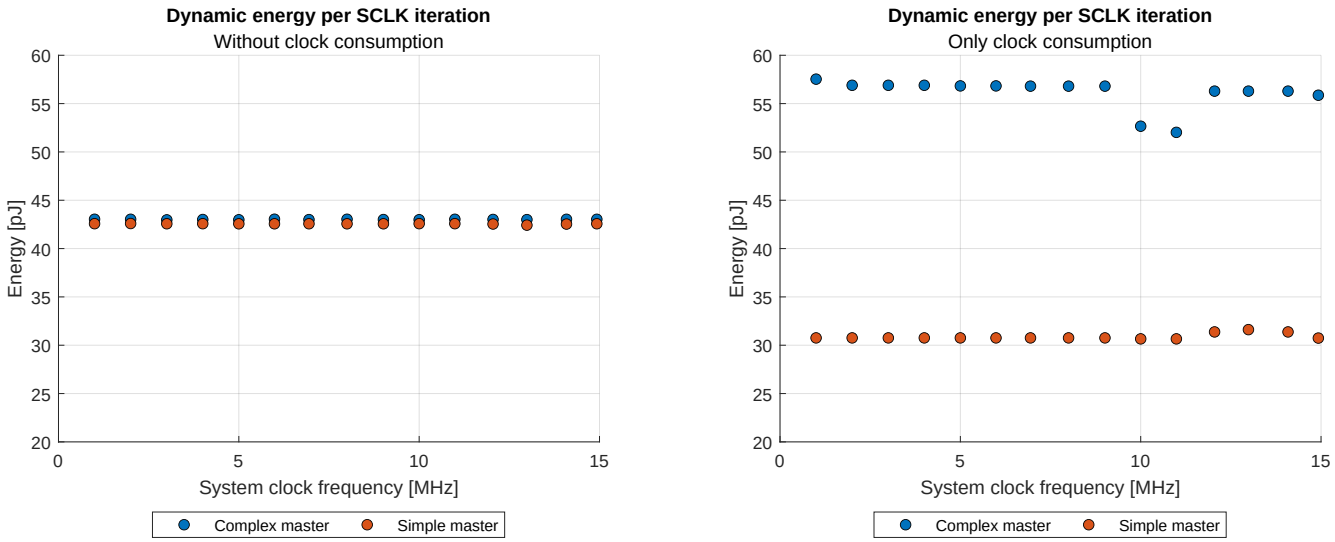
The percentage increase in energy from the simple master to the complex master is shown in table 4.1 where it is seen for the static energy that the percentage increase grows with a higher frequency, while for the dynamic energy the increase is relatively stable. Again as seen in the plots, the deviation of the estimations at the dynamic consumption for the complex master affects the results as it is seen that the increase at frequency 10 MHz and 11 MHz are deviating from the other percentages. It can also be seen in the table that the total increase in energy also rises with frequency.

Table 4.1: Increased energy in percent from simple master to complex master

| | Frequency [MHz] | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Energy type** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| **Static** [%] | 1.3 | 2.6 | 4.0 | 5.4 | 7.0 | 8.6 | 10.3 | 12.0 |
| **Dynamic** [%] | 37.1 | 36.2 | 36.2 | 36.2 | 36.1 | 36.2 | 36.1 | 36.2 |
| **Total** [%] | 10.9 | 17.0 | 21.2 | 24.3 | 26.5 | 28.3 | 29.6 | 30.8 |
| | **Frequency [MHz]** | | | | | | | |
| **Energy type** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **Average** |
| **Static** [%] | 13.9 | 15.6 | 17.6 | 20.2 | 22.5 | 25.0 | 27.8 | 12.9 |
| **Dynamic** [%] | 36.1 | 30.6 | 29.8 | 34.3 | 34.1 | 34.4 | 34.9 | 35.0 |
| **Total** [%] | 31.7 | 27.9 | 27.8 | 32.3 | 32.6 | 33.3 | 34.1 | 27.2 |

In figure 4.4 the dynamic energy consumption is separated into two plots, one where the contribution from clock consumption is removed and one where only the contribution

from the clock consumption is shown. For the plot in figure 4.4a the remaining dynamic energy consumption is the contribution from logic, signals and I/O where for the plot in figure 4.4b the dynamic energy consumption is created by clocking from the system clock, but also from the generation of SCLK. The sum of these plots equals the plot shown in figure 4.2b.




(a) Without clock consumption

(b) Only clock consumption

Figure 4.4: Energy per SCLK iteration in the different implementations of the SPI master with separation of clock contribution to dynamic energy consumption .

As seen in the plot in figure 4.4a the dynamic energy consumption in the two implementations is relatively similar however with a slightly larger consumption for the complex master. In figure 4.4b it is seen that the energy consumption caused by clocking is larger for the complex master than for the simple master and that the differences in dynamic energy consumption is mainly caused by clocking. From the figures it can also be seen that for the complex master the dynamic energy consumption from clocking is larger than the consumption from the remaining elements while for the simple master it is seen that the dynamic energy consumption from clocking is smaller than from the remaining elements.

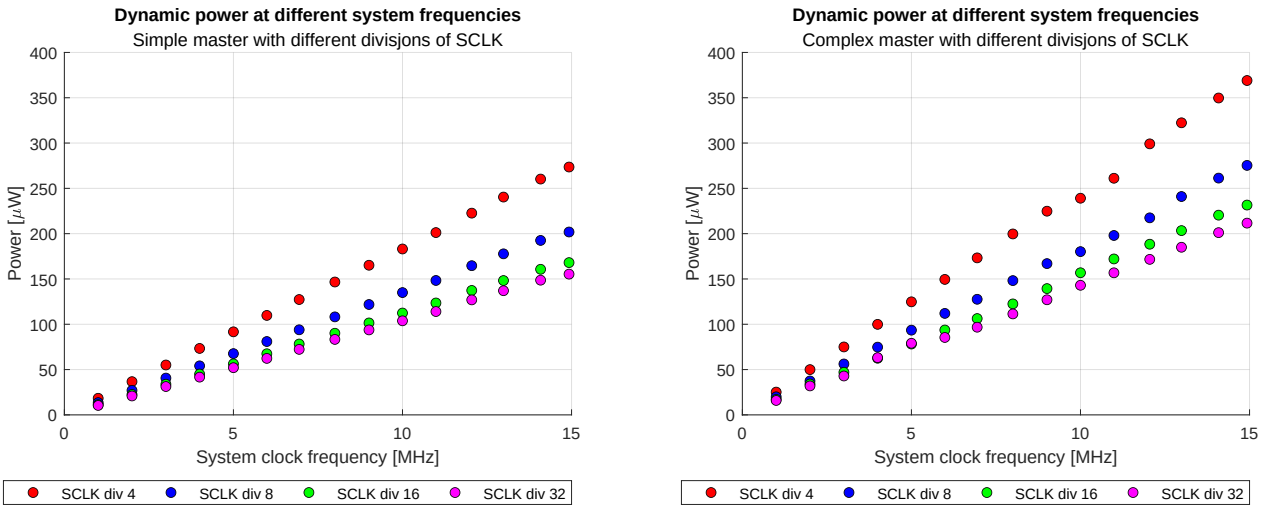
The utilized logic is presented in table 4.2 where it is seen that the complex master utilizes more resources than the simple master, but that both implementations still utilizes a small part of the available resources in the FPGA. It can also be seen that the complex master has approximately 10 times more Source Lines Of Code (SLOC) than the simple master. The numbers for SLOC are based upon the RTL description for the SPI masters as it is, but with empty lines and comments removed.

Table 4.2: Utilized resources in the FPGA with percentage of utilized FPGA resources in parenthesis

| Resource | Simple | Complex |
|---|---|---|
| **LUTs** | 60 (0.11%) | 169 (0.32%) |
| **Flip-flops** | 108 (0.10%) | 224 (0.21%) |
| **I/O ports** | 55 (44.00%) | 87 (69.60%) |
| **SLOC** | 119 | 1165 |

## 4.2 SCLK division

For the tests with different divisions of the communication clock SCLK it is interesting to see how the power and energy consumption differ with different internal adjustments. The implementations are as mentioned tested with 4 different divisions of the system clock. The resulting dynamic power consumption is shown in figure 4.5 with the results from the simple and complex master separated. The red dots in the figures are similar to the ones in figure 4.1 since these are estimated using the same clock divider. As seen for the plots in figure 4.5 all the power estimations follow an approximately linear pattern as already described for the results in the system frequency test. The lowest clock division of 4 can be seen to give the highest dynamic power consumption and the highest clock division of 32 can be seen to give the lowest dynamic power consumption for most frequencies. For the complex master in figure 4.5b it can be seen that at frequencies from 5 MHz and below, the division of 16 and 32 has a more overlap than at other frequencies and has a small deviation from the linear behavior.

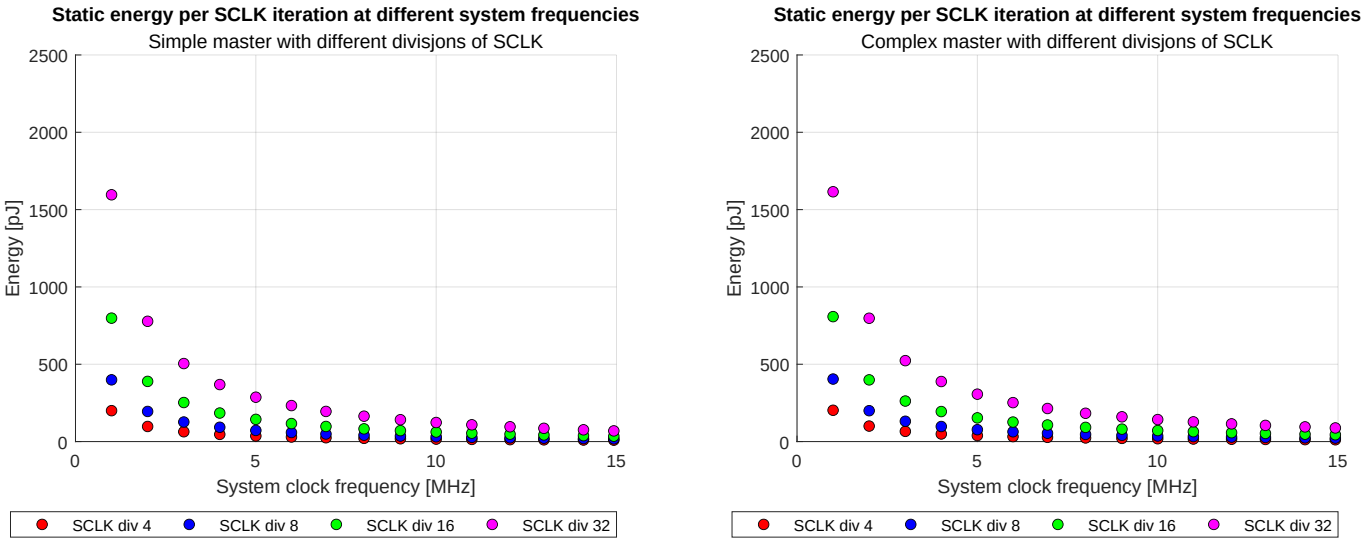



(a) Simple master

(b) Complex master

Figure 4.5: Average dynamic power consumption for different divisions between the system clock and SCLK at different system clock frequencies

The static energy consumption for the two implementations are presented in figure 4.6. As seen in the figures the static energy estimations are largest for the highest SCLK division of 32 and lowest for the lowest SCLK division of 4. It can also be seen that the energy-gap between the different divisions decreases with an increase in frequency due to the overall static energy consumption decreasing with frequency. This is caused by smaller differences in the period of the different frequencies as described for the system clock results.

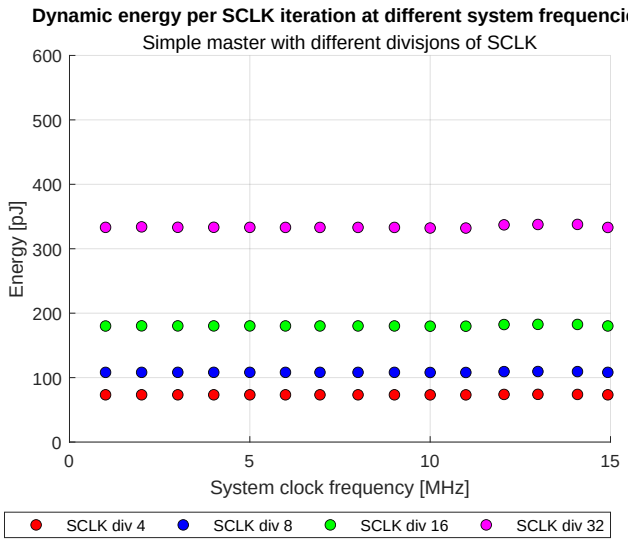



(a) Simple master

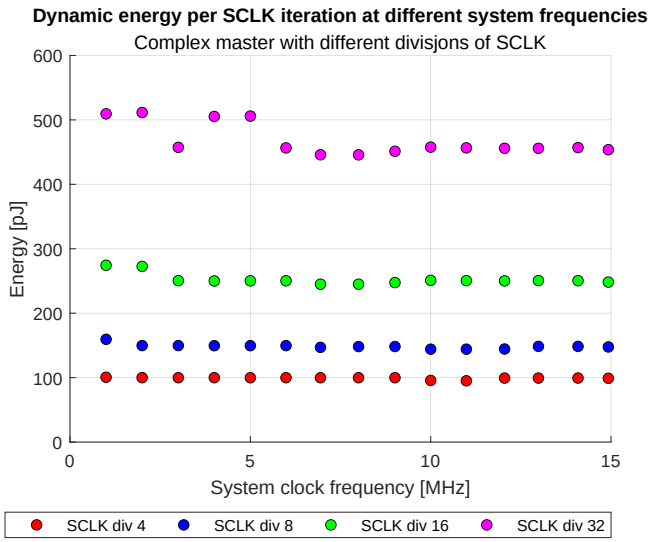
(b) Complex master

Figure 4.6: Static energy consumption per SCLK iteration for different divisions between the system clock and SCLK at different system clock frequencies

In figure 4.7 is the dynamic contribution to the energy consumption presented. As seen in the plots the highest clock division of 32 has the largest dynamic energy consumption as well and the lowest SCLK division of 4 has the lowest dynamic energy consumption. As seen in the plots the dynamic energy consumption for the simple master is lower for all clock divisions than for the complex master at the same SCLK division. As earlier mentioned the complex master has some deviations from the overall results behavior at frequencies below 5 MHz. It can be seen for 1 MHz and 2 MHz that the dynamic energy consumption is higher than for other system frequencies for especially divisions 16 and 32. It can also be seen for frequency 4 MHz and 5 MHz that the dynamic energy consumption from the 32 division also deviates. The deviation at frequency 10 MHz and 11 MHz for the SCLK division of 4 presented in the results of the system clock test, are barely visible in the plot in figure 4.7b due to the increased energy span at the y-axis of the plot. It shows that the deviations for SCLK division 16 and 32 are remarkably larger than for the ones seen with SCLK division 4.



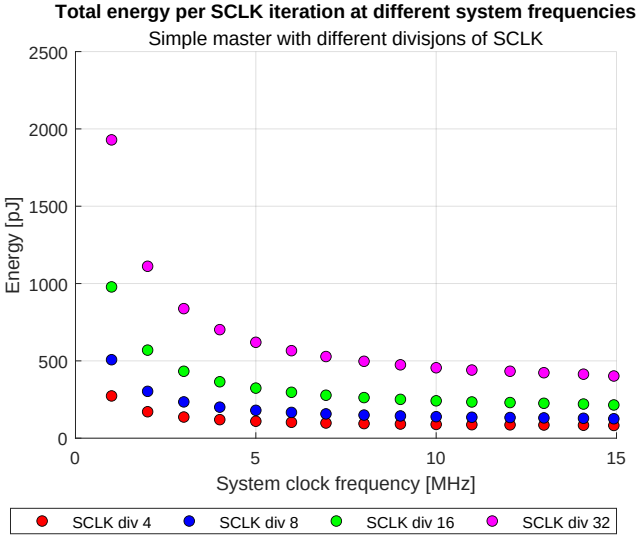

(a) Simple master



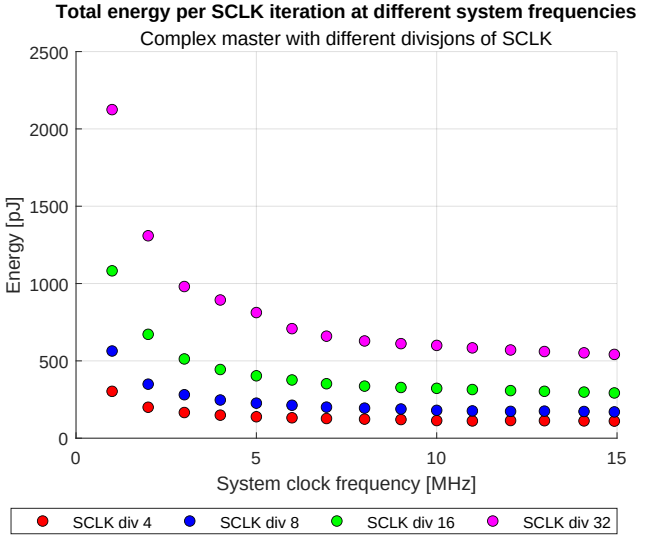

(b) Complex master

Figure 4.7: Dynamic energy consumption per SCLK iteration for different divisions between the system clock and SCLK at different system clock frequencies

For the total energy consumption per SCLK iteration are the dynamic and static energy consumption added and presented in figure 4.8 where it can be seen that the results follow the behavior of the static energy consumption shifted up with the dynamic energy consumption.




(a) Simple master




(b) Complex master

Figure 4.8: Total energy consumption at one SCLK iteration for different divisions between the system clock and SCLK at different system clock frequencies

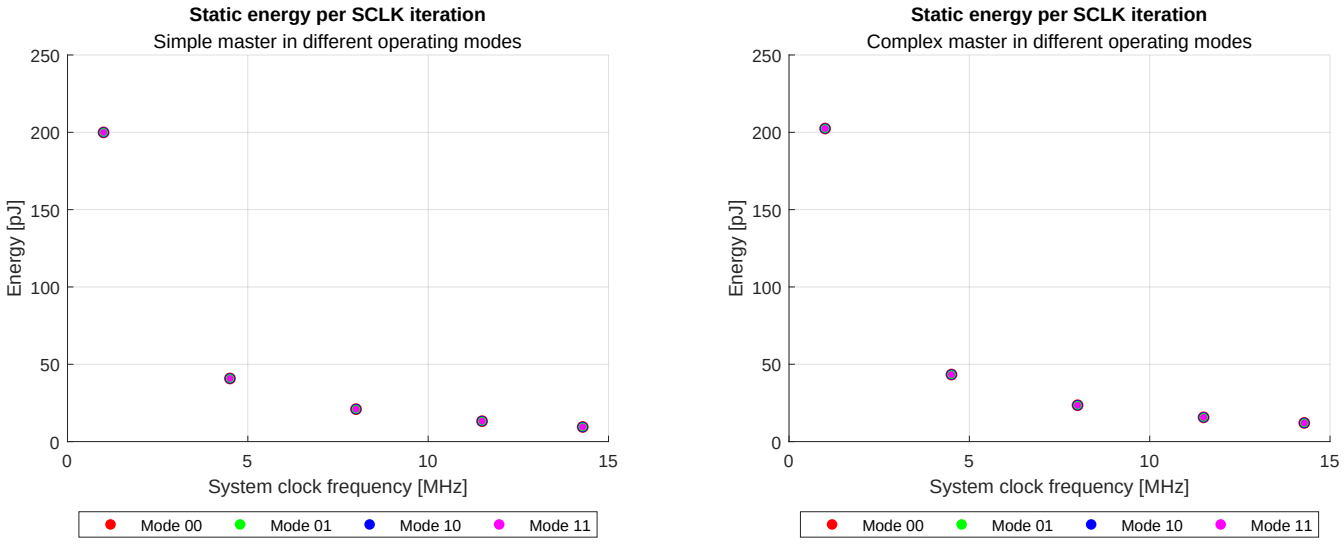
The average total energy increase in going from a system clock division of 4 to the other divisions are presented in table 4.3. It can be seen in this table that the percentage increase is larger for a larger downscale of SCLK and that the increase in energy is relatively similar for the two SPI implementations as it is only small differences in the percentage increase.

Table 4.3: Average energy increase going from SCLK division 4 to other divisions

| **Division change** | **Simple** | **Complex** |
|---|---|---|
| **4 to 8** [%] | 62.1 | 61.6 |
| **4 to 16** [%] | 188.4 | 188.8 |
| **4 to 32** [%] | 448.9 | 449.0 |

## 4.3 Alternation of operating mode

The different operating modes of the SPI transmission are as mentioned also tested by analysing the different implementations with different adjustments for CPOL and CPHA. The energy consumption for these tests are presented in figure 4.9 and 4.10 for static and dynamic energy respectively. Due to the small differences between the modes, the power plots are omitted since the differences in the plots were not visible. As seen in figure 4.9 the difference in static energy consumption between the modes is not visible as well and the different modes are therefore plotted with different dot sizes for visibility so the difference in circle size of the results are just for visual purposes. The dots for mode 00 corresponds to the results presented in figure 4.2a just with some difference in frequency for some of the dots. It can therefore be seen that the results follow the same curvature as described for the system frequency test.



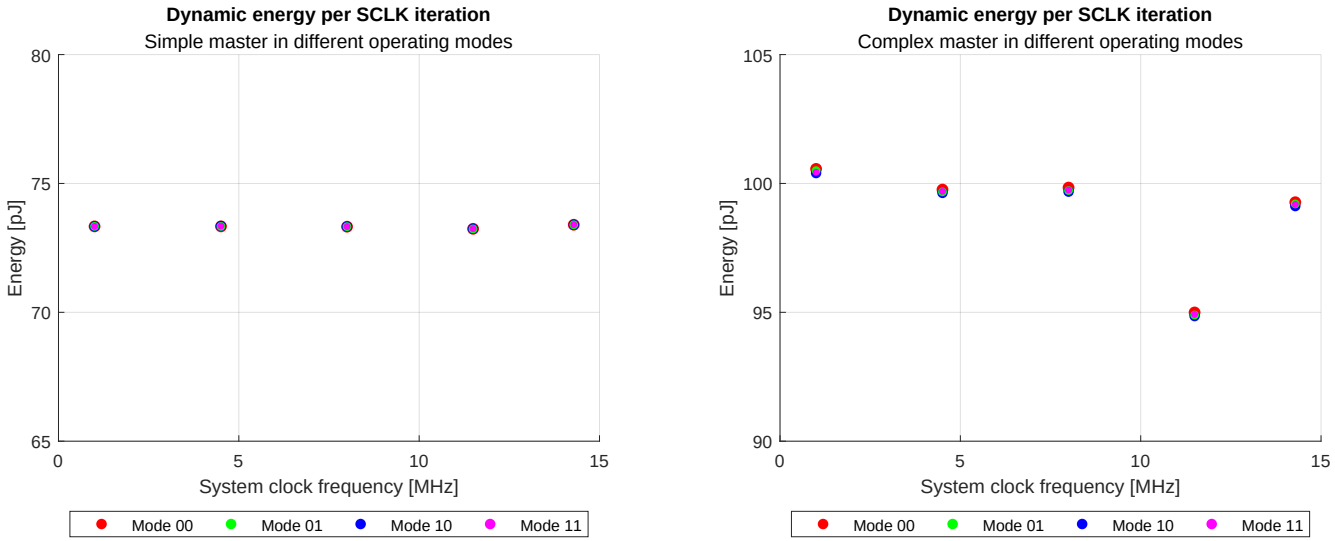

(a) Simple master

(b) Complex master

Figure 4.9: Static energy per SCLK iteration for different operating modes of the SPI masters at different system clock frequencies. Modes displayed as: CPOL CPHA

The dynamic energy consumption is presented in figure 4.10 where the results for mode 00 corresponds to the results in figure 4.2b. As seen in figure 4.10a the simple master have a relatively similar dynamic energy consumption for the different modes while for the complex master in figure 4.10b it is seen a small difference between the modes. Be aware of the y-axis of the plots as it span over a shorter range than the plots in figure 4.2b. This can especially be seen for the results at 11.5 MHz as the deviation seems larger in the plot in figure 4.10b, but is actually in the range of the deviations of 10 MHz and 11 MHz as seen in figure 4.2b. The average percentage difference between

the mode with the largest total energy consumption and the mode with the lowest total energy consumption are 0.008% and 0.109% for the simple and complex master respectively.




(a) Simple master

(b) Complex master

Figure 4.10: Dynamic energy per SCLK iteration for different operating modes of the SPI masters at different system clock frequencies. Modes displayed as: CPOL CPHA

## 4.4 Internal changes to the complex master

The changes to the complex master is as described in subsection 3.3.3 to remove the two FIFO registers used for temporal storage of the transmission and reception data. The resulting dynamic power consumption with these registers removed is presented in figure 4.11 where it can be seen that the implementation without the FIFO register uses less dynamic power than the implementation with the FIFO register. The plotted results for the implementation with FIFO registers included are the same as presented in the results for the system clock test in figure 4.1. As seen in figure 4.11 the dynamic power consumption without FIFO registers still follow an approximately linear line and has no clear deviation from the linear behavior.

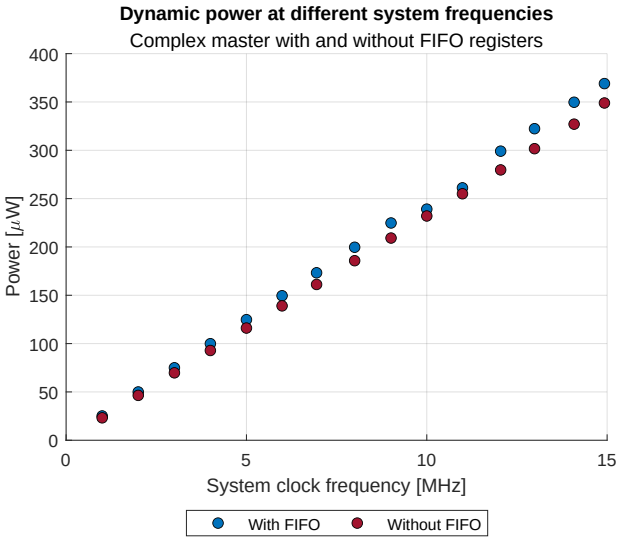
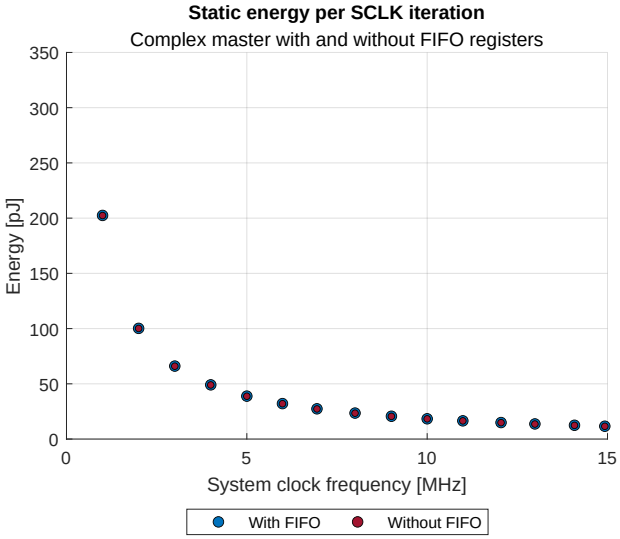
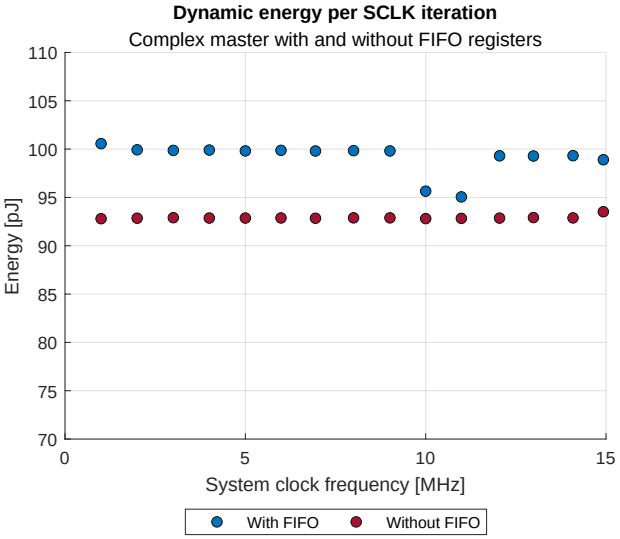



Figure 4.11: Average dynamic power consumption in the complex SPI implementation with the use of FIFO registers or not at different system clock frequencies

The static and dynamic energy consumption by the complex master with and without the FIFO registers implemented are presented in figure 4.12. In figure 4.12a it is seen that the difference in static energy consumption is relatively small as the dots overlap. In figure 4.12b it can be seen that the dynamic energy is reduced with approximately 5-7 pJ for most of the frequencies except the ones where the implementation with FIFO registers has a deviation in dynamic energy consumption. The implementation without the use of FIFO register has no large deviation in the results as the as the dark red dots are relatively aligned.




(a) Static energy




(b) Dynamic energy

Figure 4.12: Energy consumption per SCLK iteration in the complex SPI implementation with the use of FIFO registers or not at different system clock frequencies

The total energy for both the implementation with and without the two FIFO registers are presented in figure 4.13. Here it is seen that for all frequencies the implementation with the FIFO registers use more energy than the implementation without FIFO registers.

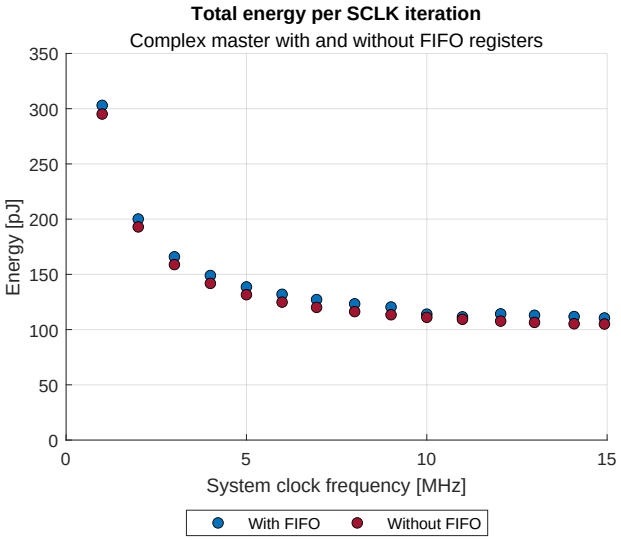



Figure 4.13: Total energy consumption per clock iteration in the complex SPI implementation with the use of FIFO registers or not at different system clock frequencies

The percentage decrease in energy is shown in table 4.4. Here it is seen that the removal of the FIFO registers has a larger impact on the dynamic energy consumption than at the static energy consumption.

Table 4.4: Average energy decrease with removal of FIFO registers

| Energy type | Percentage decrease |
|---|---|
| **Static** [%] | 0.4 |
| **Dynamic** [%] | 6.3 |
| **Total** [%] | 4.6 |

In table 4.5 is the utilized resources for the different implementations of the complex master presented. In the table it can be seen that the removal of the FIFO registers has the an impact on the utilized LUTs and flip flops, but no impact on the utilized I/O-ports.

Table 4.5: Utilized resources in the FPGA with and without FIFO registers. The percentage use of FPGA resources are displayed in parenthesis

| Resource | With FIFO | Without FIFO | Percent decrease |
|---|---|---|---|
| **LUTs** | 169 (0.32%) | 143 (0.27%) | 15.4% |
| **Flip-flops** | 224 (0.21%) | 160 (0.15%) | 28.6% |
| **I/O ports** | 87 (69.60%) | 87 (69.60%) | 0% |

## 4.5 Power optimization

Vivado has as earlier described possibilities for optimizing the utilized logic and do alternations to the design in order to optimize for a lower power consumption. The dynamic power consumption after optimization can be seen in figure 4.14 together with the dynamic power consumption from the system frequency test earlier presented in figure 4.1. In figure 4.14 it can be seen for the complex master that the dynamic power after optimization closely follows the dynamic power without power optimization. It can be seen that the optimized results has a deviation from the linear behavior of the results at frequency 6 MHz and 7 MHz and no deviation at 10 MHz and 11 MHz where the unoptimized implementation has a deviation as previously described. For the simple master it can be seen that it is a deviation from the linear behavior at 15 MHz and also a deviation at 1 MHz and 2 MHz, but the last two deviations are not easily seen in the plot. It can be seen in the plot that the dynamic power consumption after optimization is larger than before power optimization for all frequencies for the simple master.

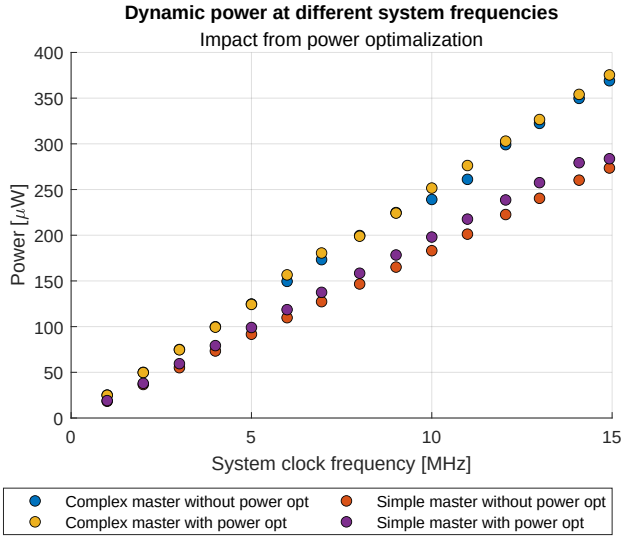
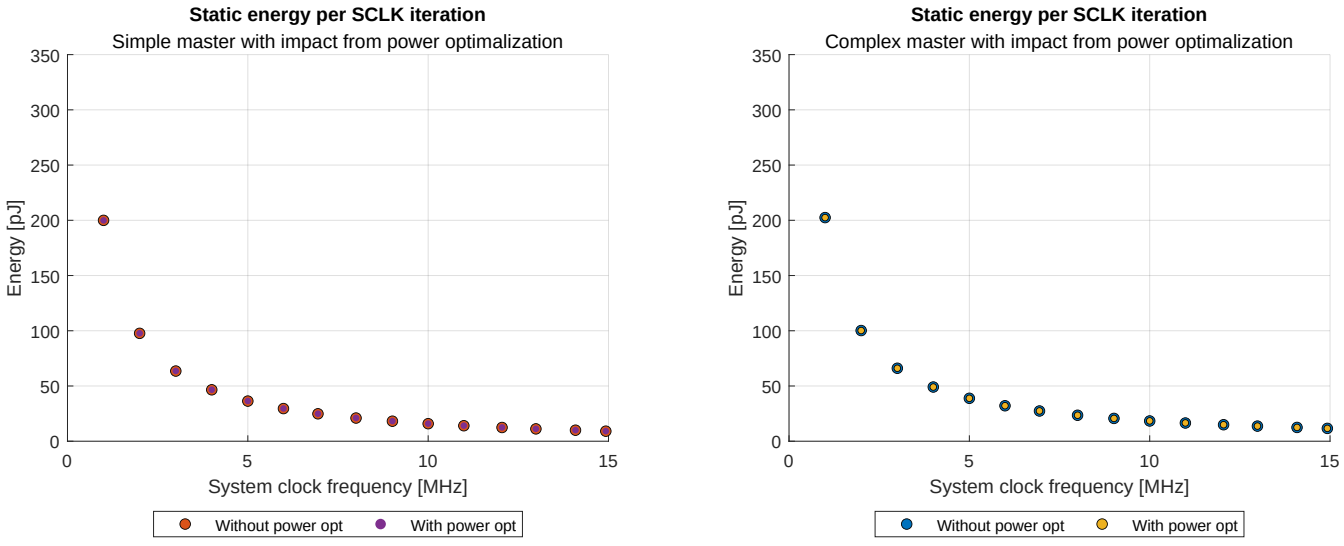



Figure 4.14: Average dynamic power consumption for the different SPI implementations with and without power optimization enabled

The static energy consumption for the different implementations are presented in figure 4.15. In these figures it can be seen that the difference in static energy consumption between the power optimized and not power optimized design is relatively small and not visible in the plots for neither the simple nor the complex implementation.




(a) Simple master

(b) Complex master

Figure 4.15: Static energy per SCLK iteration for the different SPI implementations with and without power optimization enabled

For the dynamic energy consumption in figure 4.16 are the differences in dynamic power consumption more visual. For the complex master it can be seen that for frequencies up to 9 MHz the power optimization gives a slightly lower dynamic energy consumption except for 6 MHz and 7 MHz due to the deviation. For 10 MHz and up it can be seen that the energy optimization gives a larger energy consumption than the unoptimized design. For the simple master it is seen that for all frequencies the energy optimized design gives a larger energy consumption than for the unoptimized implementation. For frequencies 1 MHz, 2 MHz and 15 MHz the dynamic energy consumption after optimization for the simple master is lower than for the other frequencies and deviates from the constant behavior of the other estimates.

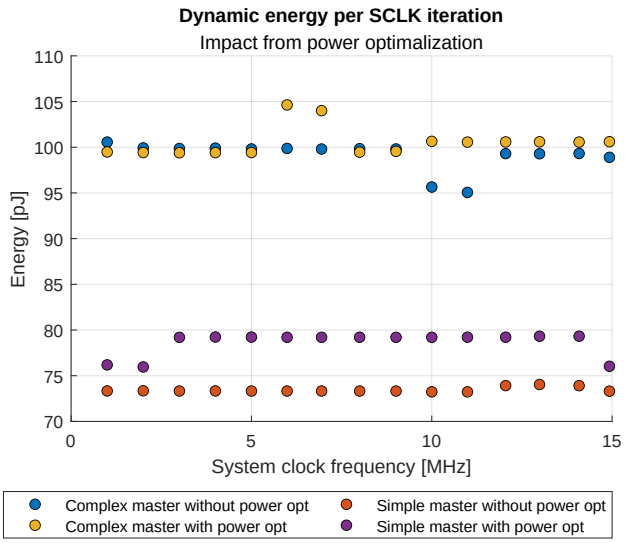



Figure 4.16: Dynamic energy per SCLK iteration for the different SPI implementations with and without power optimization enabled

The total energy consumption per clock iteration for the optimized and unoptimized implementations are presented in figure 4.17 where it can be seen that for the complex master the total difference in energy is relatively small and not visual in the plots except for the frequencies where either of the implementation deviate from the estimation trends. For the simple master it is a small increase in the total energy due to the increased dynamic energy consumption.

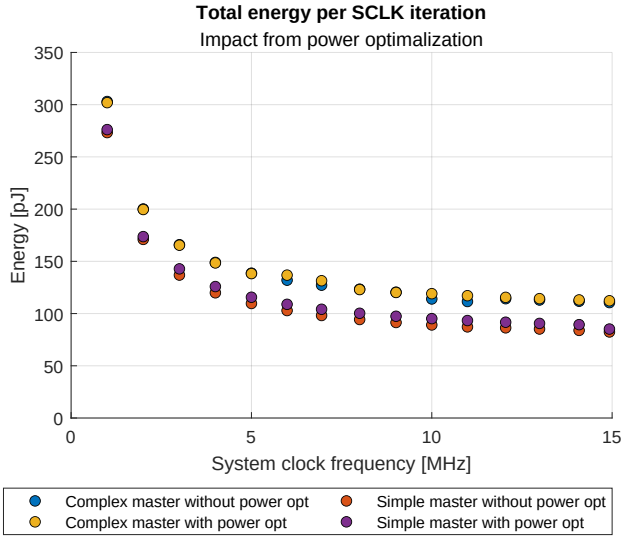



Figure 4.17: Total energy per SCLK iteration for the different SPI implementations with and without power optimization enabled

In table 4.6 can the average decrease in total energy when using power optimization be seen. Multiple of the percentages are negative which means that it actually is an increase in energy which corresponds to the results in the previous figures. The average decrease in energy for the simple master is -5.2% and for the complex master it is -1.3%. As the negative decrease actually is an increase, it can be seen that the simple master has a larger increase in energy than the complex master after optimization.

Table 4.6: Average total energy decrease from power optimizations

| | **Frequency [MHz]** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Implementation** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| **Simple [%]** | -1.0 | -1.7 | -4.3 | -5.0 | -5.5 | -5.8 | -6.1 | -6.3 |
| **Complex [%]** | 0.4 | 0.3 | 0.3 | 0.3 | 0.3 | -3.7 | -3.4 | 0.3 |

| | **Frequency [MHz]** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Implementation** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **Average** |
| **Simple [%]** | -6.5 | -6.8 | -7.0 | -6.2 | -6.3 | -6.5 | -3.4 | -5.2 |
| **Complex [%]** | 0.2 | -4.5 | -5.0 | -1.1 | -1.2 | -1.1 | -1.6 | -1.3 |

The utilized resources for the power optimized design are presented in table 4.7. Based upon the utilized logic for the unoptimized implementations in table 4.2 it can be seen that both the simple and complex master has an increase of 4 LUTs after optimization while the other resources are unchanged.

Table 4.7: Utilized resources in the FPGA with power optimization enabled. The percentage use of FPGA resources are displayed in parenthesis

| Resource | Simple | Complex |
|---|---|---|
| **LUTs** | 64 (0.12%) | 173 (0.33%) |
| **Flip-flops** | 108 (0.10%) | 224 (0.21%) |
| **I/O ports** | 55 (44.00%) | 87 (69.60%) |

# 5 Discussion

The discussion is structured by firstly discussing concerns regarding the estimation methodology before discussing the different results and lastly a brief evaluation of the tool Vivado.

The two implemented SPI masters are analysed as close to their original implementation as possible just with some alternations to enable for comparison as described in subsection 2.7. It is uncertain how well these implementation are optimized regarding power consumption so it could be possible to modify the designs in order to provide a lower power and energy consumption with the same functionality. However for many situations when designing a system, the designer would use premade modules such as IPs to speed up the design process. To modify these device will to some extend go against the reason of utilizing premade designs as it might add development time or the design could be encrypted and unavailable for modifications and the designer must use the design as delivered[7].

## 5.1 Estimation method

When analysing a digital design implemented onto a model of a FPGA, multiple uncertainties could be introduced. For instance may the utilized method introduce challenges such as inaccurate estimations or limitations to how the analysed design actually would work when it is properly implemented as a part of a larger system. When conducting an analysis on a model of an actual FPGA, the fidelity of the model is also of importance since an inaccurate model would give inaccurate power estimations.

Firstly as seen in table 4.2 the amount of utilized resources in the FPGA is below 1% for both the utilized LUTs and flip-flops for both implementations. For designs utilizing such a small amount to the logic, some challenges arises. As described in subsection 2.3.2 may a FPGA have a device static power consumption independent of utilized logic. This could mask out the actual design static power consumption of the design under test as it consists of such a small amount of logic. This challenge is solved by subtracting the device static consumption for the FPGA without the analysed design, but some uncertainty to the presented results are added since the tools not presents the exact details of the contributions to the different static power consumptions. Another result of the issues regarding the small design size can be seen for the estimation of power optimization in subsection 4.5 where the design actually uses more power and energy after optimization. This is further described in subsection 5.6.

Another element affecting the power and energy estimation is the applied timing constraints for the I/O-ports of the design. As described in subsection 3.2.1 the applied timing constraints may have an impact on the power estimations and are important for

the confidence level of the estimations[25]. As also described in subsection 3.2.1 are the correct I/O-constraints for the design not possible to determine without implementing the SPI master as a part of a larger system with the required control logic. The applied testbench creates representative stimuli for the SPI masters, but does not provide the requirements for timing constraints. Another issue regarding the implementation of the SPI masters as stand alone devices is the I/O-ports utilized for the designs as seen in table 4.2. Such I/O-ports could add a fair amount of energy consumption[25] and when the SPI master is implemented as a part of a larger system, the connections to the internal logic will be connected to internal logic instead of an I/O-port and therefore not necessary consume the additional power drawn by the I/O-ports. However the external interface of the SPI master will typically utilize an amount of I/O-ports and it is therefore not possible to just remove the I/O-contributions in the estimations since it is not separated into individual port consumptions. Wiring to other internal logic would also require some amount of energy so the additional I/O-ports can be considered replacements for these connections. These uncertainties could give a change to the estimated energy consumption and the numbers themselves should be considered with care. However, as both of the implementations are analysed with the same limited set of constraints together with alternative I/O-connections, the comparison between the modules is still of interest even if the numbers themselves are uncertain. On the other hand are the values seen to be of the same magnitude as for other SPI masters as stated in section 4.1 which means they might be in the range of the actual consumption.

As presented in section 2.2.1 a lower operating frequency could give the possibility to downsize transistors in the circuit and thereby give a lower power consumption. Since the analysis is conducted at a model of a FPGA with fixed transistor sizes, is the option to downsize the transistors not possible. If the design were implemented onto an ASIC it could be easier to downsize the transistors which could make an impact on the energy consumption for lower frequencies. As mentioned a lower frequency would also give a possibility to use a lower operating voltage $V_{DD}$ for the circuit. It is to some extend possible to alternate the operating voltages for the FPGA, but it has to stay within relatively strict operating conditions[34] so the adjustments for operating voltage are limited. Also as previously mentioned the operating voltage of a system is unlikely to be decided by the optimal operating voltage for the relatively small SPI master and modifications on alternating voltages are therefore not explored.

### 5.1.1 Reliability of estimations

The power estimations of the two SPI masters are as mentioned conducted on a model of the FPGA. When using a model of an actual physical component a concern is always how well the model represents the actual design, i.e. the fidelity of the model. For

the tools utilized in this analysis are the error deviation between the estimated power and actual produced silicon measurements ±20% for the XPE tool and ±10% for the Vivado power estimation[25, 35]. This error is due to process variation. These error deviations strengthens the concern regarding the true values of the power and energy measurements, and should also be considered when looking at the comparison between the two modules since the differences presented in the estimations may differ for an actually implemented SPI master in physical hardware.

## 5.2 System frequency

As seen for the results presented in section 4.1 the overall results show that the complex master has a larger power and energy consumption than the simple master. However it is multiple interesting elements to look further into. Firstly one important element to consider is the mentioned deviation from the overall trends of the results for frequency 10 MHz and 11 MHz for the complex master. The exact reason for this deviation is uncertain, but throughout the different tests and analysis some details are discovered. However since these discoveries are dependent on results from different tests, they are discussed later in this section in subsection 5.7 and the following discussion of the results is therefore based upon the overall trend of the estimations.

The power consumption presented in figure 4.1 has as mentioned an approximately linear behavior which corresponds to the linear frequency dependency of the switching power in equation 2.5. As later presented in figure 4.4 where the dynamic energy consumption from the clocking is extracted, it is seen that the difference between the two implementations with regards of dynamic power and energy consumption is mainly due to the clock switching and generation. As presented in subsection 2.4.1 is a large part of a digital designs total dynamic energy consumption related to clocking, which can be seen in the results where the clocking uses approximately 44% and 56% of the dynamic energy in the simple and complex master respectively. This dynamic energy consumption caused by the clock is in the upper and above the typical energy contributions from the clock presented in section 2.4.1, but due to the nature of a SPI master where the design not only has a dynamic power consumption because of the clock, but also generate the clock SCLK, this can be considered a reasonable estimate. The clocking in the complex master has a larger dynamic energy consumption than for the simple master. This can be due to the larger fanout for the clock networks in the complex master due to the utilization of more hardware. A larger clock fanout requires more wiring which increases the capacitance of the circuit which increases the dynamic power consumption based of the switching power in equation 2.5. For the other elements contributing to the overall dynamic energy consumption presented in figure 4.4a, the difference between the simple and complex master is relatively small. This result show that the amount of utilized logic and I/O ports only slightly directly

affects the dynamic energy consumption. However as more logic requires a larger fanout for the clock network, it contributes to the dynamic energy increase seen for the clocking.

As seen in table 4.1 the percentage increase in energy between the simple and complex master depends on the type of energy. For the static energy consumption it is seen that the percentage increase is smallest for the lowest frequency, but increases for larger frequencies. This can be explained by the nature of the static energy consumption. Both SPI implementations has a constant static power draw which is almost constant at all frequencies which means that the difference between the modules in static power is relatively constant at different frequencies. However, with a higher frequency the period for a clock pulse is shortened and the static energy therefore decreases with frequency. The difference in static energy consumption between the implementations is still constant, but for a higher frequency the difference has a larger share of the implementations static energy consumption at that frequency. For the dynamic energy consumption it can be seen that the percentage increase is relatively stable at the different frequencies only with some small variations. What can be seen for this comparison is that the energy cost in implementing the complex master increases with higher frequencies since for an operating frequency of 1 MHz the additional energy for the complex master is 10.9% while for 15 MHz it is 34.1%.

The complex master requires more than twice as many LUTs and flip-flops than the simple master as seen in table 4.2, but does not require twice as much energy. This could be due to a lot of the added logic in the complex master is used to provide a flexible design where transmission settings such as setup and hold time can be changed after implementation. The switching however in these logic block are relatively limited which can explain why twice as much logic does not require twice as much energy. The complex master also has approximately 10 times more SLOCs than the simple master. This is not an elements who directly affects the energy consumption as the content in these lines may invoke different amounts of logic. However it presents another consideration with exploring the requirements for the design. The complex master has more functionality than the simple master, but it also requires a more complex design process.

## 5.3 SCLK division

The test with different divisions of the SCLK gives a set of interesting changes to look at both between the SPI masters, but also how the implementations themselves reacts to the different divisions. Firstly it is seen for the dynamic power in figure 4.5 that lowest SCLK division of 4 has the largest dynamic power consumption. Since a lower division of the system clock gives a higher frequency of the SCLK it is natural that it has the largest power consumption as seen from equation 2.5 for the switching power.

The power consumption by the complex master is also higher for all divisions than the simple master which also is natural based on the discussion of the results for the system frequency test.

When looking at the energy consumption in figure 4.6 and 4.7 it is seen that the division of 32 has the largest energy consumption and the order is flipped compared to the dynamic power plot. This is not necessary surprising since the energy is given per SCLK iteration and the period of the on-time of the circuit grows when the division is larger. It is therefore seen that the reduction in power consumption for the 32 division is not enough to counter the added on-time and the 32-division therefore uses the most energy of the tested divisions. The energy consumption almost doubles between each division which is natural as the SCLK frequency is halved between each division.

The percentage increase in energy for each of the SPI implementations from division 4 to the other divisions in table 4.3 are seen to be very similar which show that the two implementations reacts similarly to the modifications in SCLK division and that the type of implementation therefore does not affects the percentage increase in energy. However as the complex master has a larger amount of energy consumption than the simple master, the consequence of an increase in energy is larger for the complex master in a system with limited energy resources.

As presented in subsection 5.1 in the beginning of the discussion, the complete benefit in energy consumption of running the design on a lower frequency is not achieved due to the fixed transistor size and limitations in operating voltage. This is also a consideration for the SCLK frequency. As the SCLK frequency for some SPI slaves could be used to clock the SPI slave during transmission, a slower SCLK clock can save energy in the SPI slave which can make it beneficial to use a higher SCLK division. However the effect of this is not explored in this thesis and is therefore just a consideration.

## 5.4 Alternation of operating mode

The results from the tests where the implementations are analysed using different operating modes show that the choice in operating mode has a relatively small impact on the energy consumption in the SPI masters. According to the theory this is not a surprising result as both the transmission time, switching activity and other parameters are similar for all the modes just with some alternations in when the data is sampled and at which clock flank.

However some difference are seen in the results which does not have a large impact on the energy consumption, but gives a picture of the nature of the implementations. For the static energy consumption in figure 4.9 it is no clear difference either between the implementations nor the different modes and the results are similar to the ones described for the system frequency test. For the dynamic energy consumption in figure

4.10 some difference can be seen. For the simple master the different operating modes does not have an impact on the energy consumption, but for the complex master some variations between the operating modes is seen. This variation however can be considered relatively small as the average difference between the mode using the most dynamic energy and the one using the least amount is 0.15 pJ. Weather it is the size of the implementations or how the functionality for modifying CPOL and CPHA is designed who causes the difference seen in the implementations are uncertain, but the results at least presents that how additional functionality is implemented may have an impact on the systems performance.

## 5.5 Internal changes to the complex master

The removal of the FIFO registers give as seen not surprisingly a lower power and energy consumption since the implementation utilizes less logic than the implementation with FIFO registers. Most of this energy reduction is in the dynamic energy even with a significant reduction in utilized logic. A FIFO register may introduce a large amount of switching activity due to a lot of data switching which naturally is removed when the FIFO register is removed.

The removal of the FIFO registers give as seen in figure 4.13 a design with a lower energy consumption. However since the SPI master always will be a part of a larger system, the removal of the FIFO registers does not necessary gives a lower energy consumption for the overall system. Without the FIFO registers the controlling logic will have to regularly send or read data to and from the SPI master while with the use of FIFO registers, the controlling logic can send or read a large amount of data in one operation. This could free the controlling logic which could do other operations or utilize other power savings techniques. The actual energy savings for an overall system is therefore important with the consideration the use of FIFO registers and will have to be further looked into upon implementation of a complete system.

## 5.6 Power optimization

When analysing the power optimized implementations the result where surprising as the implementations turned out to have a larger or relatively similar energy consumption as the unoptimized designs. For the complex master the optimized design required both slightly less and slightly more energy depending on the system frequency while for the simple master the optimized implementation required more energy at all frequencies as seen in figure 4.17. The optimized designs introduces 4 more LUTs to the implementation, but this small increase in logic should not necessary give such a large increase in energy based upon the results from the tests without the FIFO registers. It can be seen by the results in figure 4.15 and 4.16 that the main difference in energy is related to the dynamic energy consumption.

In Vivado the the key element of the power optimization technique is to add internal clock gating to parts of the designs. This addition of extra logic and controlling of the clock gates require some amount of extra energy as described in section 2.2.2. As stated in subsection 5.1 in the beginning of this section the SPI masters can be considered as relatively small designs and utilizes a small amount of the available logic. This creates an issue where the tool adds extra logic for clock gating, but the gated elements are so small that the gated logic does not save enough energy in order to compensate for the added overhead in energy from clock gating. It is seen in table 4.6 that for the complex master the average increase in energy is smaller than for the simple master. The complex master consists of a larger amount of logic than the simple master and therefore has a larger potential to clock gate larger logic blocks.

## 5.7 Deviating results

It can be seen in multiple of the results that some estimations for the dynamic consumption deviate from the overall trends of the estimates. It is mostly seen for the complex master, but is also shown after power optimization for the simple master. The exact reasons for these deviations is unknown, but it is seen for different frequencies throughout the analysis. For instance in the system frequency test a closer analysis for surrounding frequencies is conducted and the deviation is seen to occur at a range of frequencies from slightly lower than 10 MHz to slightly higher than 11 MHz, and not just for the exact integer frequencies as presented in figure 4.2b. The deviation is also seen to occur at different frequencies when the different tests are applied and also deviates in both directions so for some frequencies the consumption is larger, and for some it is lower.

As seen in figure 4.4b the deviation is seen to be generated from the dynamic energy consumption by clocking. For the complex master it is also seen that with the removal of the FIFO register the deviation does not occur as presented in figure 4.12b. After analysis of the power consumption in the complex design, it is seen that the deviation does not directly occur in the FIFO registers, but as a part of the datapath in the design where the registers are connected. For the simple master some deviations are also seen for the power optimized design which confirms that the deviation is not only due to the FIFO registers.

The estimation deviations are therefore caused by energy consumption due to clocking in the data path, but unfortunately no further reason for the deviation is discovered. However, since for all the results the majority of the estimations follow the same pattern where only a few estimates deviate, the analysis is based on the general patterns of the results. Therefore some considerations must be taken that the deviating results either may be wrong, or that some frequencies match the design better or worse than others to give a lower or higher energy consumption. A closer examination of these deviating

results could therefore be of interest for further studies.

## 5.8 Tool evaluation

As seen throughout this thesis the applied tool Vivado has a large impact on both how the designs are implemented onto the hardware as well as the method for estimation of the power consumption. This means that the tool has a large impact on the results of this thesis and should therefore also be considered for evaluation.

One of the most important aspects when analysing the power parameters of the design is naturally the reported power consumption. Vivado gives to some extend details around the power consumption on a module basis or utilization level such as power consumption in different logic types. A further detailed view of the power consumption in each of the components are not discovered. This could for instance been interesting to look into in order to further analyse the reason for the deviations in the estimations. Exact details regarding the contributions to the different power estimations such as what the power consumption from for instance clocking actually includes, are also limited which introduces some uncertainty of what the reported power actually shows.

The procedures and methods of extracting the necessary power estimations are based upon the user guides for the Vivado environment. These user guides give a good explanation of how the tool functions and how to set various settings. However the consequence and meaning of the different settings are rarely described and the method provided in this thesis is therefore based upon trial, error and considerations of what can be considered reasonable results for the estimations. Since Vivado is a complex tool with a large amount of possibilities, some considerations must be taken that there might be better methods or tools to conduct the presented analysis than the one presented in this thesis.

The reported power estimations also depend on how well the given RTL description maps down to the FPGA device. How well this is done depends not only on the tools quality, but also the author of the RTL as the RTL can can be written in different ways with different mappings to the FPGA even with same behavior. Different mappings could consist of different amounts of logic which naturally will require different amounts of power.

# 6 Conclusion

As discussed in the previous section the two different implementations of the SPI master has a difference in their requirement for energy, but they also responds similarly to applied modifications. What can be seen is that the complex master implementation has a larger power and energy consumption than the simple master for all conducted tests. When tested over a frequencyspan over 1 MHz to 15 MHz it is seen that on average the complex master requires 27.2% more energy than the simple master when all other parameters are unchanged. This increase in energy is mainly caused by the differences in dynamic energy due to clocking in the design. The complex master utilize more than twice as much logic as the simple master, which increases the energy consumption due to a larger capacitance in the circuit from both the logic and the wiring in between. It is seen that the percentage energy cost in adding the additional functionality for the complex master rises with frequency since at 1 MHz the complex master only requires 10.9% more energy while at 15 MHz it requires 34.1% more.

The complex master includes a variety of addition functionality such as access to the current status of the transmission and the possibility of adjusting transmission setting during operation. The extra energy consumption is therefore the cost of the additional functionality and a trade of between energy consumption and functionality might therefore be required when implementing a SPI master into a system. However it is seen that the complex master does not double the energy consumption even with the double amount of logic, so the energy cost in additional functionality heavily depends on the switching activity in the added logic. The two implementations still responds relatively similar to adjustments made to the design where the percentage increase in energy with different SCLK divisions applied are relatively similar.

It is seen that within the tested operating frequencies it is preferable to operate the design using the largest system clock as possible with a low division to the communication clock SCLK since this would give the lowest total energy consumption, but with the consideration that due to the implementation of the design on a FPGA, some benefits of operating the design at a low frequency such as smaller transistor-sizes and lower operating voltage were not possible.

It is seen throughout this analysis that the method of analysis has certain limitations. The SPI master will always be a part of a large system and it is therefore to some extend not profitable to look at it as a standalone device. Especially can this be concerned when the design is implemented onto a FPGA since the rest of the system can largely affect the energy consumption of the SPI master. The SPI masters implemented here are also of relatively small size and when implemented onto a large FPGA compared to the SPI designs, the resulting estimations are affected by a static overhead in the FPGA. This can however to some extend be corrected, but as seen when the design is

power optimized during implementation, the tool overestimate its optimization due to the small design and actually cause the energy to increase when the goal is a decreased energy consumption.

# 7 Further work

This thesis could to some extend be considered a preliminary study due to the different limitations in the analysis. In order to have an exact analysis of the true energy consumption for the different implementations of the SPI master, the SPI master must be implemented as a part of a combined system with control logic in order to create a more realistic environment.

## 7.1 SPI

As presented in this thesis there is multiple adjustments who could be introduced to conduct a better analysis of the SPI masters. Since an ASIC device has a lower energy consumption than the FPGA due to its customizable possibilities it can be the desired choice when implementing a ultra low power device. It could therefore be interesting to look into how different SPI implementations would behave when implemented onto an ASIC device. This would also eliminate many of the difficulties encountered in this thesis such as additional device static power and could give a more accurate estimation of the true consumption of the SPI master devices.

## 7.2 Considerations regarding MBus

As stated in the introduction it is a large variety of different possibilities when it comes to the choice in which communication protocol to use. Most of these are well documented in a variety of different resources online, but in the research during this thesis additional updated information regarding the communication protocol Mbus[4][4] are discovered and some of it is presented here since the publicly available documentation and design-files are outdated. Hopefully this information can be useful to others exploring the possibilities of this protocol.

As stated in the published and available online research, this communication protocol is designed to function as a competitor to the I2C-protocol, but with a significant reductions in power and energy consumption with promising results. The available public presence is not currently updated and the newest version of the protocol can be sort of considered more a platform where the entire system must be designed from ground around the MBus system. The MBus is therefore no longer just an interconnection between modules in a system as presented in the available public research. The protocol has also been highly customized for other system designed by the creators of MBus and no other systems can effectively utilize the beneficial power consumption of the MBus[36]. The utilization of the newest version of the MBus protocol is therefore impossible to implement without further communication with the creators and

[4]This should not be mixed with the more common M-Bus(Meter-bus) as these as completely different communication systems.

access to the implementation files. Due to their commercialization of systems utilizing the MBus as a core component, the possibilities of access to further details regarding the protocol are unknown, but the platform may be distributed in the future if it is considered beneficial by the designers[36].

# References


[1] Ahmadreza Motaqi. "Energy-performance management in battery powered reconfigurable processors for standalone IoT systems". In: *International Journal of Information Technology* 12.3 (Sept. 2020), pp. 653–668. ISSN: 2511-2112. DOI: 10.1007/s41870-020-00454-4. URL: https://doi.org/10.1007/s41870-020-00454-4.

[2] Peter Newman. *THE INTERNET OF THINGS 2020: Here's what over 400 IoT decision-makers say about the future of enterprise connectivity and how IoT companies can use it to grow revenue*. Mar. 2020. URL: https://www.businessinsider.com/internet-of-things-report?IR=T (visited on Dec. 13, 2020).

[3] European Environment Agency. *Environmental impact of energy*. URL: https://www.eea.europa.eu/help/glossary/eea-glossary/environmental-impact-of-energy (visited on June 5, 2021).

[4] P. Pannuto et al. "MBus: An ultra-low power interconnect bus for next generation nanopower systems". In: *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*. 2015, pp. 629–641.

[5] Richard Mc Sweeney, Christian Spagnol, and Emanuel Popovici. "Comparative study of software vs. hardware implementations of shortened Reed-Solomon code for Wireless Body Area Networks". In: *2010 27th International Conference on Microelectronics Proceedings*. 2010, pp. 223–226. DOI: 10.1109/MIEL.2010.5490471.

[6] Markus Rud. *Evaluation of digital serial communication for a flexible sensor interface*. Project Thesis in Electronics at NTNU. Dec. 2020.

[7] Khaled Salah Mohamed. *IP Cores Design from Specifications to Production*. Springer International Publishing Switzerland, 2016.

[8] Neil H. E. Weste and David Money Harris. *Integrated Circuit Design*. Forth edition. Pearson Education, 2011.

[9] Smitha Sundaresan and Frederic Rivoallon. *Analysis of Power Savings from Intelligent Clock Gating (XAPP790)*. v1.0. Xilinx. Aug. 2012.

[10] Shmuel Wimer and Israel Koren. "The Optimal Fan-Out of Clock Network for Power Minimization by Adaptive Gating". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 20.10 (2012), pp. 1772–1780. DOI: 10.1109/TVLSI.2011.2162861.

[11] Julien Lamoureux and Wayne Luk. "An Overview of Low-Power Techniques for Field-Programmable Gate Arrays". In: *2008 NASA/ESA Conference on Adaptive Hardware and Systems*. 2008, pp. 338–345. DOI: 10.1109/AHS.2008.71.

[12] *Vivado Design Suite User Guide; Design Flows Overview (UG892)*. v2020.2. Xilinx. Feb. 2021.

[13] HardwareBee. *The Ultimate Guide to FPGA Design Flow*. URL: https://hardwarebee.com/ultimate-guide-fpga-design-flow/.html (visited on May 9, 2021).

[14] *Power Methodology Guide (UG786)*. v14.5. Xilinx. Apr. 2013.

[15] Ian Kuon and Jonathan Rose. "Measuring the Gap Between FPGAs and ASICs". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 26.2 (2007), pp. 203–215. DOI: 10.1109/TCAD.2006.884574.

[16] I. Brynjolfson and Z. Zilic. "Dynamic clock management for low power applications in FPGAs". In: *Proceedings of the IEEE 2000 Custom Integrated Circuits Conference (Cat. No.00CH37044)*. 2000, pp. 139–142. DOI: 10.1109/CICC.2000.852635.

[17] Alireza Rakhshanfar and Jason H. Anderson. "An integer programming placement approach to FPGA clock power reduction". In: *16th Asia and South Pacific Design Automation Conference (ASP-DAC 2011)*. 2011, pp. 831–836. DOI: 10.1109/ASPDAC.2011.5722305.

[18] *SPI Block Guide V04.01*. Motorola, Inc. July 2004.

[19] DigiKey Electronics (Scott_1767). *SPI Master (VHDL)*. Mar. 2021. URL: https://forum.digikey.com/t/spi-master-vhdl/12717 (visited on Apr. 26, 2021).

[20] *Vivado Design Suite User Guide: Synthesis (UG901)*. v2020.2. Xilinx. Jan. 2021.

[21] *SPI Master Controller*. FPGA-RD-02174-1.1. Lattice Semiconductor. Feb. 2020. URL: https://www.latticesemi.com/products/designsoftwareandip/intellectualproperty/referencedesigns/referencedesign03/spimastercontroller.

[22] *Zynq-7000 SoC Data Sheet: Overview*. DS190 (v1.11.1). Xilinx. July 2018.

[23] Xilinx. *Vivado Design Suite - HLx Editions*. URL: https://www.xilinx.com/products/design-tools/vivado.html (visited on May 25, 2021).

[24] *Vivado Design Suite User Guide; Using Constraints (UG903)*. v2018.1. Xilinx. Apr. 2018.

[25] *Vivado Design Suite User Guide; Power Analysis and Optimization (UG907)*. v2020.2. Xilinx. Nov. 2020.

[26] *nRF24L01+ Single Chip 2.4GHz Transceiver Product Specification*. v1.0. Nordic Semiconductor. Sept. 2008.

[27] *UltraFast Design Methodology Guide for the Vivado Design Suite (UG949)*. v2020.2. Xilinx. Feb. 2021.

[28] *Vivado Design Suite User Guide: Logic Simulation (UG900)*. v2020.2. Xilinx. Nov. 2020.

[29] Xilinx. *AR# 57127: Vivado Simulator - Post Synthesis and Post Implementation Timing simulation options are greyed out in my VHDL Vivado project, how can I run VHDL timing simulations?* May 2014. URL: https://www.xilinx.com/support/answers/57127.html (visited on May 1, 2021).

[30] Ruzica Jevtic and Carlos Carreras. “Power Measurement Methodology for FPGA Devices”. In: *IEEE Transactions on Instrumentation and Measurement* 60.1 (2011), pp. 237–247. DOI: 10.1109/TIM.2010.2047664.

[31] Xilinx. *Xilinx Power Estimator (XPE)*. URL: https://www.xilinx.com/products/technology/power/xpe.html (visited on May 25, 2021).

[32] *Vivado Design Suite Tcl Command Reference Guide (UG835)*. v2020.2. Xilinx. Nov. 2020.

[33] Christopher J. Lukas and Benton H. Calhoun. “A 0.38 pj/bit 1.24 nW chip-to-chip serial link for ultra-low power systems”. In: *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2015, pp. 2860–2863. DOI: 10.1109/ISCAS.2015.7169283.

[34] *Zynq-7000 SoC (Z-7007S, Z-7012S, Z-7014S, Z-7010, Z-7015, and Z-7020): DC and AC Switching Characteristics*. DS187 (v1.21). Xilinx. Dec. 2020.

[35] Xilinx. *AR# 59456: 2015.2 Vivado Power - What is the deviation between dynamic power analysis and the actual production silicon measurements?* URL: https://www.xilinx.com/support/answers/59456.html (visited on May 28, 2021).

[36] *Email correspondence with Yejoong Kim, senior research engineer at the Univ of Michigan and related to the creation of MBus*. Feb. 2021.

# A Simulation/estimation parameters

This appendix presents multiple parameters and settings who have been set and used during the power estimations.

### Testbench parameters/data

Table A.1: Transmitted/received data in testbench

| Data |
|---|
| 16'b 0011100000011000 |
| 16'b 0000000000000001 |
| 16'b 1000000000000000 |
| 16'b 1111111111111111 |
| 16'b 0010101010101010 |
| 16'b 0100110011001101 |
| 16'b 1111000011111111 |
| 16'b 1111111111111110 |
| 16'b 0111111111110000 |
| 16'b 0000111111110001 |
| 16'b 1111111111111111 |
| 16'b 1000000000000000 |
| 16'b 0010101010101010 |
| 16'b 1111111111111111 |
| 16'b 1111000011100000 |
| 16'b 1111111111111110 |

### Power estimation parameters

Table A.2: Environmental parameters in power estimation

| Parameter | Setting/Value |
|---|---|
| **Temp grade** | Commercial |
| **Process** | Typical |
| **Output Load** | 0 pF |
| **Ambient temperature** | 25 °C |
| **Airflow** | 250 LFM |
| **Heat sink** | None |
| $\theta$**SA** | 0°C/W |
| **Board selection** | Medium (10"x10") |
| **Number of board layers** | 8 to 10 layers |
| $\theta$**JA** | 11.5°C/W |
| **Board temperature** | 25°C |

Table A.3: Voltage parameters in power estimation based on typical values for the operating conditions in the datasheet for the SoC[34]

| Source | Voltage [V] |
|---|---|
| **Vccint** | 1.000 |
| **Vccaux** | 1.800 |
| **Vcco33** | 3.300 |
| **Vcco25** | 2.500 |
| **Vcco18** | 1.800 |
| **Vcco15** | 1.500 |
| **Vcco135** | 1.350 |
| **Vcco12** | 1.200 |
| **Vccaux_io** | 1.800 |
| **Vccbram** | 1.000 |
| **MGTAVcc** | 1.000 |
| **MGTAVtt** | 1.200 |
| **MGTVccaux** | 1.800 |
| **Vccpint** | 1.000 |
| **Vccpaux** | 1.800 |
| **Vccpll** | 1.800 |
| **Vcco_ddr** | 1.500 |
| **Vcco_mio0** | 1.800 |
| **Vcco_mio1** | 1.800 |
| **Vccadc** | 1.800 |

NTNU

Norwegian University of
Science and Technology