UNIVERSITY OF OULU

FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

DEGREE PROGRAMME IN ELECTRONICS AND COMMUNICATIONS ENGINEERING

MASTER'S THESIS

DESIGN FOR TESTABILITY OF A LATCH-BASED DESIGN

Author	Matti Leinonen
Supervisor	Jukka Lahti
Second Examiner	Tarmo Ruotsalainen
Technical Advisor	Joni Jäntti

JUNE 2022

Leinonen M. (2022) Design for testability of a latch-based design. University of Oulu, Faculty of Information Technology and Electrical Engineering, Degree Programme in Electronics and Communications Engineering. Master's Thesis, 70p.

ABSTRACT

The purpose of this thesis was to decrease the area of digital logic in a power management integrated circuit (PMIC), by replacing selected flip-flops with latches. The thesis consists of a theory part, that provides background theory for the thesis, and a practical part, that presents a latch register design and design for testability (DFT) method for achieving an acceptable level of manufacturing fault coverage for it.

The total area was decreased by replacing flip-flops of read-write and one-time programmable registers with latches. One set of negative level active primary latches were shared with all the positive level active latch registers in the same register bank. Clock gating was used to select which latch register the write data was loaded to from the primary latches. The latches were made transparent during the shift operation of partial scan testing. The observability of the latch register clock gating logic was improved by leaving the first bit of each latch register as a flip-flop. The controllability was improved by inserting control points.

The latch register design, developed in this thesis, resulted in a total area decrease of 5% and a register bank area decrease of 15% compared to a flip-flop-based reference design. The latch register design manages to maintain the same stuck-at fault coverage as the reference design.

Key words: manufacturing testing, partial scan testing, power management integrated circuit, area, power consumption and fault coverage.

Leinonen M. (2022) Salpaperäisen piirin testattavuuden suunnittelu. Oulun yliopisto, tieto- ja sähkötekniikan tiedekunta, elektroniikan ja tietoliikennetekniikan tutkinto-ohjelma. Diplomityö, 70p.

TIIVISTELMÄ

Tämän opinnäytetyön tarkoituksena oli pienentää digitaalisen logiikan pinta-alaa integroidussa tehonhallintapiirissä, korvaamalla valitut kiikut salpapiireillä. Opinnäytetyö koostuu teoriaosasta, joka antaa taustatietoa opinnäytetyölle, ja käytännön osuudesta, jossa esitellään salparekisteripiiri ja testattavuussuunnittelun menetelmä, jolla saavutettiin riittävän hyvä virhekattavuus salparekisteripiirille.

Kokonaispinta-alaa pienennettiin korvaamalla luku-kirjoitusrekistereiden ja kerran ohjelmoitavien rekistereiden kiikut salpapiireillä. Yhdet negatiivisella tasolla aktiiviset isäntä-salpapiirit jaettiin kaikkien samassa rekisteripankissa olevien positiivisella tasolla aktiivisten salparekistereiden kanssa. Kellon portittamisella valittiin mihin salparekisteriin kirjoitusdata ladattiin yhteisistä isäntä-salpapireistä. Osittaisessa testipolkuihin perustuvassa testauksessa salpapiirit tehtiin läpinäkyviksi siirto-operaation aikana. Salparekisterin kellon portituslogiikan havaittavuutta parannettiin jättämällä jokaisen salparekisterin ensimmäinen bitti kiikuksi. Ohjattavuutta parannettiin lisäämällä ohjauspisteitä.

Salparekisteripiiri, joka suunniteltiin tässä diplomityössä, pienensi kokonaispinta-alaa 5 % ja rekisteripankin pinta-alaa 15 % verrattuna kiikkuperäiseen vertailupiiriin. Salparekisteripiiri onnistuu pitämään saman juuttumisvikamallin virhekattavuuden kuin vertailupiiri.

Avainsanat: tuotannon testaus, osittainen testipolkuihin perustuva testaus, integroitu tehonhallintapiiri, pinta-ala, tehonkulutus ja virhekattavuus.

TABLE OF CONTENTS

FOREWORD

The objective of this thesis was to reduce the digital area in a power management integrated circuit by replacing selected flip-flops with latches. The negative effects of the replacement on power consumption and manufacturing testing were to be minimized. The thesis was carried out at Nordic Semiconductor Finland from January 2022 to June 2022.

I would like to thank my manager Joni Jäntti for acting as the technical advisor for the thesis, providing the subject for the thesis and his invaluable comments and guidance during the thesis work. I would also like to thank University Lecturer Jukka Lahti for supervising the thesis and for his great work teaching several courses that provided helpful background knowledge for the thesis. Lastly, I would like to thank D.Sc. Tarmo Ruotsalainen for being the second examiner for the thesis.

Oulu, June 15, 2022

Matti Leinonen

LIST OF ABBREVIATIONS AND SYMBOLS

AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
ATE	Automatic test equipment
ATPG	Automatic test pattern generation
BIST	Built-in self-test
CF	Coupling fault
CMOS	Complementary metal-oxide-semiconductor
CUT	Circuit under test
DFT	Design for testability
DRAM	Dynamic random-access memory
FSM	Finite-state machine
I/O	Input or output
I2C	Inter-integrated circuit
IC	Integrated circuit
IDDQ	Leakage current
IoT	Internet of things
LDO	Low-dropout
LFSR	Linear feedback shift register
LOCST	LSSD on-chip self-test
LSSD	Level-sensitive scan design
MISR	Multiple-input signature register
MSB	Most significant bit
ORA	Output response analyser
OTP	One-time programmable
PMIC	Power management integrated circuit
PRPG	Pseudo-random pattern generator
ROM	Read-only memory
RTL	Register-transfer level
SAF	Stuck-at fault
SAIF	Switching activity information format
SISR	Single-input signature register
SoC	System on chip
SRAM	Static random-access memory
STUMPS	Self-testing using MISR and parallel shift register sequence generator
TF	Transition fault
TPG	Test pattern generator

a	Switching activity factor
C_L	Load capacitance
f	Clock frequency
I_{lk}	Leakage current
I_{sc}	Short-circuit current
P_{sc}	Short-circuit power consumption
P_{st}	Static power consumption
P_{sw}	Switching power consumption
t_{borrow}	Time borrowed

T_c	Clock period
t_{cd}	Contamination delay
t_{cqcd}	Clock-to-Q contamination delay
t_{cqpd}	Clock-to-Q propagation delay
t_{dqcd}	D-to-Q contamination delay
t_{dqpd}	D-to-Q propagation delay
t_{hold}	Hold time
t_{pd}	Propagation delay
t_{pw}	Pulse width
t_{setup}	Setup time
t_{skew}	Clock skew
V_{dd}	Supply voltage
max(a,b)	Maximum of a and b

1 INTRODUCTION

The emergence of the internet of things (IoT) phenomenon has brought an explosion of lightweight sensors being placed in all kinds of objects and sharing the sensor data wirelessly. For the analogue, digital, and mixed signal circuits of these IoT devices to work properly, they should be provided with clean and correct level supply voltages. Such voltages are converted from raw supplies, like batteries, with power management integrated circuits (PMICs).

A simple PMICs can be just a low-dropout (LDO) regulator, that converts the battery voltage to a clean and correct level supply voltage. Most PMICs are much more complicated, often offering safety and control features. These more complex features are often implemented, at least in part, with digital logic.

The cost, size, and power consumption are important characteristics of PMICs. These characteristics are however often in conflict with each other. Integrated circuits (ICs) have traditionally incorporated ever increasing amounts of logic on smaller areas by moving to smaller and smaller technology nodes. PMICs have not been able to fully harness the area benefits of these smaller nodes due to their power requirements. PMICs need to be able to handle all the current going through them that go to the rest of the system. Transistors with large currents going through them need to be made large, regardless of the technology size. Making large transistors in small technologies is more expensive than in larger technologies. The power density also needs to be kept at an acceptable level. If the PMIC is made too small, too much power can be concentrated on a small area, leading to high temperatures. Another consideration, especially for battery powered applications, is the quiescent current consumption. The subthreshold leakage current generally increases when the channel length of a transistor decreases in the smaller technologies.

Another option to decrease the area is to implement the same functionality with fewer transistors. Most of the digital logic area consists of D flip-flops. From the area point of view, large area savings are available if they were replaced with D latches, since D latches are roughly half the size of D flip-flops. Replacing flip-flops with laches would result in either smaller area, and reduced chip cost, or more functionality in the same area, and increased chip value. The area of digital logic directly affects the size and cost of a product. With large volumes, even a slight cost decrease can produce large profits.

The main objective of the thesis is to decrease the area of digital logic in a PMIC, by replacing selected flip-flops with latches. The latch replacement should not significantly increase the power consumption of a PMIC. The latch replacement should also not significantly affect the quality of manufacturing testing.

The scope of the thesis is as follows:

- Literature review of flip-flop and latch characteristics is carried out to help in selecting flip-flops that are suited to be replaced with latches or pulsed latches in a PMIC.
- Literature review of design for testability (DFT) methods is carried out to help in selecting a suitable DFT method for the latch based PMIC design.
- Selecting and implementing a suitable latch based PMIC design, based on findings from the literature review, and the structure of a reference PMIC.
- Selecting and implementing a suitable DFT method, based on findings from the literature review and the structure of the latch based PMIC.

The thesis is organized into the following chapters:

- **Chapter 1** introduces the thesis and describes the motivation and objectives for the thesis.
- **Chapter 2** presents the background theory of sequential logic.
- **Chapter 3** presents background theory of design for testability methods.
- **Chapter 4** presents a latch register design using partial scan as the DFT method, that reduces the area of a flip-flop-based reference PMIC design.
- **Chapter 5** presents the area, power consumption and automatic pattern generation results for the latch register design, compared against the flip-flop-based reference design.
- **Chapter 6** provides discussion on the practical work, the results, and possible future work.

2 SEQUENTIAL LOGIC

Digital circuits consist of combinational and sequential logic. Combinational logic consists of logic gates, such as NAND and NOR gates. The outputs of combinational logic depend only on the states of its inputs. Unlike combinational logic, the outputs of sequential logic also depend on the previous state of its inputs through feedback. Sequential logic has memory and is made of flip-flops and latches. [1, 2]

The distinction between latches and flip-flops is not always completely clear. Edge triggered circuits are considered flip-flops in [1] and [2], while level triggered circuits are considered latches. On the other hand, level triggered circuits are considered flip-flops in [3]. This thesis considers edge triggered circuits to be flip-flops, where the output changes on the positive or negative edge of the clock signal. Level triggered circuits are considered synchronous latches, where the output changes when the clock signal is either high or low. Asynchronous latches do not have a clock input and the output changes when the inputs are changed.

2.1 Latches

Two cross-coupled inverters form a building block of sequential circuits, a bistable element. The element, in Figure 1, has zero inputs and two complementary outputs, Q and $\overline{Q}$. The outputs have two stable states, Q=0 and $\overline{Q}$=1 or Q=1 and $\overline{Q}$=0, which makes the element bistable. It can store one bit of information, but it is not very useful as it has no inputs to control its state. [1, 2]

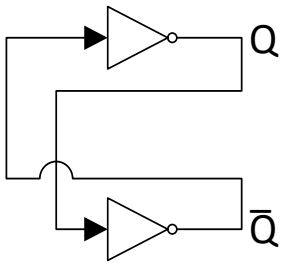

Figure 1. Cross-coupled inverter pair.

Replacing the two cross-coupled inverters with two NAND gates, like in Figure 2, or two NOR gates, like in Figure 3, gives us the SR latch. The state of the SR latch can now be controlled though the S and R inputs. The difference between the NAND and NOR implementations is the polarity at which the inputs are asserted active. The NOR implementation has active high inputs, and the NAND implementation has active low inputs. The S input sets the output Q high, and the R input resets it low. If neither the S nor R input is asserted, the output keeps its previous state. Having both inputs asserted at the same time, would lead to both outputs, Q and $\overline{Q}$, having the same value. It would mean the two outputs are no longer complementary. If both inputs are then de-asserted at the same time, the next output will be unpredictable. For these reasons, having both inputs asserted at the same time is an invalid condition. [1, 2, 3]

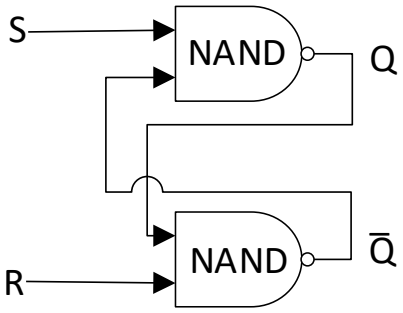

Figure 2. Asynchronous SR latch with two NAND gates.

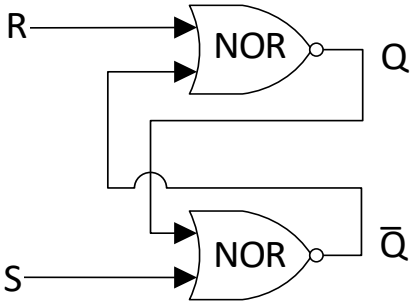

Figure 3. Asynchronous SR latch with two NOR gates.

The SR latch presented above is asynchronous by nature, the change in input is seen immediately in the output. It can be made synchronous by slightly modifying it and adding a clock input, like in Figure 4. The clock, ck, signal can now be used to control the transparency of the latch. When ck is low, the outputs of the first two NAND gates will be high, regardless of the values of S and R and the latch keeps its old output value. When ck is high, the S and R inputs have immediate effect on the Q and $\overline{Q}$ outputs. [2, 3]

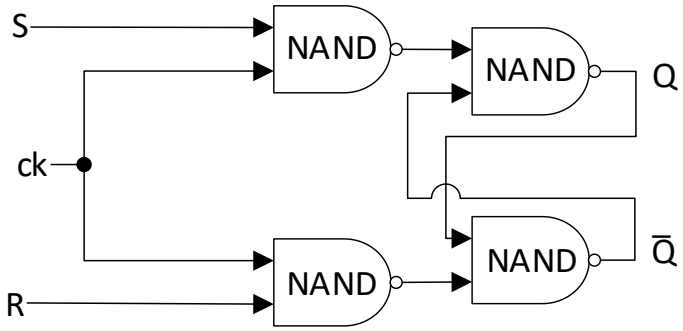

Figure 4. Synchronous SR latch with four NAND gates.

Adding cross coupled feedback to the synchronous SR latch gives use the JK latch of Figure 5. Note that the input signal names have been changed from S to J and R to K. The benefit of the added feedback is that both inputs can be active at the same time because complementary outputs, for Q and $\overline{Q}$, are guaranteed. However, there is still the problem that when both inputs and the clock are active, the outputs will oscillate back and forth from high to low, until the clock signal goes inactive. Similarly to the synchronous SR latch, the JK latch is set when the

J input and clock are high and reset when the K input and clock are high. The JK latch also keeps its previous state if the clock is low, regardless of the values of the J and K inputs. [2, 3]

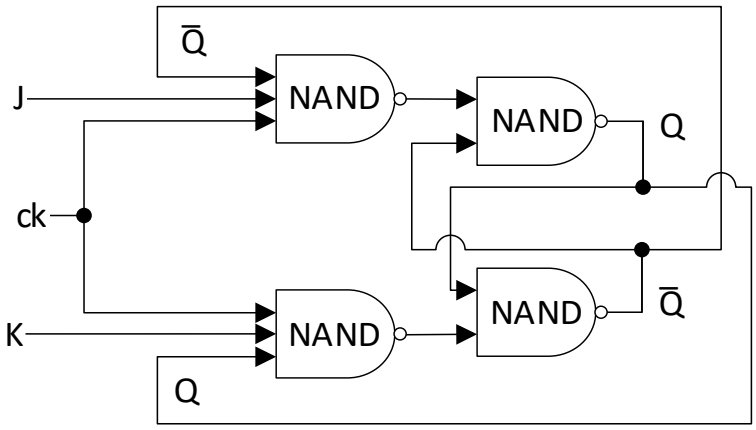

Figure 5. JK latch from the synchronous SR latch with feedback.

The clocked SR latch can be turned into a D latch by connecting an inverter from the S input to the R input. The S input is now renamed the D input and the R input is removed. The result is the D latch presented in Figure 6. The strange behaviour of setting and resetting the SR latch at the same time can now be avoided. It also has the advantage of only having one data input compared to the SR and JK latches. The clock is used to control when the latch is transparent. When ck is high, the latch is transparent, and the value of the D input is continuously loaded to the output of the latch. When ck is low, the output is latched to the last input, and the value of D input does not affect the output. [1, 2]

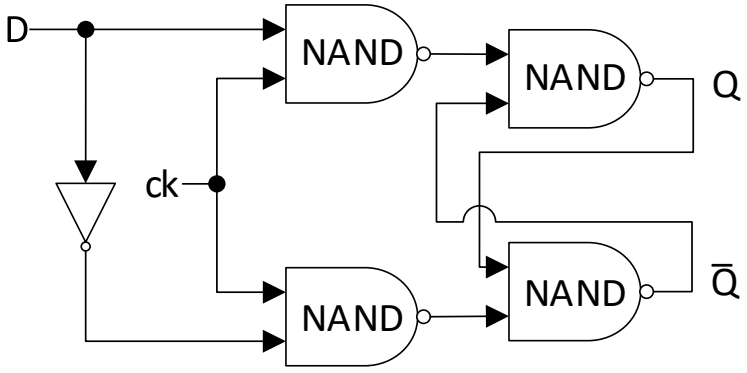

Figure 6. D latch with four NAND gates and an inverter.

The transparency of the latches makes them problematic to use in sequential circuits. Latches cannot be connected in series with the same polarity clock. If all the series connected latches are transparent when clock is high, a change in the input of the first latch can propagate through all the latches. Latches are prone to race conditions. If the input of the latch changes due to feedback paths, while the latch is still transparent, there will be an additional change to the output. Latches are also prone to glitches. Glitches at the input of the latch can propagate to the output when the latch is transparent. The solution is to modify the latches to be edge-triggered instead of level-triggered, giving us flip-flops. [2, 3]

2.2 Flip-flops

A simple way of making latches edge triggered is to make the clock pulse very short, giving us the pulsed latch or the pulsed flip-flop. An edge detector, often also called a pulser, can be used to detect the rising edge of a normal 50% duty cycle clock signal and convert it to a narrow output pulse. [3]

A simple edge detector can be made by connecting the clock signal to the input of an AND gate and connecting an inverted and delayed clock signal to the other input, like in Figure 7. The clock signal is delayed by the gate delay of the inverters and the number of inverters can be used to control the width of the output pulse. An odd number of inverters is needed to invert the clock signal. [3]

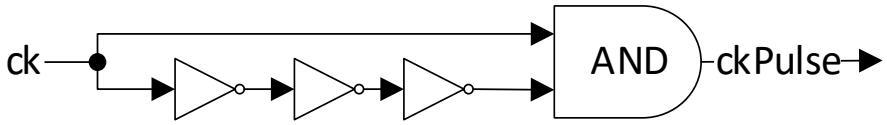

Figure 7. Edge detector using an AND gate and three inverters.

If the edge detector is integrated inside each sequential element, the element is often called a pulsed flip-flop, shown in Figure 8. Several latches can be made to share a single edge detector, often called pulsed latches, to save area and power consumption.

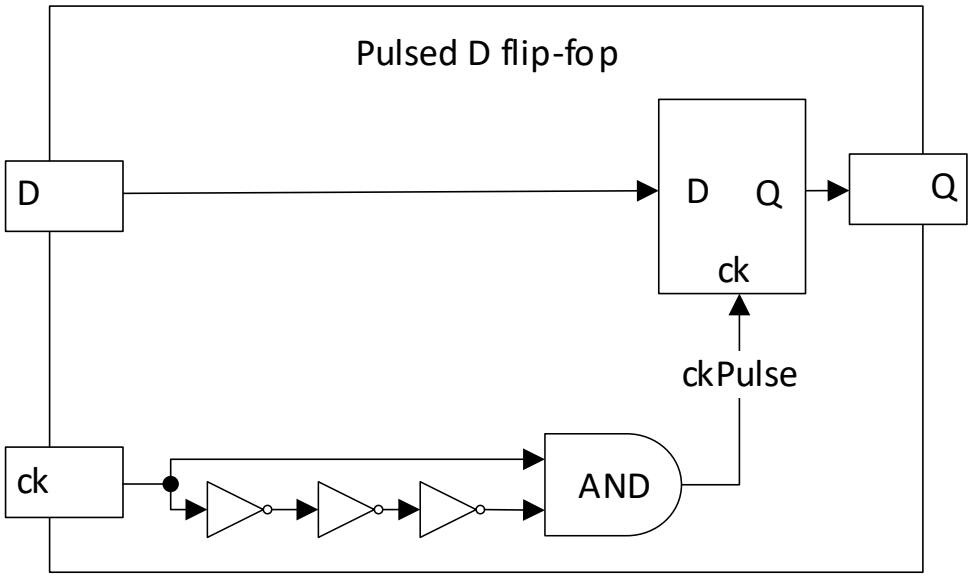

Figure 8. Pulsed D flip-flop from an edge detector and a D latch.

The problem with using edge detection circuitry is guaranteeing the width of the output pulse. The gate delays of logic gates vary with different process and environmental corners such

as temperature and supply voltage. Using pulsed latches can be impractical for designs with fast clocks. [2, 3]

A more widely used method of achieving edge triggered operation is to use a primary-secondary structure, giving us the flip-flop. In it, a primary and a secondary latch are connected in series and using complementary clocks. The first latch is called the primary and the second latch the secondary. Since the latches use complementary clocks, both are never transparent at the same time. The flip-flop will be positive edge triggered if the inverted clock is provided to the primary latch. For negative edge triggering, the inverted clock is provided to the secondary latch instead. [1, 2]

A primary-secondary D flip-flop can be constructed from two D latches, like in Figure 9. It works by loading the value of its data input to its output on the positive edge of the clock signal. When the clock signal is low, the primary latch is transparent, and the value of the data input signal is loaded to the output of the primary latch. On the other hand, the secondary latch is disabled, and it keeps its old state. Changes in the data input can only affect the output of the primary, but not the secondary. When the clock signal rises, the primary latch closes, and it retains the state it had just before the clock edge. At the same time, the secondary latch becomes transparent, and the output of the primary is loaded into the secondary. Since the primary latch is disabled, a change in the data input will not affect the output of the flip-flop. The output of the flip-flop can change only at the positive edge of the clock. A waveform to illustrate the positive edge active operation of the primary-secondary D flip-flop, of Figure 9, is shown in Figure 10. [1]

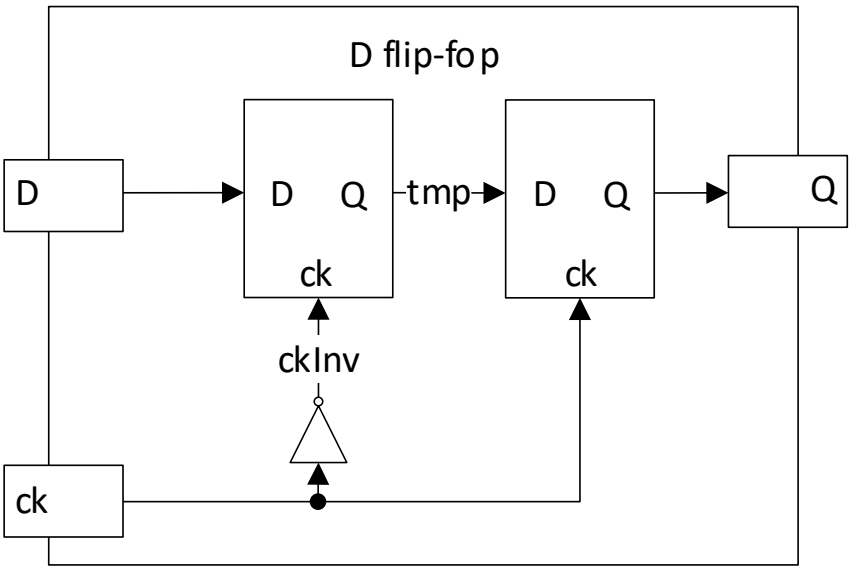

Figure 9. Primary-secondary D flip-flop from two D latches and an inverter.

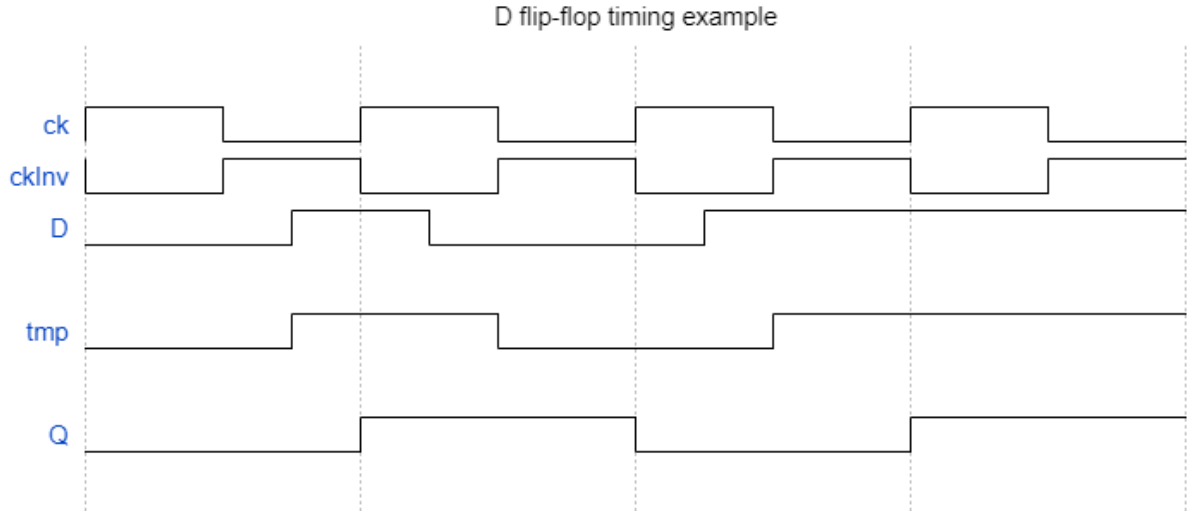

Figure 10. An example waveform of a positive edge active primary-secondary D flip-flop.

The D flip-flop is the most widely used type of sequential element [2]. It has the benefit of having only one data input compared to the SR and JK flip-flops. It can also be constructed out of a smaller number of logic gates than the JK flip-flop [4]. It also does not suffer from the transparency of the latches. However, as the D latch is roughly half the size of the primary secondary D flip-flop, it is an enticing option even with its timing problems. The pulsed flip-flop also brings a slight size decrease, being roughly ¾ the size of a primary secondary flip-flop.

2.3 Resettable sequential logic

A reset is used to force the digital circuit into a known state, especially after the voltage supplies are provided. When a flip-flop or a latch is reset, it ignores the data input and resets the output low. If the output of the sequential element is instead reset to high, it is then often called a set. The reset can be either active high or active low. An active high reset causes a reset when it is high and an active low when it is low. The reset can be either synchronous or asynchronous. The reset is asynchronous if it resets the flip-flops and latches immediately after the reset is asserted, regardless of the clock signal. The reset is synchronous if it resets the flip-flops and latches only after the clock is active. [1]

A synchronous reset can easily be implemented with combinational logic to the data input of a latch or a flip-flop. For example, an active low synchronous reset can be implemented by connecting the data signal and the reset signal through an AND gate to the D input of a D flip-flop, like in Figure 11. When the reset signal is low, the output of the AND gate is low, regardless of the data signal. When the reset signal is high, the output of the AND gate is determined by the data signal. [1]

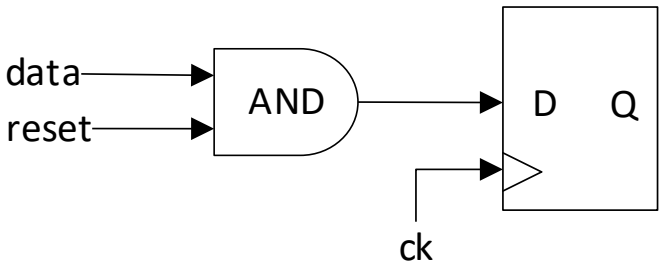

Figure 11. Synchronously resettable D flip-flop.

To implement an asynchronous reset, the internal structure of the of the flip-flop or latch needs to be modified [1]. For example, an active low asynchronous set can be implemented by adding a S input to the NAND gate producing the Q output of a D latch. An active low asynchronous reset can be implemented by adding a R input to the NAND gate producing the $\overline{Q}$ output. Figure 12 shows an asynchronously settable and resettable D latch. When the reset signal is low, it forces the $\overline{Q}$ output high and Q output low. When the set signal is low, it forces the Q output high and the $\overline{Q}$ output low. [3]

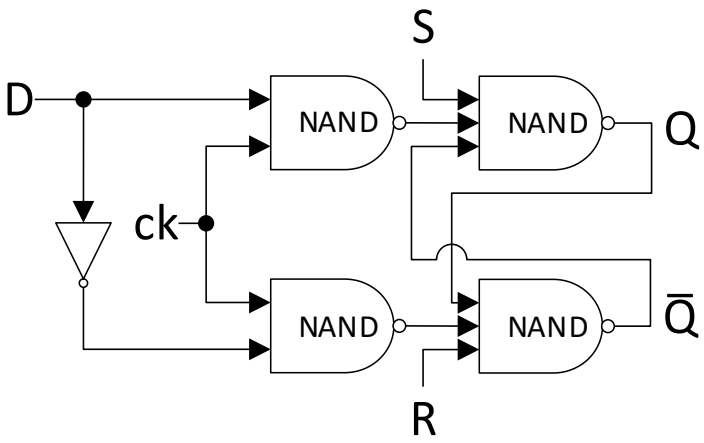

Figure 12. Asynchronously resettable and settable D latch.

2.4 Registers

Flip-flops, using the same clock, can be grouped together to form registers. A group of n flip-flops can store n-bits of binary data. In addition to flip-flops, a register can also have combinational logic to control how data is transferred to the flip-flops. [4]

If all the flip-flops of a register are loaded simultaneously, it is called a parallel loaded register. This type of register is an important building block in digital circuits. A simple register with no combinational logic, like in Figure 13, transfers the data from its inputs to the outputs on each active clock edge. Typically, a control signal and additional logic is needed to control whether a new value is loaded to the register or whether the old value is kept. [4]

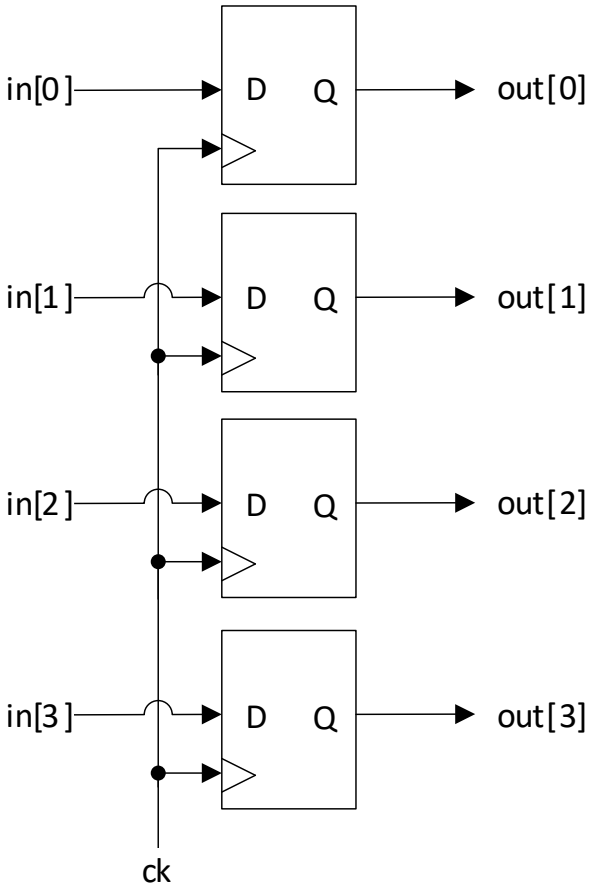

Figure 13. A four bit register with D flip-flops.

Keeping the old value can be achieved by stopping the clock from running. When the control signal is inactive, a clock gate is used to disable the clock to the register. The flip-flops inside the register are prevented from updating their value due to the lack of the clock signal. This method has the disadvantage of adding logic on the clock path. [4]

Another way to retain the state of the register is to control the data input of the register. The register output can be fed back to the input for the register to retain its value. Multiplexer combinational logic and the control signal can be used to select whether the register output or the regular load signal is fed to the input of the register. [4]

A clock gate is larger than a multiplexer, but as the same clock gate can be used to stop the clock to all the flip-flops in a single register, whereas every flip-flop in a register needs their own multiplexer to feed back the old value, from the area point of view, clock gating is beneficial for multi-bit registers.

A shift register differs from a typical register by the fact that the flip-flops are connected in series. The input of the first flip-flop is connected to a serial input and the output of the last flip-flip is connected to serial output. Data is loaded to the shift register by inputting it to the serial input. On each clock cycle the data is shifted forward from all flip-flops in the shift register to the next one in the shift register. A simple shift register consists only of flip-flops, but combinational logic can be inserted to control when the data is shifted forwards, similar to the case with the parallel loaded register. [4]

Registers are typically made from D flip-flops [1]. The parallel loaded register could be constructed using D latches to save area. Due to the latch being transparent on the whole active clock level, clock gating could be used as the loading control to reduce glitches at the same

time. The shift register is problematic to construct using D latches. Due to the latch being transparent on the whole active clock level, data input to the first latch would propagate through as many latches as the gate delays would permit [2]. A solution would be to have every second latch be positive level active and every other latch negative level active.

2.5 Timing analysis of sequential logic

The use of sequential logic allows for orderly operation of a circuit. Sequential storage elements store the state of the circuit. Combinational logic calculates the next state of the circuit from the current state and the external inputs. The next state of the circuit is loaded to the sequential elements on the next active clock edge or level. For the operation of the circuit to be clearly defined, the memory elements cannot mix data from the next or previous cycles with the data from the current cycle [5]. Minimum and maximum propagation delays can be defined between sequential elements.

2.5.1 Timing of flip-flops

On the active edge of a clock, the data input of a flip-flop is transferred to the output. The output starts to change after a clock-to-Q contamination delay, t_{cqcd}, at the earliest and the change has finished after a clock-to-Q propagation delay, t_{cqpd}, at the latest. [1, 5]

The data propagates through the combinational logic to the next flip-flop after a contamination delay, t_{cd}, at the earliest and after a propagation delay, t_{pd}, at the latest. The flip-flop samples the correct value, if the data inputs settle a setup time, t_{setup}, before the active clock edge and stay stable a hold time, t_{hold}, after. Possible skew, t_{skew}, between the clocks to the launching and receiving flip-flops needs to be also taken to account in the timing analysis. The time window, that the data inputs need to be stable in, is show in Figure 14. [1, 5]

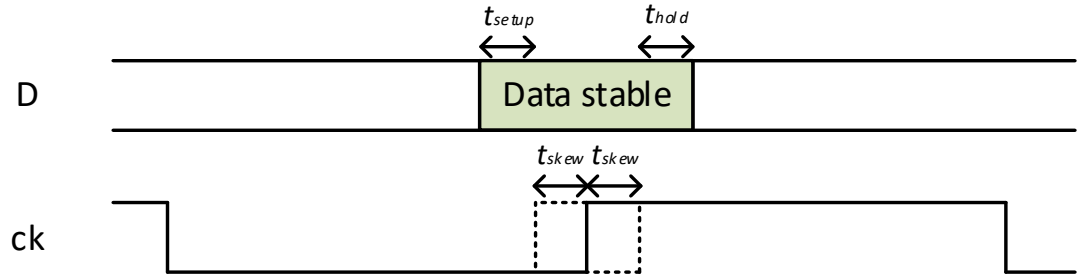

Figure 14. The setup and hold requirements of a D flip-flop.

Since the clock period is often a fixed requirement and the clock-to-Q propagation delay, setup time and hold times are specified by the technology used, the only variable usually controllable by the designer is the propagation delay of the combinational logic paths. The maximum propagation delay allowed between flip-flops for the data to arrive setup time before the next active clock edge is

$$t_{pd} \leq T_c - t_{cqpd} - t_{skew} - t_{setup}, \quad (1)$$

where t_{pd} is the propagation delay between two flip-flops, T_c is the clock period, t_{cqpd} is the clock-to-Q propagation delay of a flip-flop, t_{skew} is the clock skew between the flip-flops and t_{setup} is the setup time requirement. [1, 5]

The minimum contamination delay allowed between flip-flops for the data to arrive hold time after the current active clock edge is

$$t_{cd} \geq t_{hold} + t_{skew} - t_{cqcd}, \tag{2}$$

where t_{cd} is contamination delay between two flip-flops, t_{hold} is the hold time requirement, t_{skew} is the clock skew between the flip-flops and t_{cqcd} is the clock-to-Q contamination delay of a flip-flop. [1, 5]

2.5.2 Timing of latches

Analysing the timing of latches is more complicated, as the latch is transparent on the active clock level and not just the edge. All consecutive latches need to use non-overlapping clocks so that they are not transparent at the same time. The simplest way is to use complemented clocks. This means one latch is transparent for the first half clock cycle and the next latch is transparent for the other half cycle. Another way is to use clocks with less than 50% duty cycle, with enough phase shift between them to not overlap. [5]

Latches load the new data at any point in the active clock level, depending on when it arrives. If the data arrives before the active clock edge, the data is loaded from the input to the output on the active edge of the clock. A change in output is seen after a clock-to-Q contamination delay, t_{cqcd}, at the earliest and after a clock-to-Q propagation delay, t_{cqpd}, at the latest. If the data arrives when the clock level is already active, the data is loaded from the input to the output after a D-to-Q contamination delay, t_{dqcd}, at the earliest and after a D-to-Q propagation delay, t_{dqpd}, at the latest. [5]

Similarly, to the flip-flops, the data propagates to the input of the next latch after a contamination delay, t_{cd}, at the earliest and after a propagation delay, t_{pd}, at the latest. The latch samples the correct value, if the data input has settled setup time, t_{setup}, before the inactive clock edge and stays stable a hold time, t_{hold}, after. [5]

Due to the level transparent nature of latches, they can be used in a way that the data arrives more than the setup time and clock skew before the inactive clock edge. In that case the setup time and clock skew can be left out of the maximum allowed propagation delay analysis. The maximum allowed propagation delay between two latches using different clock edges is

$$t_{pd} \leq \frac{T_c}{2} - t_{dqpd}, \tag{3}$$

where t_{pd} is the propagation delay between two latches, T_c is the clock period and t_{dqpd} is the D-to-Q propagation delay of a latch. The clock period is divided by two, as two back-to-back latches use complemented clocks. [5]

The minimum contamination delay allowed between two latches for the data to arrive hold time after the inactive clock edge to the first latch is

$$t_{cd} \geq t_{hold} + t_{skew} - t_{cqcd}, \tag{4}$$

where t_{cd} is contamination delay between two latches, t_{hold} is the hold time requirement, t_{skew} is the clock skew between the latches and t_{cqcd} is the clock-to-Q contamination delay of a latch. The clock-to-Q contamination delay is used for minimum contamination delay analysis instead of the D-to-Q contamination delay, as the output of the first latch can change from the active edge of the clock at the earliest. [5]

2.5.3 Timing of pulsed latches

Pulsed latches behave the same as the normal latch, with the difference being that they use short pulses as clock instead of the normal 50% duty cycle clock. The pulsed latches can use the same clock signal if the clock pulse is shorter than the propagation delay between the pulsed latches and the hold time requirement is not violated. If the pulses are short enough, the timing of pulsed latch starts to resemble that of the flip-flop. [5]

The data input of the pulsed latch must settle setup time and clock skew before the inactive edge of the clock pulse. The setup time may be before the active clock edge or after, depending on the width of the clock pulse and the clock skew. If the clock pulse is wider than the setup time and clock skew combined, the data needs to only arrive when the latch is already transparent, and the timing analysis for the propagation delay is the same as for the normal latch. If the clock pulse is shorter, the data needs to arrive while the latch is not yet transparent. In that case the setup time and clock skew need to be considered in the maximum allowed propagation delay, similarly to the flip-flop case. The maximum allowed propagation delay between two pulsed latches is

$$t_{pd} \leq T_c - t_{dqpd} - \max(0, t_{setup} + t_{skew} - t_{pw}), \tag{5}$$

where t_{pd} is the propagation delay between two latches, T_c is the clock period, t_{dqpd}, is the D-to-Q propagation delay of a latch, t_{setup} is the setup time, t_{skew} is the clock skew and t_{pw} is the pulse width of the clock. [5]

The data output of the first latch can change at the active clock edge at the earliest. The data input of the second latch cannot change until a hold time and clock skew after the inactive edge of the pulse. The minimum contamination delay allowed between two latches for the data to arrive hold time after the inactive clock edge to the first latch is

$$t_{cd} \geq t_{pw} + t_{hold} + t_{skew} - t_{cqcd}, \tag{6}$$

where t_{cd} is contamination delay between two latches, t_{pw} is the width of the clock pulse, t_{hold} is the hold time requirement, t_{skew} is the clock skew between the latches and t_{cqcd} is the clock-to-Q contamination delay of a latch. [5]

As can be seen from comparing the contamination delay equations of flip-flops (4) and pulsed latches (6), the contamination delay of the pulsed latch depends on the pulse width and needs to be larger. This can result, in pulsed latch designs, to more hold time violations, that need to be fixed.

2.5.4 Time borrowing

Due to the level transparent nature of a latch, data can arrive to the input of the latch at any time on the active clock level and it will propagate to the output. The propagation delay from one latch to the input of the next does not need to be exactly half a clock cycle. Some logic blocks can have larger delays while some have shorter delays. Slow logic using time allocated to faster logic is called time borrowing. As flip-flops are edge triggered, time borrowing is not possible with them. [5]

The direction of time borrowing depends on when the latches update on the clock cycle. If the latches update in the beginning of the active clock level, the next latch can borrow time and update later in the active clock level. If the latches update at the end of the clock level, the next latch can update earlier and not have to wait for the end of the active clock level, giving more time to the next latches. Time borrowing can happen over multiple latches, as long as the data does not arrive so late as to cause a setup time violation in any of the latches. [5]

As the latches update as soon as the data arrives on the active level, they naturally move to updating at the active clock edge. In that case, the amount of time allowed for borrowing is at its highest. If the data departs the first latch on its active clock edge, it normally arrives at the next latch half a clock cycle later. The circuit will still operate correctly if it arrives setup time and clock skew before the inactive clock edge of the second latch instead. The maximum time that can be borrowed therefore is

$$t_{borrow} \leq \frac{T_c}{2} - t_{setup} - t_{skew}, \tag{7}$$

where t_{borrow} is the time borrowed, T_c is the clock period, t_{setup} is the required setup time and t_{skew} is the clock skew between the latches. [5]

If the latches are transparent shorter amount of time in the cycle, the amount of time that can be borrowed reduces. The maximum time that can be borrowed with pulsed latches is

$$t_{borrow} \leq t_{pw} - t_{setup} - t_{skew}, \tag{8}$$

where t_{borrow} is the time borrowed, t_{pw} is the width of the clock pulse, t_{setup} is the required setup time and t_{skew} is the clock skew between the latches. If the pulse width is shorter than setup time and clock skew, time borrowing is not possible. [5]

2.5.5 Fixing timing violations

Violating either the setup or hold time constraints means that memory element can sample the input while it is still changing leading to an ambiguous output. [5]

To fix setup time violations, the designer can redesign the combinational logic paths between the sequential elements to be shorter or add sequential elements in the middle of the combinational logic. As we can see from equations (1), (3) and (5), the clock period can also be made larger to increase the allowed propagation delay.

To fix hold time violations, the designer can add buffers between the sequential elements, that increase the contamination delay. As we can see from equations (2), (4) and (6), the clock period does not affect the contamination delay and so it cannot be used to fix hold time violations. [5]

2.6 Power consumption of CMOS logic

The power consumption in digital complementary metal-oxide-semiconductor (CMOS) logic consists of static and dynamic power consumption. Dynamic power consumption results from the switching activity of the circuit and consists of switching power consumption and short-circuit power consumption. When the value of a logic gate changes, its load capacitance needs to be charged or discharged resulting in the switching power consumption of

$$P_{sw} = C_L * V_{dd}^2 * f * a, \tag{9}$$

where C_L is the load capacitance, V_{dd} is the supply voltage, f is the clock frequency and a is the switching activity factor. Short-circuit current results from a momentary short-circuit through the transistors from the supply voltage to ground when the logic gate is switching its state. Short-circuit power consumption can be expressed as

$$P_{sc} = I_{sc} * V_{dd}, \tag{10}$$

where V_{dd} is the supply voltage and I_{sc} is the average short-circuit current. [6]

The static power consumption represents the power consumption when the circuit is inactive. The transistors in the circuit conduct small amounts of leakage current event when in the cut-off state. The static current consumption can be expressed as

$$P_{st} = I_{lk} * V_{dd}, \tag{11}$$

where V_{dd} is the supply voltage and I_{lk} is the leakage current. [6]

3 DESIGN FOR TESTABILITY

Modern integrated circuits (ICs) are made with nanometre-scale lithographic techniques. Some of these manufactured ICs are faulty due to manufacturing defects. During the manufacturing process, all the ICs need to be tested to remove the faulty ones. Inadequate testing leads to faulty ICs being shipped to the customers.

Manufacturing testing of digital logic is typically done with the help of automatic test equipment (ATE) and automatic test pattern generation (ATPG) tools. ATPG tools use algorithms to generate test patterns that detect faults in a faulty circuit. ATE is computer-controlled equipment that applies test patterns to the circuit under test (CUT) and compares the circuit responses to the stored responses of a fault-free circuit. Fault simulations are used to estimate the quality of the generated test patterns. [7]

The test patterns are usually generated based on specific fault models, that represent specific faults resulting from manufacturing defects. The stuck-at fault (SAF) model is a widely used fault model. It assumes that a faulty signal line is stuck to either a logic 1 or a logic 0. The stuck-at fault can be detected by driving the specific signal line to the opposite state than the assumed fault and observing whether the line was stuck or not. Test pattern that detects both stuck at 1 and stuck at 0 faults on every signal line need to be generated to reach 100% stuck-at fault coverage. Test patterns generated based on the stuck-at fault model also detect most faults based on other models. The stuck-at fault model however does not guarantee detecting all possible defects and a combination of tests, made with different fault models, are typically used in digital testing. [7]

For the circuit to behave correctly, it should not only perform the correct logical operation, but also perform it fast enough for the change to propagate to the next flip-flop. At-speed testing refers to tests that detect delay defects from a circuit running at functional speed. The transition fault (TF) model assumes a single faulty gate is either slow to rise or slow to fall, so that the transition does not reach the flip-flop or primary output within the clock period. Transition faults are detected by first setting the target node to an initial value, then causing a transition to the target node, and finally observing the fault effect from a scannable flip-flop or a primary output. [7]

Some defects may not affect the correct operation of the circuit but can result in increased power consumption. Leakage current (IDDQ) test measures the CUT supply current in steady state condition to detect defects. The IDDQ test can detect shorts between signal lines or between signal lines and supply lines, from the increased leakage current of the shorts. IDDQ testing requires that the internal nodes can be controlled, to toggle the internal nodes to different states, but does not require observation, since the pass and fail condition is determined only on the supply current consumption. [7]

Integrated circuits can consist of deep sequential logic. Internal signal lines in this kind of circuit can be extremely hard to control and observe from the primary inputs and outputs of the circuit, which can result in very complex test patterns and low fault coverage. [7]

Design for testability (DFT) methods have been developed to improve the testing of digital integrated circuits. They refer to methods, where the testability of the digital circuit is considered already in the designing of the circuit. Logic structures, meant to facilitate testing, are added to the design to reduce the time and therefore the cost of testing, and to improve testing fault coverage. [7]

3.1 Scan design

Scan design is the most widely used DFT approach for improving testability. In it, the controllability and observability of sequential elements is improved by providing external access to them. This is done by converting the sequential elements to scannable sequential elements, called scan cells, and connecting them to form shift registers, called scan chains. [7]

The use of scan chains allows test stimulus to be easily shifted to the storage elements inside the IC, without having to control the primary inputs of the circuit for several clock cycles in order to bring the internal storage elements to a desired state. The use of scan chains also allows the test response to be easily shifted out of the internal storage elements. Since the inputs to combinational logic blocks can be controlled directly and the outputs of combinational logic blocks can be observed directly, the test pattern complexity required to test specific faults is greatly reduced. [7]

The scan cells have two inputs. The data input is connected to the normal combinational logic path. The scan input is connected to the output of another scan cell. The scan chain is made accessible by connecting the scan input of the first scan cell, in the chain, to a primary input of the IC and connecting the output of the last scan cell, in the chain, to a primary output of the IC. [7]

Shifting the test stimulus in and the test response out of a scan chain made of n storage elements takes n clock cycles. Having several scan chains reduces the clock cycles needed for shifting, with the disadvantage of one primary input and output needed for each scan chain. [7]

A scan design has several modes of operation. In normal mode, all the added scan functionality is disabled, and the design operates like before adding the additional logic. In scan mode, all scan related fixes, to improve test coverage or to guarantee safe operation during scan testing, are turned on. In shift mode, all scan and shift related fixes are turned on, and scan chains are enabled to allow shifting in test patterns or shifting out test results. Table 1 below shows a set of scan design rules and their suggested fixes. [7]

Table 1. Scan design rules

Rule	**Recommended fix**
Avoid tristate buses during shift	Two or more bus drivers trying to drive a tristate bus to opposite logic values causes bus contention. It should be guaranteed that only one driver controls the bus at a time during shift. This can be done by forcing one bus driver enable signal high and all others low during shift.
Avoid bidirectional input or output (I/O) ports during shift	Conflicts can occur at a bidirectional port during shift operation if it is used as an input port and the output tristate buffer is controlled by logic connected to a scan cell. The output tristate buffer should be disabled during shift. This can be done by forcing the enable signal controlling the output tristate buffer low during shift.
Avoid gated clocks during shift	Clock gating can prevent shift operation from happening correctly by blocking the scan clock from reaching the scan cells. Clock gating should be disabled during shift. This can be done by forcing the enable signal of all clock gates high during shift.
Avoid derived clocks during scan	A clock generated internally in the digital logic is a derived clock. They should be bypassed during the scan test, in order to test the logic driven by the derived clocks. This can be done by using a multiplexer to provide a scan clock instead of the derived clock during scan.

Avoid combinational feedback loops during scan	The use of combinational feedback loops can lead to loss of fault coverage, since they cannot be controlled, or their value determined. Combinational feedback loops should be removed from the design. If removing is not possible, they should be broken during scan. This can be done by inserting a scan cell in the loop that is used during scan.
Avoid asynchronous set and reset signals during scan	Internally generated asynchronous set and reset signals to scan cells, that are not directly controlled from a primary input, can prevent shift operation from happening correctly. Asynchronous set and reset signals to scan cells should be forced to inactive state during scan.
Avoid clocks driving data during scan	Clock used as a data input can cause race condition when the test response is captured. Clock path to data input of flip-flop should be blocked during scan.
Avoid floating buses during scan	Bus-holders should be added for the tri-state buses to keep their last value.
Avoid floating inputs during scan	Floating inputs should be tied to logic 0 or logic 1 during scan.
Avoid non-scan storage elements in full-scan designs	Should be made transparent, bypassed, or initialized into a known state during scan.

The added scan operation of the circuit is controlled with the use of added test signals or test clocks. A primary input pin is often added to the design for the signal that is used to enter scan mode. Extra pins for the rest of the test signals, the scan inputs and the scan outputs can be saved by connecting them to existing primary I/O pins with multiplexers and using the scan mode signal as the enable for the multiplexers. The different test signals and test clocks used in different types of scan designs are discussed in the following sub-chapters.

Typically, all the sequential elements in a design are connected to scan chains. The advantage of full-scan is that all the inputs to combinational logic can be controlled and all the outputs to combinational logic can be observed. It makes it easier for ATPG programs to create test patterns to detect faults. A disadvantage of full scan is that the added propagation delays of scan logic on critical paths might lead to timing violations. Another disadvantage is the increased area overhead caused by the scan logic. [7]

Partial scan refers to designs where some of the sequential elements are not scannable. Sequential elements in critical paths can be left as normal sequential elements to meet timing requirements. Another advantage of partial scan is that it reduces the area overhead of using scan cells. Since not all sequential elements are controllable and observable in partial scan, the test generation complexity is increased. This can result in reduced fault coverage and increased testing time compared to full-scan. [7]

3.1.1 Muxed-D scan style

The most widely used scan style is the muxed-D style. It was developed at Stanford by M. J. Y. Williams and J. B. Angell in 1973 [8]. In it, the D flip-flops are replaced with edge triggered muxed-D scan cells, shown in Figure 15. The muxed-D scan cell is made from a D flip-flop and a multiplexer. The multiplexer is used to select between the data input, DI, and scan input, SI. [7]

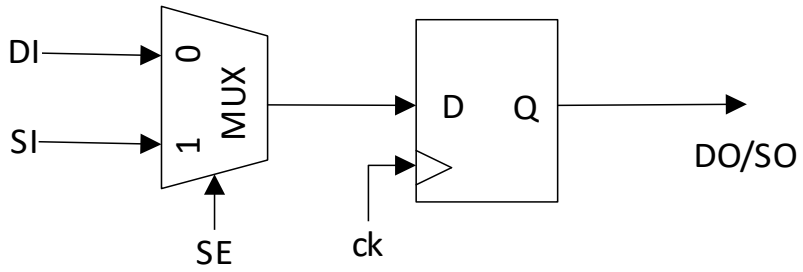

Figure 15. Muxed-D scan cell made from a D flip-flop and a multiplexer.

In normal and scan modes, the value at the data input is selected and loaded to the output of the flip-flop on the active edge of the clock. In shift mode, a shift enable, SE, signal is used to select the scan input. The value from the scan input is loaded to the output of the flip-flop on the active edge of the clock. [7]

The scan testing of a muxed-D scan design consists of the following steps, also shown in Figure 16 [7].

- All scan related fixes are turned on by setting the scan mode, SM, signal high.
- The scan chain portion of the first test pattern is loaded from the primary scan input, PSI, to the chain by setting the shift enable, SE, signal high and clocking the circuit n times, where n is the number of scan cells in the scan chain.
- The test pattern is applied to the primary inputs.
- The test response is observed from the primary outputs.
- Shift enable, SE, signal is set low, and the CUT is clocked once to capture the test response of the combinational logic to the scan cells.
- The test response from the scan cells is shifted out from the primary scan output, PSO, and at the same time the next test pattern is shifted in from the primary scan input, PSI, by setting the scan enable, SE, high and clocking the circuit n times, where n is the number of scan cells in the scan chain.
- The capture and shift operations are repeated until all test patterns are tested.

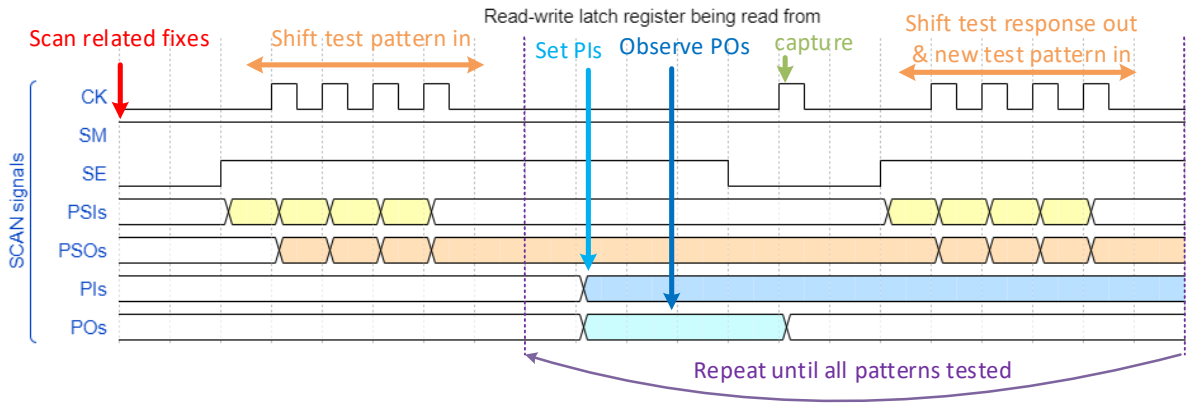

Figure 16. Muxed-D scan testing example waveform.

A disadvantage of using muxed-D scan cells is the added multiplexer delay to the data path and the area increase. A benefit is the compatibility of muxed-D scan cells with modern designs

primarily using edge triggered D flip-flops. Another benefit is that the existing design automation tools provide good support for muxed-D scan style. [7]

3.1.2 Clocked-scan style

In the clocked-scan style, the D flip-flops are replaced with clocked-scan cells. The clocked scan cell also has a data input and a scan input. In contrast to the muxed-D scan cell, the input selection in the clocked scan cell is done using two independent clocks, the data clock, and the shift clock. [7]

In normal and scan modes, the value from the data input, DI, is loaded to the output on the active edge of the data clock, CKD. In shift mode, the value from the scan input, SI, is loaded on the active edge of the shift clock, CKS, instead. [7]

The scan testing of a clocked-scan design is similar to the muxed-D scan design. The main difference between them is in the shift operation. The clocked-scan design does not have the shift enable signal. The shifting is done through the primary scan input, PSI, and the primary scan output, PSO, by clocking the shift clock, CKS, n times, where n is the number of scan cells in the scan chain. During the shift operation, the data clock, CKD, is held inactive. The shift clock, CKS, is held inactive during the capture operation. The test response is captured to the scan cells by clocking the data clock, CKD, once.

A big advantage of the clocked-scan style is that it does not add any logic delays to the data path. The disadvantage is that it requires routing an extra shift clock to all the clocked-scan cells. [7]

3.1.3 LSSD style

Level-sensitive scan design (LSSD) was developed at the International Business Machines Corporation by E.B. Eichelberger and T.W. Williams in 1977 [9]. It is the primary scan style used for level-sensitive, latch-based designs. In it, the D latches and D flip-flops are replaced with LSSD scan cells, shown in Figure 17. The scan cell contains two latches, a primary D latch, L1, and a secondary D latch, L2, and three clocks, the primary clock, CKA, the scan clock, CKS, and the secondary clock, CKB. The primary latch has a data input, DI, and a scan input, SI. The selection of which input is loaded to the output of the primary latch is made with the primary and scan clocks. The secondary clock is used to load the data at the primary latch output to the secondary latch output. [7]

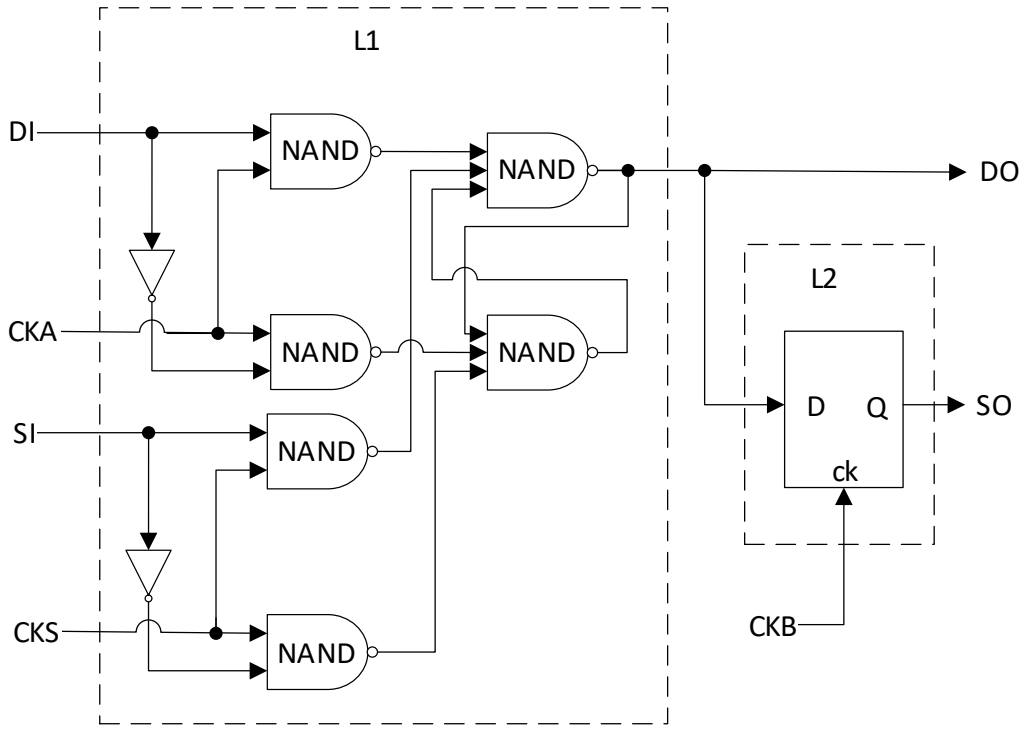

Figure 17. Single-latch LSSD scan cell from a primary two-input D latch and a secondary D latch.

In normal and scan modes, the value at data input is loaded to the primary latch output on the active level of the primary clock. In shift mode, the value at scan input is loaded to the primary latch output at the active level of scan clock. For flip-flops, the data at the primary latch output is loaded to the secondary latch output on the positive edge of the secondary clock in all modes. For latches, the data is loaded with the secondary clock in only shift mode. All the clocks need to be operated in a non-overlapping manner to avoid race conditions. [7]

When replacing D latches with the LSSD scan cell, the data output, DO, is taken from the output of the primary latch and the scan output, SO, is taken from the output of the secondary latch, like in Figure 17. There also needs to be at least two non-overlapping system clocks, CKA1 and CKA2, supplied to the CKA ports of the LSSD scan cells to prevent back-to-back latches from being transparent at the same time. When replacing D flip-flops with the LSSD scan cell, both outputs are taken from the output of the secondary latch. [7]

The scan testing of a LSSD design is similar to the muxed-D scan design. The shifting is done through the primary scan input, PSI, and the primary scan output, PSO, by clocking the shift clock, CKS, and the secondary clock, CKB, n times in a non-overlapping manner, where n is the number of scan cells in the scan chain. During the shift operation, the primary clock, CKA, is held inactive. The shift clock, CKS, is held inactive during the capture operation. The test response is captured to the D flip-flop scan cells by clocking the primary clock, CKA, and secondary clock, CKB, in a non-overlapping manner once. The test response is loaded to the D latch scan cells by clocking the two system clocks, CKA1 and CKA2, in a non-overlapping manner once. [7]

The scan enabled LSSD scan cell uses a scan enable, SE, signal to choose between the data input, DI, and the scan input, SI, of the primary latch instead of a scan clock. The secondary latch has the same functionality as in the traditional LSSD style. [10]

The advantage of using LSSD is that it allows scan chains to be used in latch-based designs. A disadvantage is that two additional clocks need to be routed when replacing latches and one

additional clock needs to be routed when replacing flip-flops, resulting in increased routing complexity, and ultimately increased area. [7]

Replacing the D latches in a design with LSSD scan cells results in an additional secondary D latch being added for every D latch in the design, wasting a lot of area. The L1L2* scan optimization technique saves this area by removing these dummy latches. In it, latches in the design, that are independent of each other, are grouped in pairs of L1 and L2* latches. The logic connected to these latches can be tested independently of each other with two different test modes, L1test and L2test. In the L1test, the L1 latch is the primary latch and L2* latch is used as the secondary latch for the L1 latch. In the L2test the L2* latch is the primary latch and the L1 latch is used as the secondary latch for the L2* latch. A problem with this optimization technique is choosing which latches can be grouped together. Real designs often cannot completely group all latches together and some dummy latches need to be used. Another problem is that the ATPG tool needs a remodelled netlists for both tests, where the parts that are not tested are removed from the files. [11]

3.1.4 Latches in scan designs

While the LSSD scan style presented in chapter 3.1.3 is usually used in designs containing latches, there are other ways of dealing with latches in scan designs. The simplest solution that comes to mind is to not replace the latches with scannable elements. In that case, the latches need to be initialized into known states, bypassed, or made transparent during the scan test [7]. The disadvantage is that we have now a partial scan or a non-scan design with the decreased testability that comes with that.

The principle of initializing the latches to known states is used in [12], where the latch registers are modelled as combinational gates and preloaded with values before each scan test. This method allows the ability of the latches to hold data to be tested also. Preloading the background data requires adding multiplexers to the data input of the latches and OR gating a force write signal to the clock inputs of the latches. A downside of this method is that the background data needs to be supplied to the scan model by manually modifying the netlist provided to the ATPG tool. This manually modified model needs to be then also verified against the functional model with simulations. Having the same background data for all the latch registers during the test can cause some combinational logic faults to be untestable due to the reconvergence of the background data. A way around this problem is to use different background data for different groups of registers. Another way is to replace some key latches by scannable flip-flops. [12]

A single-latch and single-clock solution is proposed in [13], that uses the pulsed latch principle covered in section 2.1.5.3. A test signal is used to select between the data and scan inputs of the latch. The upper width of the test clock pulse is bounded by the shortest propagation delay of the latch design. [13]

A problem with the single-latch and single-clock solution is that it is not supported by the existing design automation tools.

An asynchronous scan-latch controller solution is proposed in [14], that uses the asynchronous symmetric pulse persistent protocol to control the scan chain made from latches. The asynchronous scan-latch controller is used to control the transparency of the latches in shift mode. The active edge in the scan clock causes the asynchronous scan-latch controller to pulse the clock input of each latch in the scan chain in turn. The last latch in the scan chain is updated first, followed by the rest all the way the first latch in the scan chain. The pulses to the clock

inputs of the latches are timed in a manner that no two latches are transparent at the same time. [14]

A disadvantage with the asynchronous scan latch controller solution is the added design complexity of the controller and the additional clock tree needed from the controller to the latches. This solution is also not supported by the existing design automation tools.

3.2 Logic built-in self-test

Logic built-in self-test (BIST) is a DFT method that incorporates testing features on the CUT itself. Logic structures to generate test patterns and to analyse the output responses are embedded on the designed circuit. [7]

Logic BIST techniques can be divided into two main categories [7].

- Online BIST techniques, where the test functionalities are performed while circuit operates in normal mode, with normal functionality.
- Offline BIST techniques, where the test functionalities are performed in a test mode.

In functional offline BIST techniques, the tests are performed based on the functional specification of the circuitry. In structural offline BIST techniques, the tests are preformed based on the structure of the circuitry instead. The test pattern generation and the output response analysis in a structural offline BIST can be done either with the help of the functional storage elements, an internal BIST, or with separate added logic, an external BIST. [7]

BIST schemes most commonly convert the storage elements of a circuit to scan cells for combinational circuit testing. Some schemes, involving sequential testing by applying test patterns to the inputs of the circuit and analysing the responses at the outputs also exist. [7]

A typical logic BIST, using the structural offline BIST scheme, is shown in Figure 18. It consists of a test pattern generator (TPG) that automatically generates test patterns for the CUT once it is activated. An output response analyser (ORA) is used to automatically compact the output response into a signature. A BIST controller activates the test, compares the ORA signatures against golden signatures and reports a pass or a failure once the BIST operation is complete. [7]

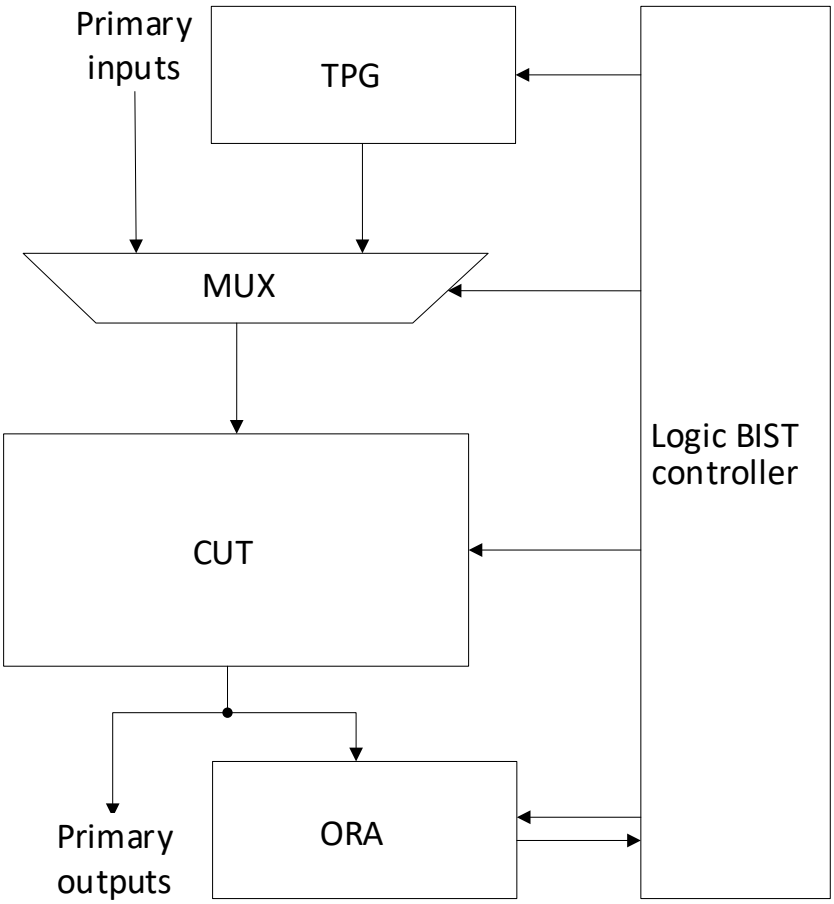

Figure 18. Structure of a typical logic BIST.

The structural offline BIST technique has several advantages compared to the traditional scan techniques. Test costs are reduced as it requires simpler ATE and ATPG, less test time and less tester memory. The tests can be run at functional clock speed, allowing the detection of delay faults. The tests can be run at any time, without the help of an external tester. [7]

There are also several disadvantages compared to traditional scan techniques. There are even more stringent BIST design rules that need to be followed compared to the scan design rules. Since any unknown values that propagate to the ORA in a BIST design will corrupt the signature, following BIST design rules is mandatory for BIST designs. In contrast, breaking many of the scan design rules in a scan design will only result in the loss of fault coverage. Another disadvantage is that the BIST fault coverage is less than that of a scan design, and additional test points might need to be added to increase it to a sufficient level. [7]

3.2.1 Test pattern generation

Several different methods of constructing TPGs exist for generating test patterns for exhaustive, pseudo-exhaustive and pseudo-random testing. [7]

Exhaustive testing achieves maximum fault coverage by applying all possible 2^n input patterns to a circuit consisting of n inputs [7]. Exhaustive testing becomes impractical already for circuits consisting of more than 20 inputs [15].

Pseudo-exhaustive testing aims to reduce test length, while keeping the high fault coverage of exhaustive testing. Pseudo-exhaustive testing aims to exhaustively test a circuit, without applying all possible input patterns by taking advantage of the fact that all the outputs of a circuit do not usually depend on all the inputs. [7]

Pseudo-random testing aims to reduce test length even further by sacrificing fault coverage. In it, a pseudo-random sequence of test patterns is generated with a pseudo-random pattern generator (PRPG). Fault simulation is used to calculate the fault coverage of the pseudo-random test patterns. It is the most used technique for BIST test pattern generation. A downside of pseudo-random testing is that random pattern resistant faults can limit the fault coverage. Test point insertion, mixed-mode BIST, and hybrid BIST techniques can be used to increase the fault coverage of random pattern resistant circuits. [7]

Considering how test patterns are generated in scan designs, generating the test patterns with an ATPG tool comes to mind. Good test patterns can be generated with an ATPG and saved in read only memory (ROM) on the circuit [15]. This method is impractical due to high area overhead of the ROM [15].

Binary counters can be used to create exhaustive test patterns. They are simple to design but require more hardware than the typical linear feedback shift register (LFSR). [15]

Linear feedback shift registers use very little hardware, and they are the preferred TPG method for logic BISTs. A LFSR is a shift register made from flip-flops and feedback from outputs of selected flip-flops to the inputs of selected flip-flops through XOR gates. [15]

3.2.2 Output response analysis

For a logic BIST, storing the golden responses on-chip and doing bit by bit comparison of the output responses is impractical due to the extreme memory area that would be required. Output response analysis techniques can be employed that compact the output response into a signature. The signature is compared to a stored golden signature, that is compacted with the same compression mechanism from a golden output response. Compaction differs from compression by it being lossy instead of loss-less. It is important for the compaction technique to produce different signatures for faulty and fault-free circuits, otherwise the errors are masked. [7]

Signature analysis is the most used output response analysis technique. Serial signature analysis technique is used in circuits having a single output and parallel signature analysis technique in circuits having multiple outputs. Ones count and transition count methods, for example presented in [7], are simpler but suffer from more fault masking. [7]

A single-input signature register (SISR) uses an additional XOR gate at the input of an LFSR for compacting the output response. The LFSR needs to be initialized to a seed value, usually all zeros, before loading the output sequence. [7]

A problem with the SISR is that one analyser is needed for each output of the CUT. Hardware can be saved by choosing each output at a time with a multiplexer and using one common analyser, but the test time is increased by the number of outputs in the CUT. Superposition can be used to compact several output sequences to the same LFSR. A multiple-input signature register (MISR) uses n extra XOR gates to connect n L-bit output sequences to the different stages of a single LFSR. [7][15]

3.2.3 Logic BIST architectures

Many architectures exist for incorporating BIST techniques into a design. The simplest architectures do not use any special structures in the CUT and are used in the board or system level. Some architectures, test-per-scan BISTs, utilize scan chains in the BIST circuitry and provide better access to the internal circuitry of the CUT. Other architectures, test-per-clock BISTs, go even further and use redesigned internal storage elements for test pattern generation and output response analysis to reduce the testing time. Some architectures have been developed that remove the need for an ORA by using concurrent checking circuitry. [7]

Architectures utilizing scan chains provide better fault coverage for sequential circuits and are therefore the focus of this chapter. A curious reader can find many other architectures proposed in literature, for example in [7], that are not covered here.

3.2.3.1 LSSD on-ship self-test

The LSSD on-chip self-test (LOCST) architecture incorporates an external scan chain, comprising of all primary inputs and outputs, to the design. The external scan chain is connected to the internal scan chain, consisting of LSSD scan cells. A PRPG is used to generate test patterns that are shifted into the combined scan chain. The system is clocked to capture the output response to the scan cells. The output response is compacted to a signature by shifting it to a SISR, with the combined scan chain. The signature is compared to a golden signature and a pass, or a failure signal is generated. [7]

3.2.3.2 Self-testing using MISR and parallel SRSG

The Self-Testing Using MISR and parallel shift register sequence generator (STUMPS) architecture is widely used in industry, due to it being easy to integrate with traditional scan. A PRPG is used to generate test patterns, that are loaded into several scan chains in parallel. The system is clocked to capture the output responses to the scan cells. The output responses are compacted by shifting them to a MISR, with the scan chains. New test patterns are generated and shifted in at the same time, while the output responses are shifted out. [7]

3.2.4 Latches in logic BIST designs

Non-scannable latches in a logic BIST design can cause unknown values to emanate to the ORA, corrupting the BIST operation. They should be dealt with by bypassing them or initializing them into known states.

A more practical way to deal with latches is to make them scannable, increasing the fault coverage that can be achieved. Making the latches scannable means that they can be used in logic BIST architectures utilizing scan chains. The LSSD scan cell is typically used as the replacement cell for the latches. Like in the scan design case, the downside is the added area of the LSSD scan cells.

A way of integrating a latch-based register bank to a logic BIST, using the STUMPS architecture, is presented in [16], that uses an optimized LSSD scan style. Each latch, in the register bank stores one bit of data in functional mode. For the scan shifting the latches are

grouped into pairs of L1 and L2* latches that form LSSD scan cells. When the test patterns are shifted through the scan chain both the L1 and L2* latches of each LSSD scan cell contains the same data. When the data is dispatched downstream from both latches in the response capture window, the correlated data can lead to some faults being masked. Another consideration is the data capture into the L1 and L2* latches. When the test response is shifted out with the scan chains, by first clocking the L1 latch, the test data is destroyed in the L1 latches. A logic BIST clock control scheme, that supresses the L1 clock during the first shift operation roughly half of the time, can be used to improve the testability of the logic feeding the L1 latches. [16]

3.3 Memory built-in self-test

Digital circuits often require storing large amounts of data. This is often achieved with the use of embedded memories. Memories can be constructed out of flip-flops, but they are not ideal for large memories, due to their high area. Other memory types exist, that trade performance for smaller area. The simplest static random-access memory (SRAM) cell is made from just two cross-coupled inverters described in section 2.1.1 [1]. Dynamic random-access memory (DRAM) cell stores a bit of data in the presence or absence of charge in a capacitor and is even smaller [1].

Memories made from flip-flops or latches can be tested using scan or logic BIST methods by replacing the flip-flops or latches with scan cells. If the area overhead is unacceptable, or the other, non-scannable, memory types are used, other testability methods are required.

Memory BIST is to most used DFT method for testing large, embedded memories. It is used to detect manufacturing defects in the address decoding logic, the read and write logic, and the memory cells of the embedded memory. [15]

The memory BIST test typically consists of writing values based on a specific test algorithm to the memory and reading the written values from the memory. One such popular algorithm is the checkerboard algorithm, consisting of writing and reading 0s and 1s from the memory in a checkerboard pattern. Another popular family of algorithms are the march tests algorithms, consisting of applying sets of operations, called march elements to the memory. A curious reader can find many march test algorithms described in literature, for example in [7], [15] and [17], and even a methodology for creating your own march test algorithms, described in [18]. Memory BISTs using pseudo-random testing also exist but are not as common due to them providing worse fault coverage with a larger set of test patterns. [7][15]

A typical memory BIST, shown in Figure 19, consists of a memory BIST controller, an address generator, a test pattern generator, and an output response analyser. The memory BIST controller is used to create the testing control signals and can be implemented, for example, as a finite-state machine (FSM). The address generator is used to step through all the addresses in the memory. For march tests, the address generator should be able to generate addresses in both ascending and descending order. A reversible maximum-length LFSR is typically used as an address generator, instead of a reversible binary counter, due to the smaller area. The test pattern generator is used to generate test data based on an algorithm and can be made, for example, as an FSM. Output response analysis can be done with an MISR but is often done by deterministic comparison due to the regular structure of the memory. A reference data generator can be used to generate the expected read data and it can be compared against the actual read data. A mutual comparator can be used, when multiple identical memories are tested simultaneously, by comparing the read data from each memory to each other. [15]

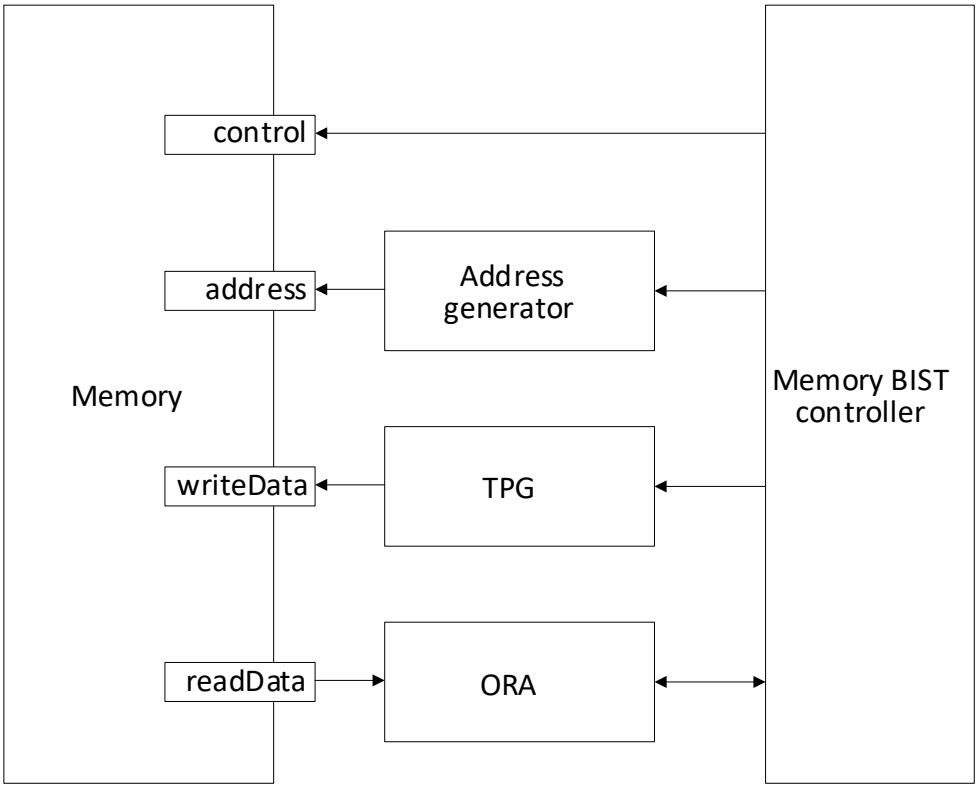

Figure 19. Structure of a typical memory BIST.

A simple and area optimal TPG can be implemented with a simple FSM that implements a fixed test algorithm. The downside is that the FSM needs to be changed every time the test algorithm is changed.

Several programmable memory BISTs architectures have been proposed in literature, for example in [19], that allow the test algorithm to be changed after manufacturing, providing flexibility. A common downside of programmable memory BIST architectures is the area overhead of the more complicated TPG. Some architectures reduce the area overhead by loading each operation one at a time instead of programming all the operations in the beginning, like is done in [20] with the help of scan chains, and in [21] and [22] with the help of an embedded microprocessor. The obvious downside is that the testing time is increased.

The architectures proposed in [19], [20] and [22] dealt with only a single embedded memory. Every embedded memory having their own memory BIST in a multi memory design would result in unnecessary area overhead.

Distributed memory BIST architectures reduce the area overhead of the BIST logic by sharing some of it for multiple memories. One such distributed memory BIST architecture is proposed in [23], that shares the BIST controller for all memories. A limitation is that all the memories need to be the same type and size [23]. An improved distributed memory BIST architecture is proposed in [24], that allows memories that have the same number of words or memories that have the same word width to share BIST logic.

3.4 Test point insertion

Designs usually contain many internal nodes that are hard to control or hard to observe, even in a scan or a logic BIST design. Test point insertion is a common method of improving the

controllability and observability of the internal nodes, by inserting control points or observation points [7].

Adding an observation point consists of adding a scan flip-flop and connecting a hard to observe node to its input. Observation points can be shared by connecting several hard to observe nodes through an XOR gate network to a single observation flip-flop. [7]

Control points consists of a scannable flip-flop and combinational logic connecting its output to a hard to control node. In a multiplexer control point, a multiplexer is inserted to the hard to control node. In test mode, the multiplexer selects the control flip-flop output to drive the hard to control node. In normal mode, the multiplexer selects the normal functional path. [7]

An OR control point, shown in Figure 20, is used to for improving the controllability of hard to control high nodes. The control point is OR gated with the original net driver, so that it can be used to set the hard to control node into high state. An AND gate is used to enable the control point in test mode and disable it during normal functional operation. The control value is loaded to the scannable flip-flop with a scan chain, during scan testing. [25]

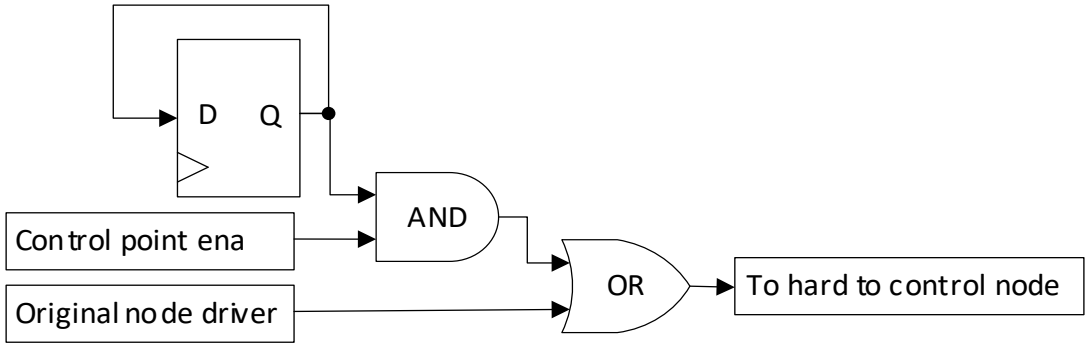

Figure 20. An OR control point.

An AND control point, shown in Figure 21, is used for improving the controllability of hard to control low nodes. The control point is AND gated with the original net driver, so that it can be used to set the hard to control node into a low state. A NAND gate is used to enable the control point in test mode and disable it during normal functional operation. The control value is loaded to the scannable flip-flop with a scan chain, during scan testing. [25]

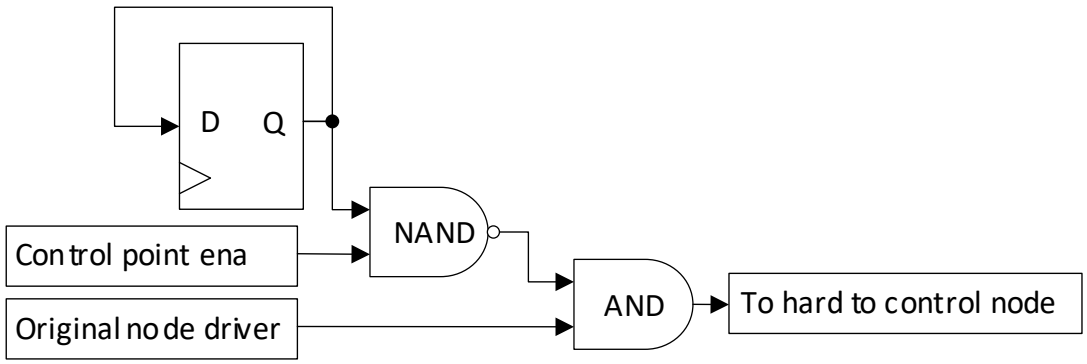

Figure 21. An AND control point.

A downside of using control points is that it adds additional delay to the logic path, that can cause hold time violations. Adding control points to critical paths should be therefore avoided. [7]

4 LATCH REPLACEMENT

This chapter presents an implementation of a latch-based register bank, using partial scan as the DFT method. Chapter 4.1 presents a reference PMIC model that is used as the starting point for the latch register design. Chapter 4.2 presents the reasons why flip-flops in the registers were selected to be replaced with latches. Chapter 4.3 presents the implementation of the latch registers, developed in this thesis. Chapter 4.4 presents the reasons for selecting partial scan as the DFT method. Chapter 4.5 presents the implementation of the partial scan design, developed in this thesis.

4.1 A reference PMIC model

A power management integrated circuit that is used to provide clean supply voltages to the rest of an electric device, is used as a reference design for this thesis. It consists of voltage regulators, converters, and their control circuitry. A simplified model of the reference PMIC is shown in Figure 22. It can be controlled by two host SoCs, with the inter-integrated circuit (I2C) communication protocol.

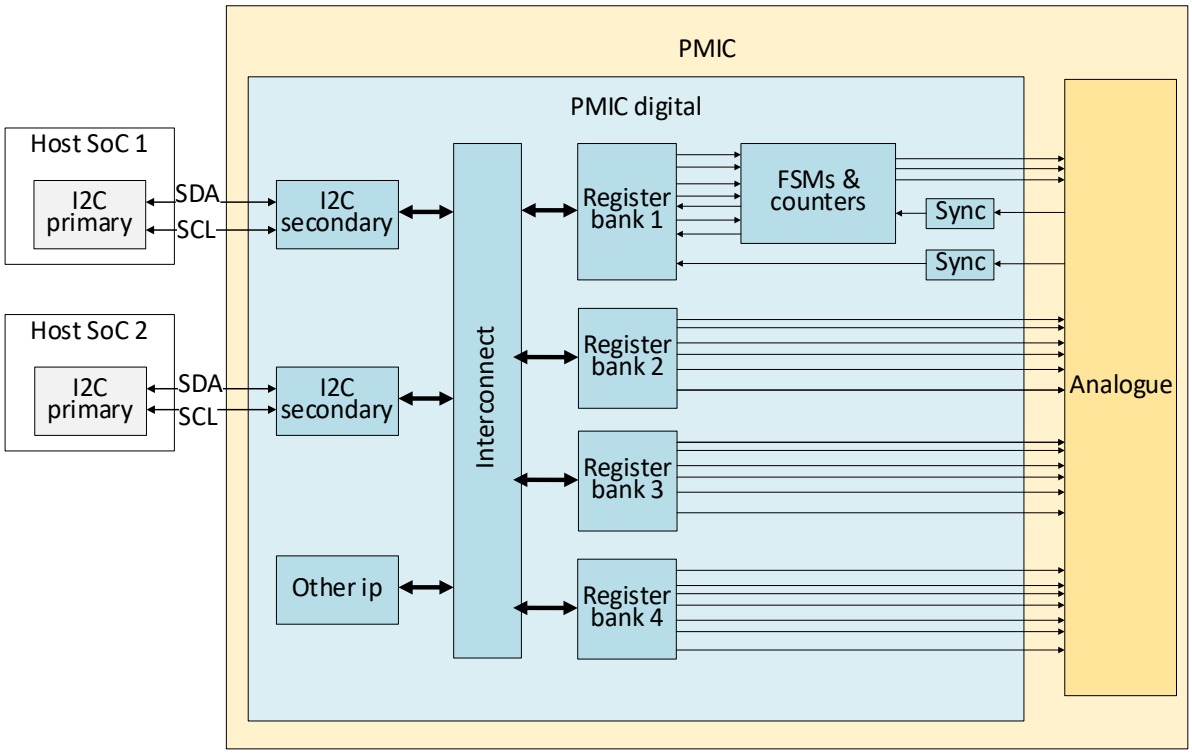

Figure 22. Simplified model of a PMIC.

The digital logic is used to control the operation of the PMIC, and it consists of several different blocks. The two I2C secondaries are used to convert the I2C transfers to the Advanced Microcontroller Bus Architecture (AMBA) Advanced Peripheral Bus (APB) protocol used by the interconnect. The interconnect is used to arbitrate several hosts accessing the same register banks simultaneously. It uses the APB protocol to access the register banks. The register banks are banks of control and status registers that a host SoC can write to control the PMIC or read the status of the PMIC. Finite state machines determine the state of the PMIC. They are

controlled by control registers and status signals from analogue logic blocks, and they create control signals to analogue logic blocks. Counters are used to provide accurate timing for state machines. Synchronizers are used to synchronize asynchronous status signals coming from analogue logic blocks.

4.1.1 AMBA APB protocol

A System-on-chip (SoC) often consist of several components, such as processor cores, memory, input or output (I/O) devices on the same chip. Connecting blocks using different interfaces and communication protocols would require the use of glue logic. [26]

Different bus protocols have been developed to allow easy integration and re-use of functional blocks. The advantage of designing functional blocks to use a common bus protocol is to allow them to be developed independently of each other. It also allows the functional blocks to be re-used easily in different designs utilizing the same protocol. [26]

The Advanced Microcontroller Bus Architecture (AMBA) by ARM is a standard for connecting and managing functional blocks in a SoC. It provides several bus protocols with different levels of performance. One such protocol is the Advanced Peripheral Bus (APB) protocol. [27]

The APB protocol is used to access the programmable control registers of low-bandwidth peripheral devices. It is a low-cost interface, optimized for minimum power consumption and reduced complexity. It can be interfaced with other, higher performance, AMBA bus protocols. [27]

The original APB specification Rev E was released in 1998. There have been several revisions since. The AMBA 2 APB Specification defined the interface signals, defined basic read and write transactions, and added the APB bridge and the APB secondary components. The AMBA 3 APB Protocol Specification v1.0 added wait states to the transactions and error reporting. The latest revision, AMBA APB Protocol Secification v2.0 added transaction protection and sparce data transfer. [28]

Signal transitions in the APB protocol happen at the positive edge of the clock. The protocol uses independent buses for read and write data, that can be up to 32 bits wide. The two buses cannot operate simultaneously, due to sharing handshake signals. [28]

Table 2 below presents the signals used in the APB protocol and short explanations on what they are [28]. The following sub-chapters present the read and write protocols. Error reporting, transaction protection and sparce data transfer are not relevant for this thesis and are not covered here. A curious reader can read the AMBA APB Protocol Specification v2.0 [28] to familiarise him- or herself with the additional features.

Table 2. AMBA APB protocol signals

Signal	**Description**
PCLK	Clock.
PRESETn	Active low reset.
PADDR	Secondary address. APB bridge indicates which location in the secondary device is accessed.
PPROT	Protection type. APB bridge indicates protection level of access. Tied low when not used.
PSELx	Secondary selection. APB bridge indicates which secondary device is accessed.

PENABLE	Transfer enable. APB bridge enables the transfer.
PWRITE	Transfer direction. APB bridge indicates write access when high and read access when low.
PWDATA	Write data. Bus containing write data from APB bridge.
PSTRB	Write strobes. APB bridge indicates which byte lanes are updated during a write transfer.
PREADY	Ready handshake. Secondary device indicates write or read access has been finished. Tied high when read access or write access is not extended.
PRDATA	Read data. Bus containing read data from a secondary device.
PSLVERR	Transfer error. Secondary device indicates read or write transfer failure. Tied low when not used.

4.1.1.1 APB write transfer

Figure 23 below shows the AMBA APB write transfer protocol. The setup phase of the write transfer starts at time 2 with the register address PADDR and write data PWDATA becoming valid. At the same time select PSEL and write PWRITE signals are set high to indicate that this secondary device has been selected and that it is a write access, respectively.

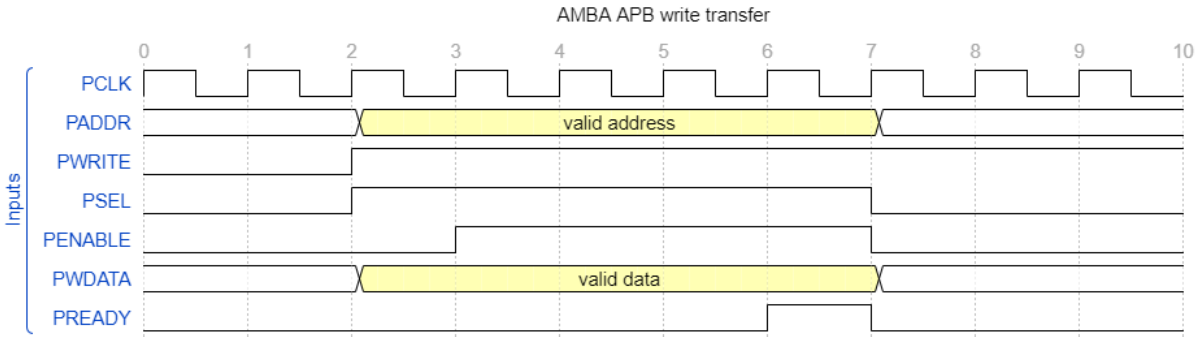

Figure 23. AMBA APB write transfer.

The access phase starts at time 3 with the enable PENABLE signal being set high. The secondary device indicates that it can complete the transfer on the next clock edge by setting the ready PREADY signal high at time 6. The access phase ends at time 7 and the enable PENABLE signal is deasserted. In case there is another transfer to the same secondary device, the select signal PSEL is left high. Otherwise, it is also deasserted at the end of the access phase. [28]

The address PADDR, write data PWDATA and control signals need to remain valid during the setup and access phases, from time 2 to 7. The address PADDR, write data PWDATA and write PWRITE signals can be left as is after the transfer until another access, to reduce power consumption. [28]

The ready PREADY signal is used to extend the write transfer by any number of cycles from zero upward. The ready PREADY signal can have any value when the enable PENABLE signal is low. When the enable PENABLE signal is high, it indicates that the secondary device can finish the access on the next clock edge. In the case that the secondary device has a fixed two cycle access, the PREADY signal can be tied high. [28]

4.1.1.2 APB read transfer

Figure 24 below shows the AMBA APB read transfer protocol. It is similar to the write transfer protocol presented in section 2.2.1. The read data RDATA bus is used in the read access instead of the write data WDATA bus. The difference in the setup phase, at time 2, is that the PWRITE signal is set low to indicate that it is a read access and the read data PRDATA bus does not have to contain valid data. [28]

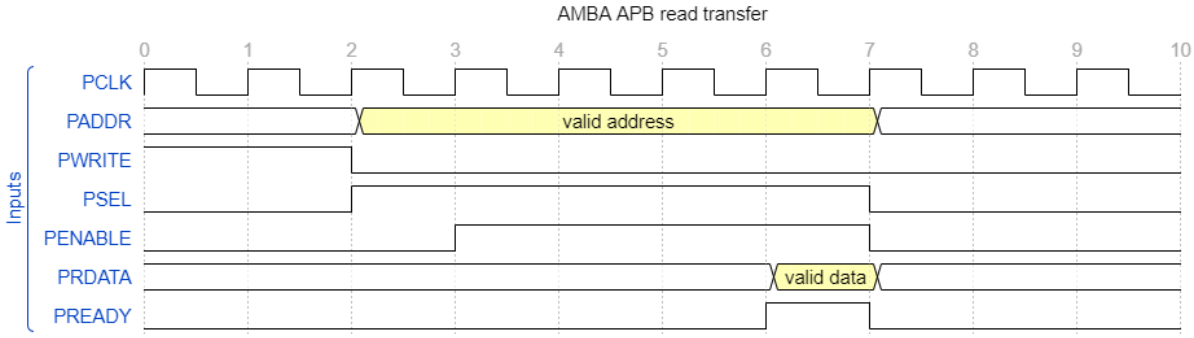

Figure 24. AMBA APB read transfer.

The access phase also starts at time 3 with the enable PENABLE signal being set high. The secondary device sets the ready PREADY signal high, at time 6, to indicate that the read data RDATA bus contains valid data. [28]

4.1.2 Register banks

The PMIC contains several different register banks. The banks are divided into one bank for digital registers, and three banks for different analogue logic blocks. The register bank 1 contains all the control registers and status registers for digital logic. The register banks 2-4 contain control registers used to control different analogue logic blocks directly.

The banks are made of several different types of registers, that can be 1 to 8 bits wide. In total, there are 443 registers, storing 1973 bits of data. On average, one register is storing about 4.5 bits of data.

The PSEL signal of the APB protocol is used to select which register bank is accessed. The PADDR signal is used to select which register is accessed in the bank. If the register being accessed is less than 8 bits wide, the most significant bits (MSB) of the PWDATA signal are discarded in the write access and the PRDATA signal is zero padded to 8 bits in the read access. The PREADY signal is tied high in the register banks to indicate the write and read accesses are not extended.

Since the read and write buses are a byte wide, the PSTRB signal is unnecessary and not used in the register banks. The PPROT and PSLVERR signals are also not used and tied low.

4.1.2.1 Read-write register

The most common type of register used is the read-write register shown in Figure 25. It is a simple parallel loaded register that can be written to and read from with the APB protocol. During the read access, when PWRITE, PSEL, PENABLE signals are high and the PADDR indicates this specific register is accessed, the multiplexer is used to select the 4 least significant bits of the PWDATA that are loaded into the register. During all other times, the multiplexer selects the output of the register, and the old value is kept. The read-write register, of Figure 25, implemented in SystemVerilog, is shown in Figure 26.

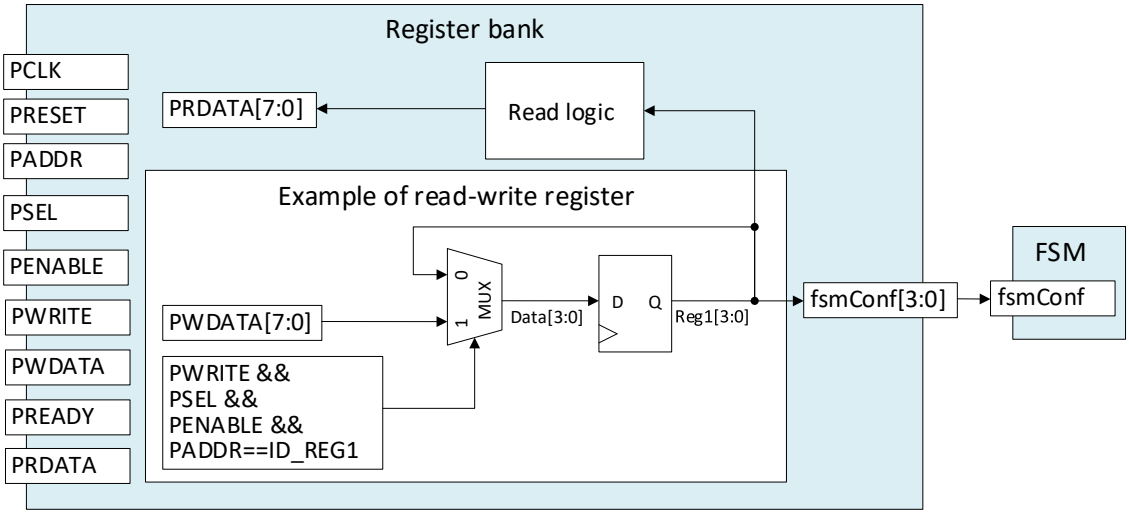

Figure 25. Example of a read-write register.

```
always_ff @(posedge PCLK or posedge PRESET) begin : la_ReadWrite_Register
  if (PRESET) begin
    Reg1 <= CONST_RESETVALUE_REG1;
  end else begin
    if (PWRITE && PSEL && PENABLE && PADDR == CONST_ID_REG1) begin
      Reg1[3:0] <= PWDATA[3:0];
    end
  end
end : la_ReadWrite_Register

assign fsmConfig[3:0] = Reg1[3:0];
```

Figure 26. Example of a read-write register coded in SystemVerilog.

From the area and power consumption point of view, it is beneficial to use clock gating to control the loading of multi-bit registers instead of multiplexers. In that case, the multiplexers, of Figure 25, are replaced with a single clock gate. PWDATA is now connected straight to the inputs of the flip-flops and loading new data to the register is controlled by the clock gate. Synthesis tools can do this optimization automatically when automatic clock gating is enabled. The Design Compiler synthesis tool, made by Synopsys, gates the clock for registers that are 3-bits or larger, by default [29].

4.1.2.2 One-time programmable register

The exact output voltages for voltage regulators vary with process variations. This limitation is often overcome by trimming the output voltage to the correct value. The problem is that the correct trimming values are only known after manufacturing, while the reset values of normal registers must be defined before manufacturing. The use of one-time programmable (OTP) memory allows the default values of OTP registers to be programmed after manufacturing.

The OTP register, shown in Figure 27, is the second most common type of register used. The OTP register consists of a flip-flops and OTP logic. The flip-flops store the state of the OTP register. The OTP logic is combinational logic that determines the next state of the OTP register, based on the write data and the OTP values in OTP memory, and decodes the output of the OTP register based on the state of the OTP register and the OTP values in OTP memory. The use of the OTP register, and not just the OTP memory, allows the value of the OTP register to be changed with a register write. In case of a reset, the OTP register is reset back to the OTP value, coming from the OTP memory.

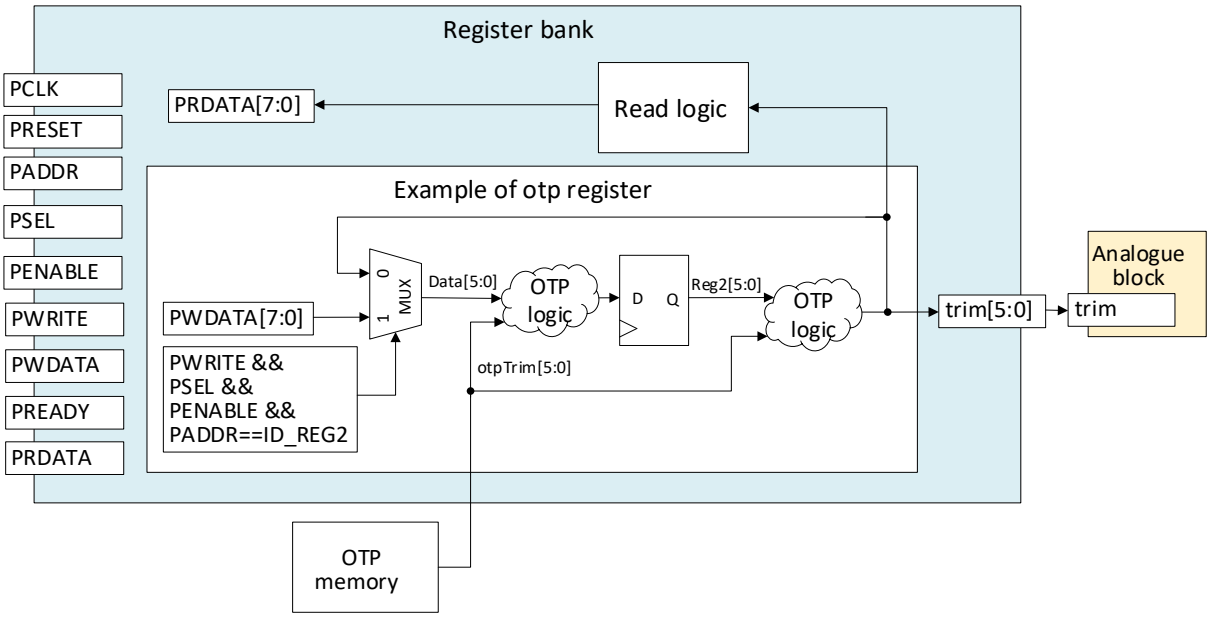

Figure 27. Example of an OTP register.

4.1.2.3 Set-clear register

The set-clear register, shown in Figure 28, can be used to turn some functionality on or off bit wise, without modifying the value of other bits. When a 1 is written to a bit position with the set address, the flip-flop corresponding to that bit location is set to 1. When a 1 is written to a bit position with the clear address, the flip-flop corresponding to that location is cleared to 0. Writing 0 to a bit location with the set or clear address does not change the state of the register. The state of the set-clear register can be read from both the set and clear addresses. The reset value of each bit location can be defined to be either 1 or 0.

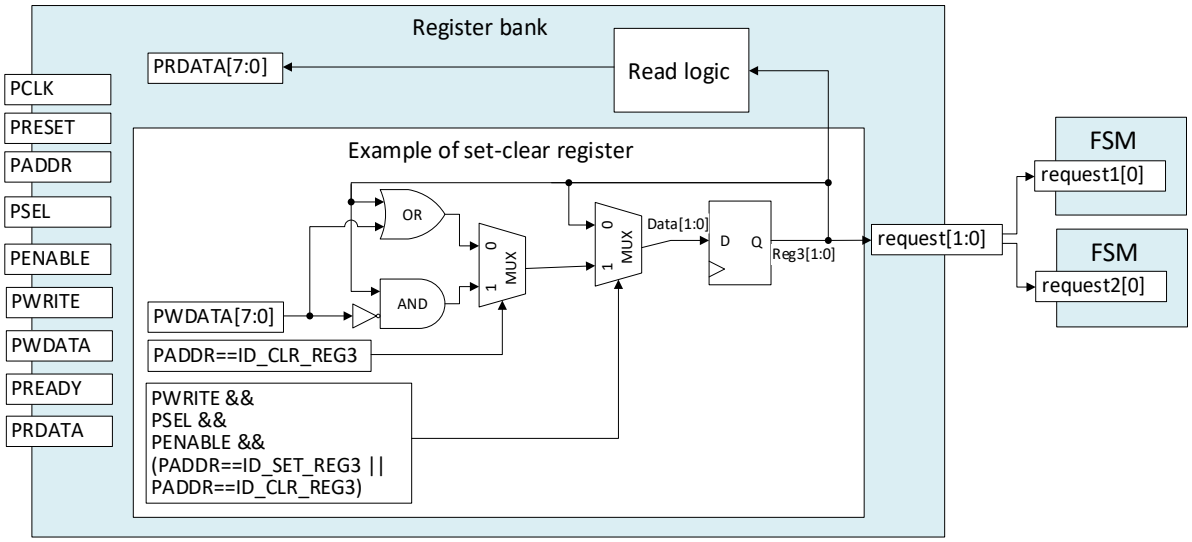

Figure 28. Example of a set-clear register.

The advantage of using a set-clear register is that the controlling host can only set or clear some specified bit locations in the register, while not affecting the others. This is especially useful when there are several hosts controlling the same registers. For example, if the PMIC contains several LDO regulators utilized by different hosts, they can be independently turned on and off with the set-clear register without the different hosts needing to know whether the LDO regulator utilized by a different host needs to be on or off. With a read-write register, all the bit locations are updated on each write and the host needs to know what value to write into each bit location. Reading the state of a read-write register, modifying the read data, and writing the modified data back takes more time and power, than using a set-clear register.

4.1.2.4 Task register

The task register, shown in Figure 29, is used to execute some functionality once. When a 1 is written to a bit position in the task register, a one clock cycle positive pulse is created in the corresponding flip-flop. This is caused by the fact that a 0 is loaded to the flip-flop, when the task register is not being written to. Writing a 0 to a bit position in the task register does not affect the state of the corresponding flip-flop. The reset value of the task register needs to be 0 for all bit locations.

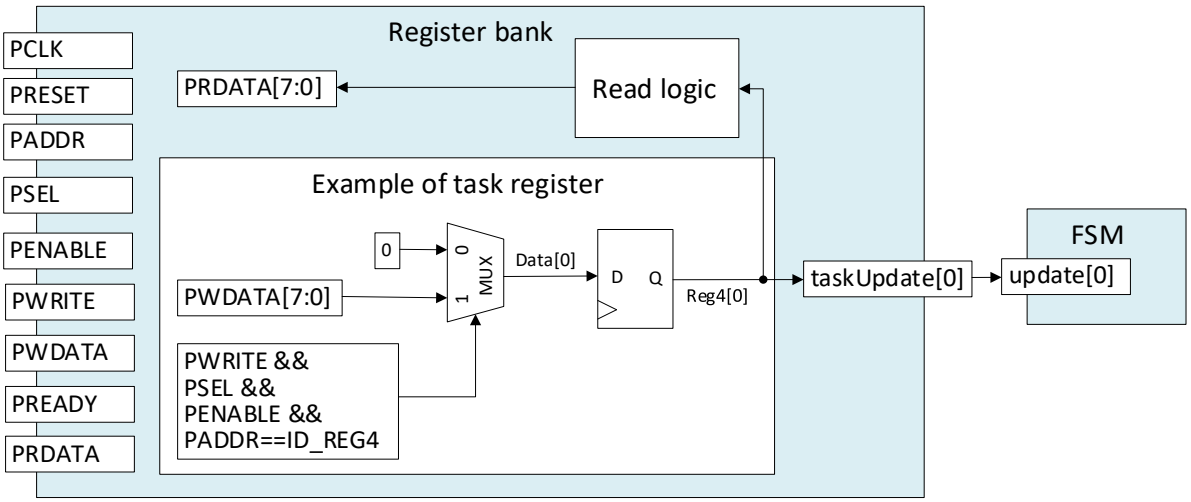

Figure 29. Example of a task register.

The advantage of using a task register is that the controlling host can write pulse like controls to the registers with a single write access. Writing a 1 and then a 0, like would need to be done with a read-write register, takes more time and power than using a task register.

4.1.2.5 Read-only register

A read-only register, shown in Figure 30, differs from the other types of register as it is not actually a register but just a signal mapping that can only be read. A digital signal or a combination of several signals can be routed to a multiplexer input that is selected by PADDR. The flip-flops after the read logic are not necessarily needed but are used to reduce timing problems. If the combinational read logic is large and flip-flops at the output of the register bank are not used, the setup time from the flip-flops in the registers, to the flip-flops in the interconnect read logic could be easily violated.

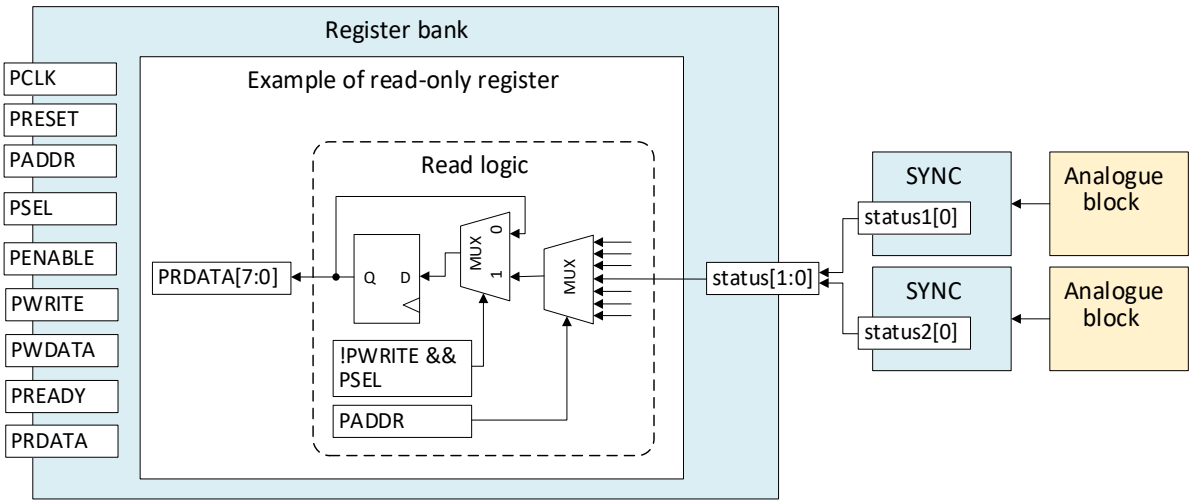

Figure 30. Example of a read-only register.

4.2 Latch replacement selection

Latches and pulsed latches are an enticing alternative to flip-flops from the area point of view. However, it is important to not just consider the area benefits of replacing flip-flops with pulsed latches or latches, but to also consider the effect on power consumption and hidden costs. Power consumption is an extremely important metric, especially for battery powered devices, and it should not be increased from the replacement. Another important consideration is the effect of the replacement on the time to market of the products it is implemented in. If the design difficulty is increased significantly or the design flow made longer, the replacement may become unpractical.

4.2.1 Pulsed latch considerations

When considering the pulsed latch style, the effect of a pulser on power consumption needs to be considered. A pulser creates pulses from the clock and is therefore switching all the time. It stands to reason that the switching power consumption of a pulser is large compared to a flip-flop or a latch. The pulser in [30] consumes almost 14 times as much power as a single latch and 7 latches need to share a single pulser to achieve slightly lower power consumption than 7 flip-flops. It was found in [31], that 8 latches need to share a pulser, so that the power consumption is slightly smaller than that of 8 flip-flops.

In the case of the reference design presented in chapter 4.1, the maximum register width is 8 bits, but the average register width is only roughly 4.5 bits. The other arrays of flip-flops, found in counters and FSMs, are similarly much less than 8 bits wide on average. To get the power consumption lower than that of flip-flops, several registers would need to share the same pulser. The pulser was integrated into the clock gate of the registers in [31]. Sharing a pulser for several registers would prevent its integration with the clock gates and the power consumption of the pulser would increase since it would not be gated. Pulsed latches could be only used for the largest registers to keep the power consumption down, but it would negate much of the area benefit of using latches. From the power consumption point of view, the pulsed latches are not ideal in this case.

The effect of pulsed latches to the synthesis flow needs to be considered. A pulsed latch synthesis flow is presented in [32], that uses the normal flip-flop-based synthesis flow but adds additional steps, where the flip-flops are replaced, pulsers are placed, and their effect on timing is analysed and violations are fixed. From the synthesis flow point of view, pulsed latches cause extra synthesis steps and are not ideal.

The pulsed latches are suited for scan testing. The scan synthesis flow and pattern generation could be done with the flip-flop-based design flow and then the scannable flip-flops could be replaced with scannable pulsed latches.

Another consideration is the technology library support. The library used for this thesis does not contain any pulsers. Pulsers can be made from existing discrete library cells, but the area would be much larger than that of a custom pulser library cell. For full area benefit, custom pulser library cells would need to be created.

4.2.2 Latch considerations

Since a primary-secondary flip-flop is made of two latches, it would stand to reason that short circuit and leakage power consumption of a latch would be roughly half of that of a flip-flop. It would also stand to reason that the switching power consumption of the clock pin of the latch would be half of that of a flip-flop, as the clock pin of the flip-flop is driving two latches. The data input of a flip-flop is only driving a single latch, so the switching power consumption the data input would be expected to be roughly the same. The switching power consumption of the data output would also be expected to be the roughly same, assuming the same load. The technology library used in this thesis mostly follows these assumptions, except for the switching power consumption of the clock pin. The load capacitance of the clock pin of the latch is roughly double that of a flip-flop, also leading to roughly double the switching power consumption. The unexpected switching power consumption of the technology library used leads to the power consumption of latches to be higher than that of flip-flops. Since the pulser is eliminated by using latches instead of pulsed latches, the latch design style is still preferred over the pulsed latch style, from the power consumption point of view.

In addition to the power consumption, latches also have the benefit that they fit to the flip-flop-based synthesis flow without additional steps and there is no need to create a custom pulser library cell, compared to pulsed latches.

Latches however have a large drawback compared to flip-flops and pulsed latches. Due to the transparent nature of latches, feedback from the output of a latch to its input is not allowed. Two back-to-back latches being transparent at the same time is also not allowed. These restrictions mean that only some flip-flops in a flip-flop-based design can be replaced with latches, without serious redesigns.

A basic synchronizer is just two series connected D flip-flops, where the second flip-flop is used to block metastability propagating from the first flip-flop. D latches, clocked with normal 50% duty cycle clocks, are not well suited for synchronizers. Connecting two D latches to the same clock would mean both are transparent at the same time and the asynchronous signal can propagate through them. The clock to one of the latches can be inverted, but now we have a primary-secondary flip-flop. At least a third latch would need to be added, that removes the metastability of the primary-secondary flip-flop. A problem is that the first flip-flop would have half the time to recover from a metastable state with a third latch, compared to the two-flip-flop synchronizer.

Counters consist of flip-flops, that store the value of the counter, and combinational logic, that calculate the next value of the counter. The current value of a counter is used in calculating the next, meaning there are feedback paths from the outputs of the flip-flops to the inputs. Latches are not well suited to replace the flip-flops as the feedback will cause race conditions. The counters would need to be redesigned without feedback paths from the output of a latch to the input of the same latch, leading to increased design difficulty.

State machines similarly consist of flip-flops and combinational logic. Similar feedback paths exist, from the outputs of the flip-flops to the inputs, that are used in determining the next state of the FSM, making latches not well suited for FSMs.

Register banks consists of read-write, OTP, set-clear, task and read-only registers. Their structure is presented in chapter 4.1.2. The read-write and OTP registers contain a feedback path used for keeping the old value of the register. This feedback can be removed by using clock gating, instead of multiplexers, to control whether a new value is loaded. The feedback paths in set-clear registers are used to determine the loaded value based on the previous value of the register and cannot be removed with clock gating, making set-clear register not suitable

for latches. The read-only register does not contain flip-flops in the register bank as it is a signal mapping.

Read-write and OTP registers are by far the most common types of registers, and they make up of roughly 30% of all flip-flops in the design. Even with only replacing their flip-flops with latches, large area gains are available.

4.3 Latch register design

The read-write register, shown in Figure 25, contains a feedback path through a multiplexer, that is used to keep the register value when it is not written to. The feedback can be removed by using clock gating to control the loading of the register instead of a multiplexer, allowing the flip-flops of the read-write registers to be replaced with latches.

Now we have a situation where there is a path from the flip-flops generating PWDATA signals to the latch registers, and from the latch registers to flip-flops in the read logic. A problem with connecting a positive level active latch between two positive edge triggered flip-flops is that a change in the output of the first flip-flop on a clock edge will propagate straight through the latch to the input of the second flip-flop, which might cause a hold time issue. A possible solution would be to make sure the data at the output of the first flip-flop is stable before enabling the clock to the latch, by modifying the write protocol, and setting the path from the first flip-flop to the second flip-flop as a multi-cycle path. This solution would result in a 3 cycle APB write access, so it is not optimal. A better solution is to latch the write data through a negative level active clock.

The resulting read-write register made with latches and clock gating is shown in Figure 31. The negative level active write latches load the data in PWDATA to wDataLat on the negative level of the clock. The clock gate contains a negative level triggered latch and an AND gate, that are used to create the gated clock signal. When the register is written to, the clock gate is enabled, and a clock pulse is provided to load the value from wDataLat into the register latches on the positive level of the clock. The read-write latch register, of Figure 31, implemented in SystemVerilog, is shown in Figure 32.

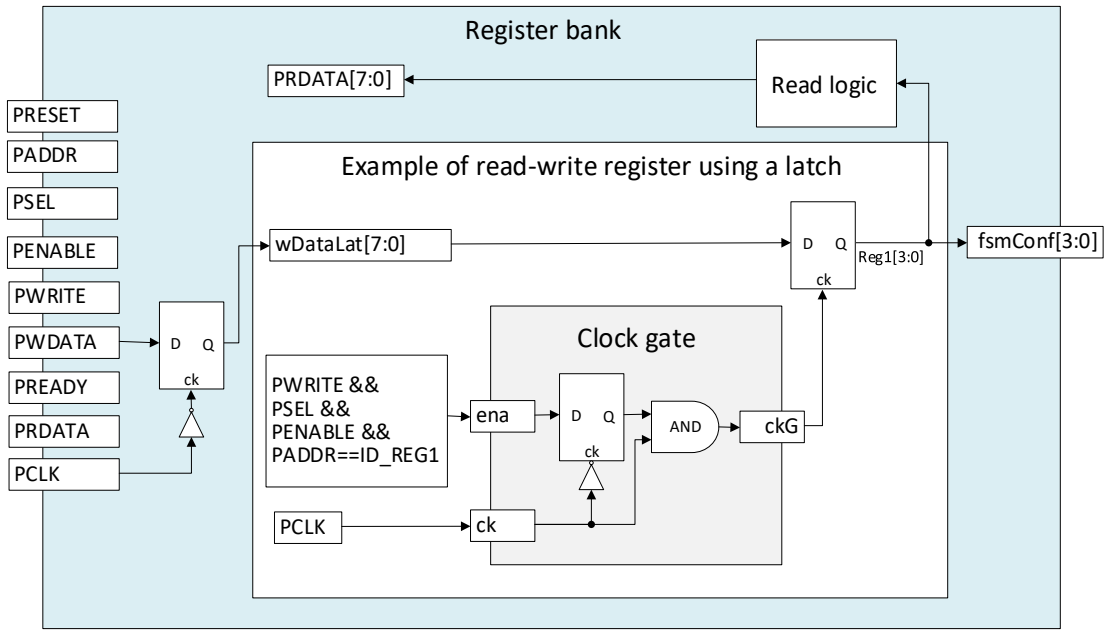

Figure 31. Example of a read-write register using latches and clock-gating.

```
// Clock gate enable
assign ena = (PWRITE && PSEL && PENABLE && PADDR == CONST_ID_REG1);

// Clock gate
ClockGate u_ClockGate(.ck(PCLK), .ena(ena), .ckG(ckG));

// Latch register
always_latch begin : la_ReadWrite_LatchRegister
  if (PRESET) begin
    Reg1 <= CONST_RESETVALUE_REG1;
  end else begin
    if (ckG) begin
      Reg1[3:0] <= wDataLat[3:0];
    end
  end
end : la_ReadWrite_LatchRegister

assign fsmConfig[3:0] = Reg1[3:0];
```

Figure 32. Example of a read-write latch register coded in SystemVerilog.

By latching the write data, PWDATA, through negative level active latches instead routing it straight to the register latches, positive edge triggered operation can be maintained, with the negative level triggered write latches acting as the primary latches and the positive level triggered register latches acting as the secondary latches. A difference to a flip-flop-based implementation is that the master latches can be shared by all the latch-based registers in a single register bank.

OTP register, shown in Figure 27, also contains a feedback path through a multiplexer that can be implemented with a clock gate. An OTP register made with latches and clock gating is shown in Figure 33. When the register is written to, the clock gate is enabled, and a clock pulse is provided to load the value determined by wDataLat and otpTrim into the register latches. There is a possible race condition if the OTP values come from a positive level triggered source. This is not however the case as the OTP values are stable during normal functional operation of the circuit.

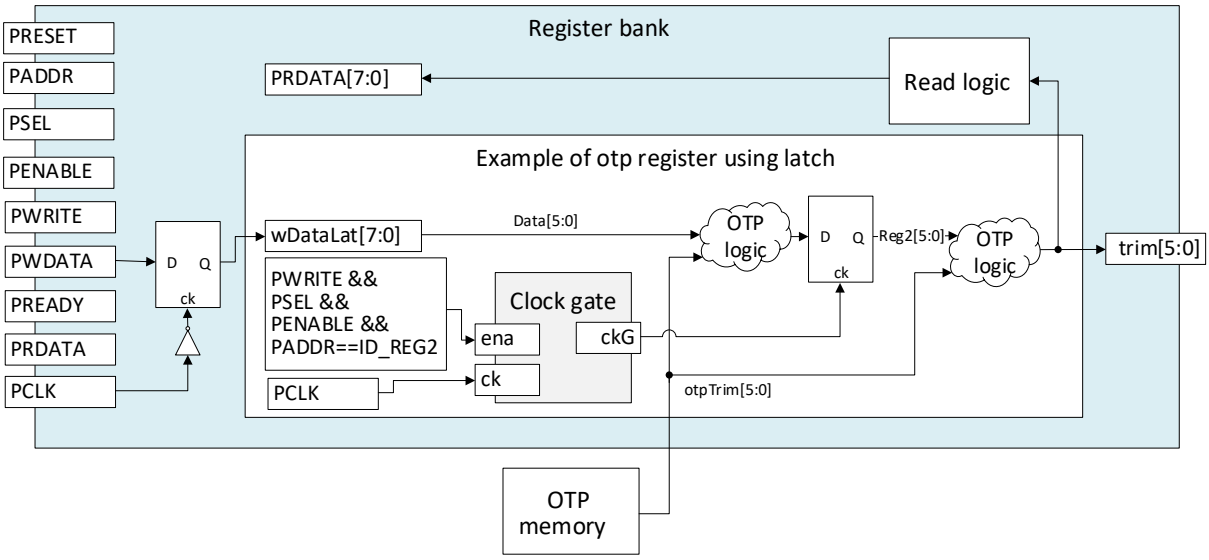

Figure 33. Example of an OTP register using latches and clock-gating.

A write access to the read-write latch register of Figure 31, is shown in Figure 34. The setup phase starts at time 2, when the address PADDR indicates this register is accessed and write data PWDATA become valid. At the same time, PSEL and PWRITE signals go high to indicate that this register bank is selected and that it is a write transfer, respectively. At time 3, the write access starts by PENALBE signal going high, causing the clock gate enable to go high. PREADY signal being tied high indicates the write access can be completed on the next cycle. The write access finishes at time 4, with PENABLE signal going low. At the same time, a clock pulse is created through the clock gate and the data in wDataLat is loaded to the register latches.

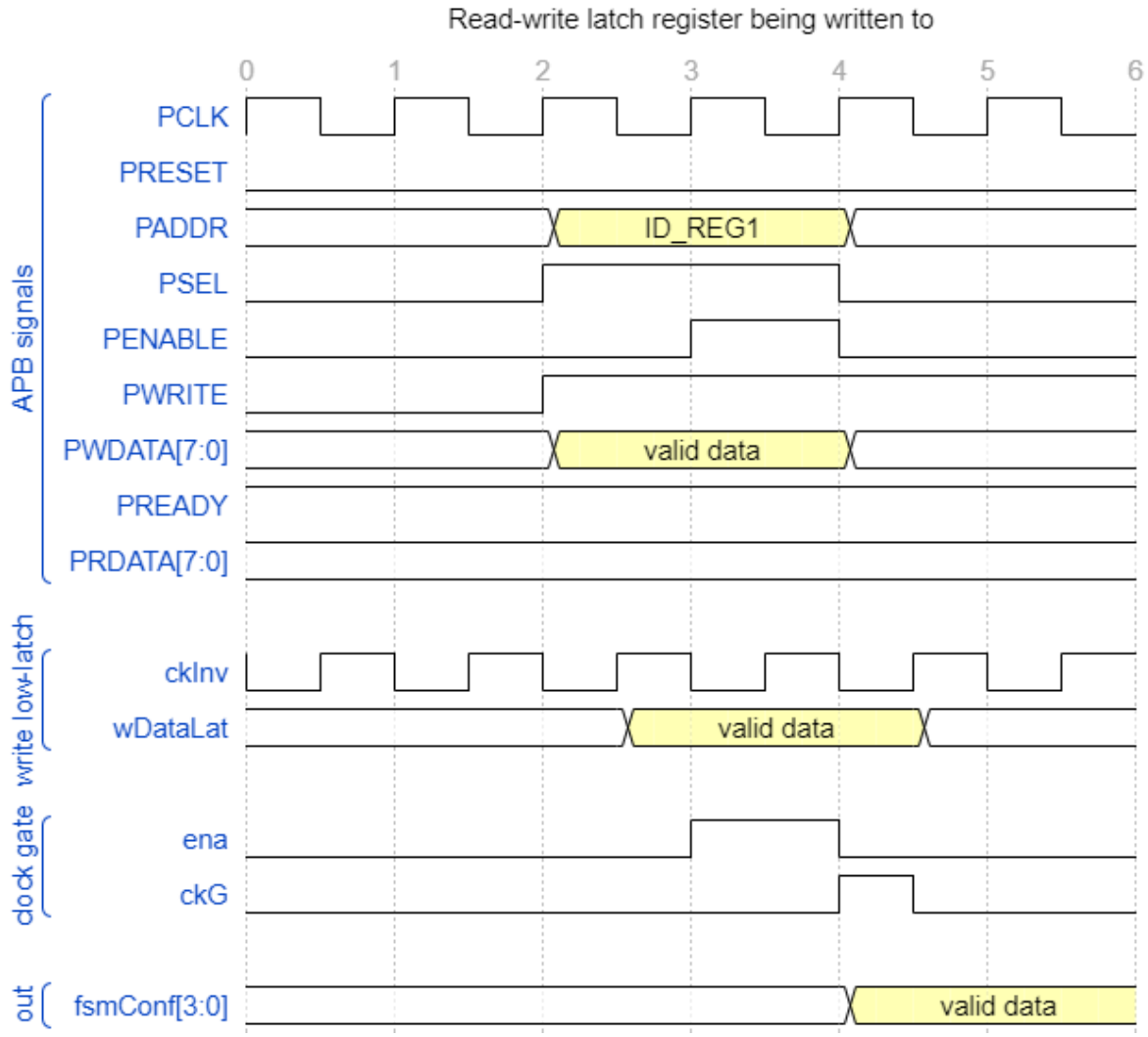

Figure 34. A read-write latch register being written to with the APB protocol.

If the negative level active latches, creating the wDataLat signal, were not used, correct data from PWDATA would be first loaded to the latch register at time 4, but then invalid data would propagate to the latch register and be loaded to it. The negative level active latches keep the data at the input of the latch register valid for half a clock cycle longer, until the latch register clock becomes inactive.

The use of latches does not affect the APB read access, as the clock to the latch register is inactive during the whole access and the latch will keep its value like a flip-flop would. A fixed two cycle read access to the read-write latch register of Figure 31, is shown in Figure 35. The setup phase starts at time 2, when the address PADDR indicates this register is accessed. At the same time, PSEL signal goes high and PWRITE signal goes low, to indicate that this register bank is selected and that it is a write transfer, respectively. At time 3, the read access starts by PENALBE signal going high. PREADY signal being tied high indicates the read access can be finished on the next cycle. The read access finishes at time 4 with the PENABLE signal going low.

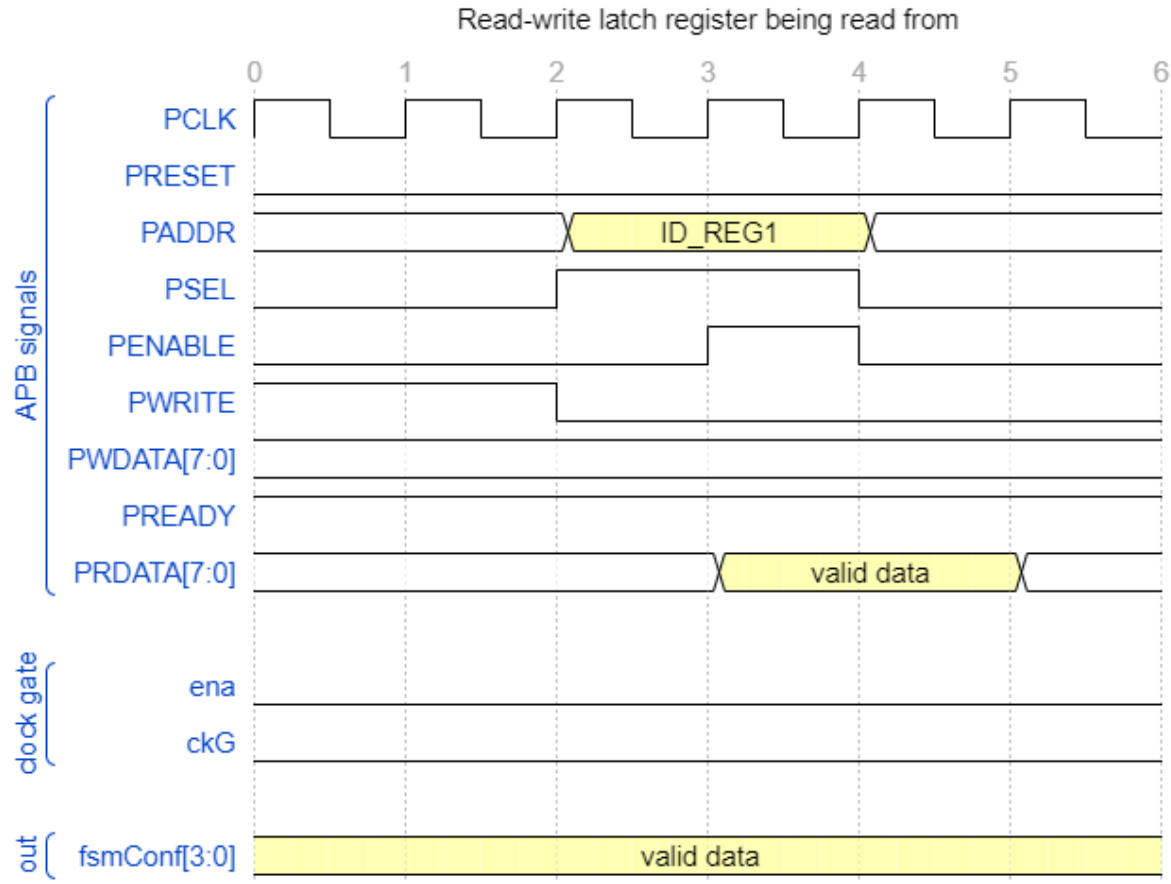

Figure 35. A read-write latch register being read from with the APB protocol.

4.4 DFT method selection

Manufacturing testing is an important step in ensuring that defective components are not shipped to customers. For the latch replacement to be practical, for actual products, the fault coverage should not be decreased significantly, while keeping the area overhead and testing time of the DFT solution manageable. The different DFT options are considered in this chapter, and one is selected to be studied further.

4.4.1 Scan design considerations

The flip-flop-based reference design is manufacturing tested using the muxed-D scan style. From the design flow perspective, using the muxed-D scan style for the latch register design would be preferred. Scan testing is a well-known and often utilized method to test flip-flop-based circuits. The transparent nature of latches makes them ill-suited for use in scan chains.

4.4.1.1 Full-scan design consideration

For the controllability and observability of a full-scan design to be preserved, the latches would need to be made scannable. A proposed custom muxed-D scan cell is shown in Figure 36. It

acts as two latches in normal mode and as a single flip-flop, with both data inputs XORed together, in scan mode, SM.

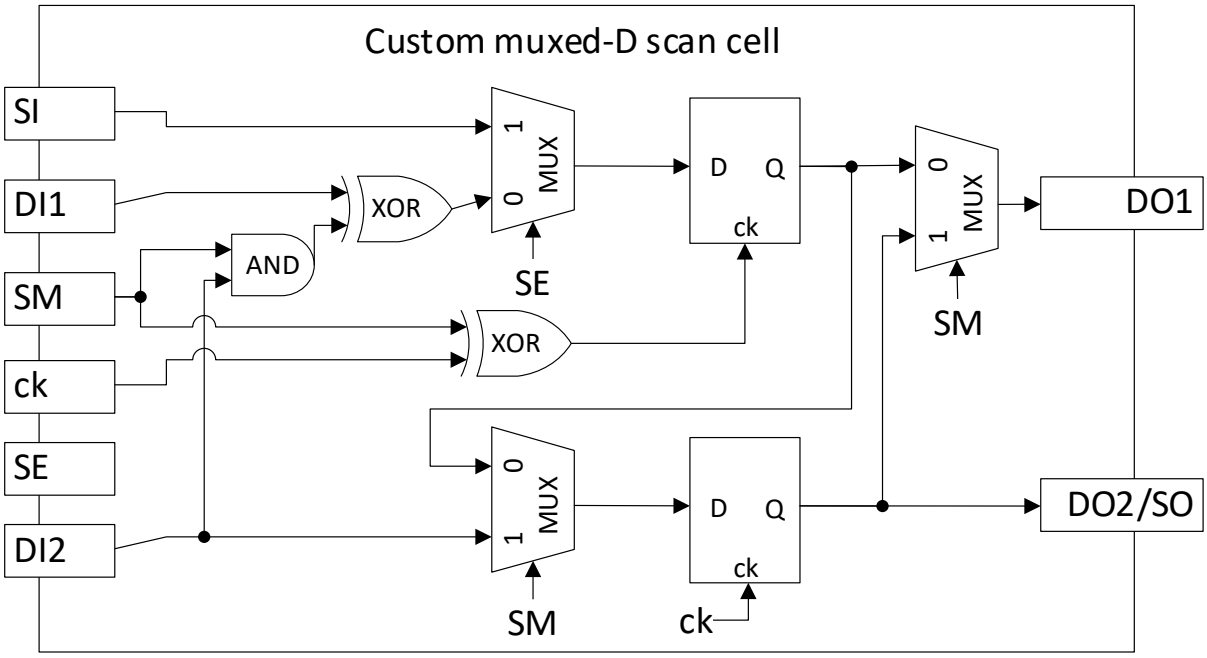

Figure 36. A custom muxed-D scan cell.

The downside of this proposed scan cell is that the area benefit of using latches is destroyed by all the testability logic. Another disadvantage is that the data in both data outputs is the same in scan mode, SM, which could make some faults downstream of the scan cell untestable.

LSSD scan style has been typically used for scan testing latches. A downside is that it would require converting the design into LSSD style and creating LSSD library cells. The standard LSSD style is not suitable, as the added dummy latches would destroy the area benefit. The optimized LSSD style in [11], is better from area point of view. The downside of it is that it requires two different scan modes and the logic downstream of the L1 and L2* latches to be able to be tested independently of each other. The scan modes could be combined by using an XOR gate to combine both inputs together and using a multiplexer to route the L2 output to the L1 output, similarly to the proposed muxed-D scan cell of Figure 36. The area overhead of the optimized LSSD style would be smaller than using the muxed-D scan cell of Figure 36, but it would still not be very optimal.

From the other full-scan methods briefly introduced in chapter 3.1.4, the asynchronous scan-latch controller, presented in [14], is also an unsuitable solution. Integrating the controller on-chip would be unacceptable due to the area overhead of the controller and the clock tree. Having the controller off-chip is unrealistic due to the need for one test clock for each latch in the design.

4.4.1.2 Partial-scan design consideration

Since the area overhead that would result from making latches transparent is unacceptably large, leaving them non-scannable can be considered, resulting in a partial scan design. By leaving the latch registers non-scannable, they need to be sequentially tested, with a sequential depth of

2. The controllability and observability are reduced, as the latches and the logic around the latches now needs to be controlled from flip-flops in the write logic and observed from flip-flops in the read logic.

The reduced controllability and observability can negatively affect the fault coverage. The fault coverage loss should be able to be minimized, to an acceptable level, by improving the testability with control and observation points.

The reduced controllability and observability can also result in more test patterns being needed, for the same fault coverage, resulting in increased testing time. The number of test patterns needed can also be reduced with control and observation points, but likely not enough to match a full-scan design, without increasing the area significantly.

4.4.2 Logic BIST considerations

Logic BIST schemes require the latches to be made scannable, to reach high fault coverages, leading to high area overhead from the scan testability logic. They also suffer from the added area overhead of the BIST logic. Logic BIST schemes are unsuitable in this case, as in the field testing is not required, and they are not investigated further.

4.4.3 Memory BIST considerations

Memory BIST schemes have been found suitable for manufacturing testing large, embedded memories. The memory in this case differs from typical memories, bringing several challenges in implementing a memory BIST.

Embedded memories are often large, making the area overhead of the BIST logic negligible. In this case, only 1973 bits are stored in the register banks and of those, only 1618 bits are in read-write and OTP registers. Making the memory BIST small enough not to increase the area significantly would be a real challenge.

Embedded memories are usually uniform, made from the same type, width, and length memory. In this case the memory is made of several different types of registers, with different widths. Due to the irregular structure of the memory, the output response analysis and the address generation is made more complicated. The known distributed memory BIST architectures, of chapter 3.3, are also not suitable due to the irregular structure.

The expected read data depends on the type and width of the register accessed. Generating the expected read data on-chip without storing this information seems unfeasible and storing the type and width of the register in each address would lead to large area overhead. A MISR and comparing the output to a golden signature would most likely need to be used, leading to area overhead and some fault masking.

Since the register banks in this case only contain the number of registers that is necessary for the design, the number of registers is unlikely to be a power of 2. This would result in empty register locations being tested when using an LFSR as the address generator, leading to unnecessarily long testing time. A binary counter could be used as the address generator to avoid this issue, but the area would be larger.

Another structural difference is that the outputs of the registers in the digital register bank are connected to the rest of the digital logic. To facilitate the testing of the digital logic downstream of the registers, the latches would need to be made transparent or bypassed during scan testing anyways.

For the above reasons, memory BIST schemes are not well suited in this case, and they should be only considered, if the fault coverage from a partial scan design is deemed too low. Partial scan is deemed the most promising option and selected as the DFT style to be researched further.

4.5 Partial scan design

To keep the area benefit of using latches, they can be left non-scannable, resulting in a partial scan design. The partial scan design can be sequentially tested, by making the latches transparent during the shift operation, and then using multiple capture cycles to sequentially observe the latches.

The reduced controllability and observability of the partial scan design results in loss of fault coverage and an increased number of test patterns. The loss of fault coverage can be reduced, with the strategic use of control and observation points. The main downside of the partial scan design is the increased number of test patterns, which cannot be so easily decreased without large area overhead.

Chapter 4.5.1 presents how latches were made transparent during scan testing. Chapter 4.5.2 present how the observability was increased. Chapter 4.5.3 presents how the controllability was increased. All the modifications to the reference design, using the replacement technique developed in this thesis were made in the register-transfer level (RTL), using SystemVerilog as the coding language.

4.5.1 Making latches transparent

To maximise the fault coverage of the partial scan design, the latches need be made transparent during scan testing. For the Tessent ATPG tool, made by Siemens AG, to identify the latches as transparent, the latches need to be transparent between setting values on primary inputs and measuring values from primary outputs, while the clock is low [25].

Transparency for the register latches can be achieved by OR gating the clock to all the positive edge active latches with the shift enable signal, shown in Figure 37. This allows the latch to be transparent when patterns are shifted in and out, but also allows the ATPG tool to manipulate the clock to the latches in capture, allowing the latches to be sequentially tested. The clock gate latches, and the write latches are negative clock level active and do not need any modifications to be identified as transparent.

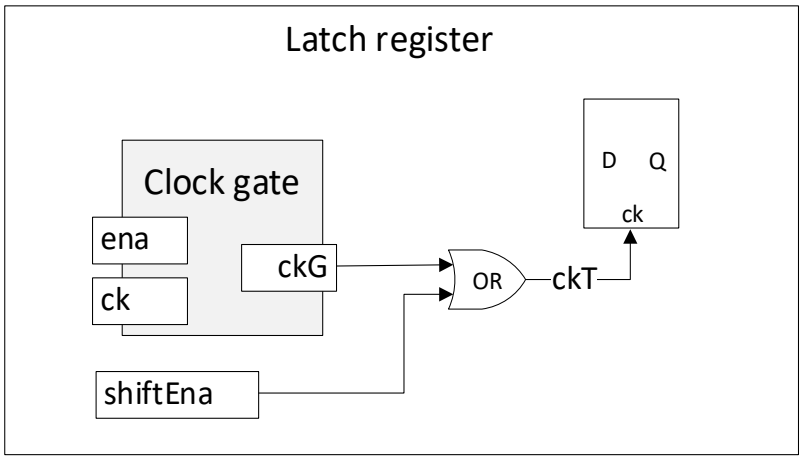

Figure 37. Making register latches transparent with an OR gate.

4.5.2 Improving observability

The enable for the clock gates of the latch registers is generated in a large combinational logic tree. When scannable flip-flops are used, faults in the clock gating logic can be seen in whether the value in the flip-flop is updated. In a latch-based register, the effect of the clock gating logic needs to be sequentially observed from flip-flops downstream of the latches. The structure of the analogue register banks means that only one latch register, in a register bank, can be observed at a time from the flip-flops in the read logic. The reduced observability results in a large decrease in the fault coverage of the clock gating enable logic.

A solution to this loss in fault coverage would be to add observation points for observing the clock gating logic of all the latch registers. This solution however would result in unnecessarily high area overhead and increased power consumption.

A more area and power efficient solution is to keep the first bit of each latch register as a flip-flop, instead of replacing all the flip-flops with latches. Since the clock gating logic is the same for all the bits in a register, by keeping one flip-flop in each register, the clock gating logic can be observed from it.

A latch register, with one bit location kept as a flip-flop, is shown in Figure 38. Bit location 0 of each latch register is kept as a flip-flop and the input is taken straight from PWDATA[0], instead of latching it through a negative level active latch. The rest of the bit locations in the latch register are implemented with latches. The clock to the flip-flop needs to be taken from before the OR gate making the latches transparent during shift, so that shifting in values to the flip-flops is not disturbed. The clock gate now also needs to contain an OR gate to enable the clock to the flip-flops during shift. The latch register, of Figure 38, implemented in SystemVerilog, is shown in Figure 39.

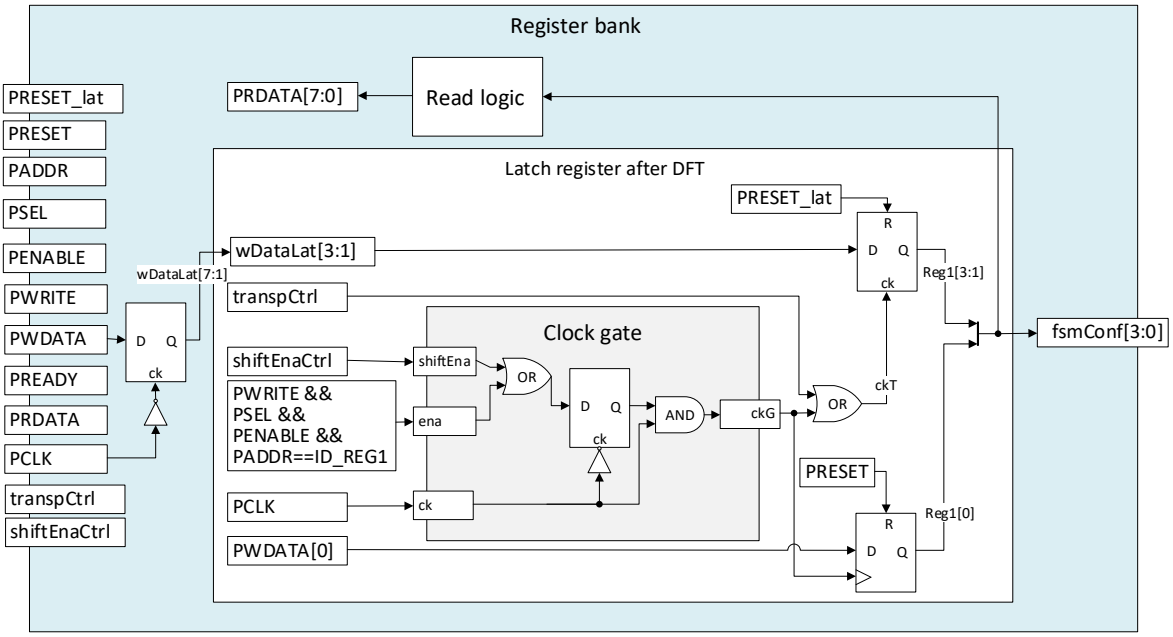

Figure 38. A read-write latch register after DFT measures.

```
// Clock gate enable
assign ena = (PWRITE && PSEL && PENABLE && PADDR == CONST_ID_REG1);
// Clock gate
ClockGate u_ClockGate(.ck(PCLK), .ena(ena), .shiftEna(shiftEnaCtrl), .ckG(ckG));

// Making latch transparent during shift
assign ckT = ckG | transpCtrl;

// Flip-flop register, first bit
always_ff @(posedge ckG or posedge PRESET) begin : la_ReadWrite_Register
  if (PRESET) begin
    Reg1_FF[0] <= CONST_RESETVALUE_REG1[0];
  end else begin
    Reg1_FF[0] <= PWDATA[0];
  end
end : la_ReadWrite_Register

// Latch register, other bits
always_latch begin : la_ReadWrite_LatchRegister
  if (PRESET_lat) begin
    Reg1_lat[3:1] <= CONST_RESETVALUE_REG1[3:1];
  end else begin
    if (ckT) begin
      Reg1_lat[3:1] <= wDataLat[3:1];
    end
  end
end : la_ReadWrite_LatchRegister

// Combining flip-flop and latch registers into one signal
assign fsmConfig[0]   = Reg1_FF[0];
assign fsmConfig[3:1] = Reg1_lat[3:1];
```

Figure 39. A read-write latch register after DFT measures coded in SystemVerilog.

4.5.3 Improving controllability

The added OR gates, making the latches transparent during shift, bring a problem of controllability. The port of the OR gate, that is connected to the shift enable, cannot be fully stuck-at tested since it is always low during capture. This results in large fault coverage loss as there is one of these OR gates for every latch register. The controllability can be increased by OR gating the shift enable with a control point, like in Figure 40. An AND gate is used to disable the control point when not in scan mode. Two flip-flops are used in the control point so that sequential control patterns can be created by the ATPG tool. The same control point is shared by all the OR gates, that make the register latches transparent, minimising the area overhead. The control point allows the OR gates to be fully stuck-at tested.

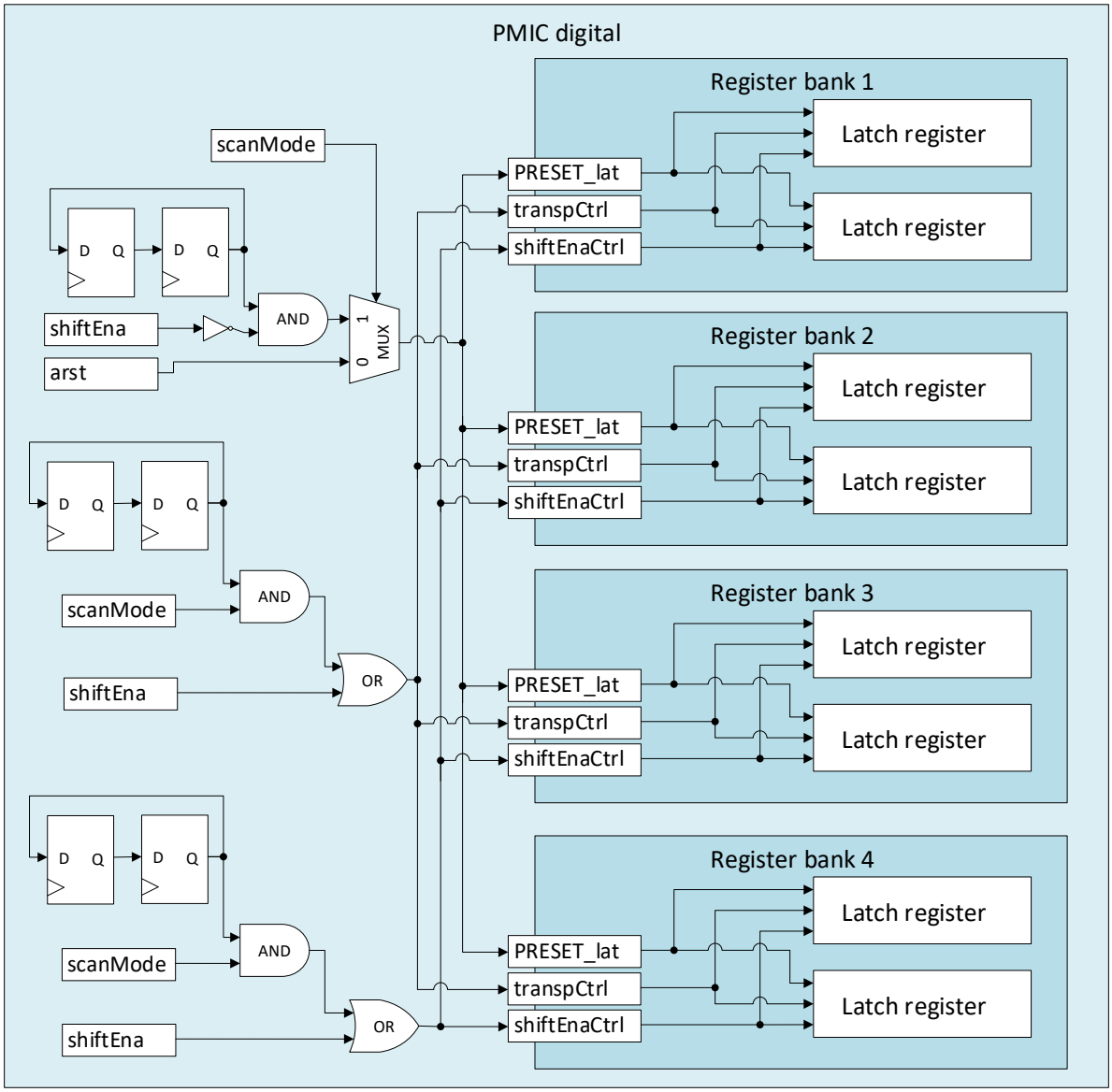

Figure 40. Increasing fault coverage by adding control points.

Since the address, PADDR, signal is used to create the enable signal to the clock gates of the latch registers, the clock to only one register can be on at a time, resulting in low controllability. The controllability can be increased by OR gating the shift enable with a control point, like in Figure 40. The control point is identical to the control point used for the OR gates making latches transparent. Two different control points are used to avoid masked faults, since the control points converge at the OR gate making the latches transparent. All the latch register clock gates share the same control point, making it possible for the ATPG tool to enable the clock for all the latch registers simultaneously, resulting in increased fault coverage.

The reset pin of scan flip-flops can be stuck-at tested by shifting in an opposite value to the reset value, pulsing the reset active and then shifting out and checking that the value inside the flip-flop was changed to the reset value [33]. The reset pin of latches cannot be tested this way since the latches are not scannable. The reset pin of latches can be made testable by connecting it to a control point, like in Figure 40. A multiplexer is used to select the control point in scan mode and the normal reset in normal mode. An AND gate is used to ensure that the reset is inactive during shift, and that the latches are kept transparent. Two flip-flops are used in the control point, so that sequential reset patterns can be created. The pattern for the reset is shifted to the scannable flip-flops and then executed during capture.

5 RESULTS

A reference design, implemented entirely with flip-flops, and a proposed latch register design, were synthesized. This chapter presents three types of results gathered from the synthesized designs. The area results of the latch register and reference designs are compared to see the area benefit of the latch replacement. Then, the power consumption results are compared to see the effect of the latch replacement on power consumption. Lastly, the ATPG results are compared to see the effect of the latch replacement on fault coverage and testing time.

5.1 Area results

To get area results, the latch register design, described in chapters 4.3 and 4.5, and the reference design, described in chapter 4.1, were synthesized. Design Compiler [29], made by Synopsys, was used as the synthesis tool.

All read-write and OTP registers that are 3-bits or wider, were implemented with latches in the latch register design. Read-write and OTP registers, that are 2 bits or smaller were implemented entirely with flip-flops. Both designs were synthesized with automatic clock gating, with the tool configured to insert clock gates to registers that are 3-bits or wider.

Area results, for the synthesized reference and latch register designs, are presented in Table 3. The total sequential area of the latch register design is decreased roughly 9% compared to the reference design. This results from 1220 flip-flops being replaced with latches in the latch register design. Some of the area gain is lost because of the 28 low active write data latches and the 6 control flip-flops that were added to the latch register design.

Table 3. Area results

	Reference	**Latch register**	**Latch register as % of reference**
Data latches	0	1220	
Flip-flops	5428	4214	
Clock gates	600	600	
Combinational area	408 969 um2	421 977 um2	103%
Sequential area	707 620 um2	640 810 um2	91%
Total area	1 116 588 um2	1 062 787 um2	95%
Gates in NAND2 equivalents	71 211	67 780	95%

The latch replacement also results in roughly 3% combinational area increase compared to the reference design. This mostly results from there being an extra OR gates in the clock tree for each latch register.

The increased combinational area is overshadowed by the decreased sequential area. The latch replacement ends up resulting in an overall area decrease of 5%, compared to the reference design.

The benefit of the latch registers is somewhat masked by all the area in the rest of the digital logic. The benefit is seen more clearly from the register bank area results, presented in Table 4.

The latch replacement results in roughly 15% register bank area decrease, compared to the flip-flop-based reference design.

Table 4. Register bank area results

	Reference	Latch register	Latch register as % of reference
Register bank 1 area	138 059 um2	118 726 um2	86%
Register bank 2 area	41 169 um2	34 499 um2	84%
Register bank 3 area	95 444 um2	81 799 um2	86%
Register bank 4 area	81 341 um2	66 113 um2	81%
Total register bank area	356 014 um2	301 137 um2	85%

5.2 Power consumption results

Power consumption is an important metric, along with area, that should be considered when assessing the overall benefit of the latch replacement. The synthesized gate level netlist and a switching activity information format (SAIF) file, generated from RTL simulations, were used to gather the power consumption results. The activity information was taken from a simulation, where register bank 1 was being written to and read from. This activity information was chosen, so that the effect of the latch replacement would be seen more clearly than from more realistic activity information, where the register banks are not accessed as often.

The power consumption results, for the reference and latch register designs, are presented in Table 5. The dynamic power consumption of the latch register design is roughly 1.6% larger than for the reference design. The static power consumption of the latch register design is roughly 0.7% smaller. Since the dynamic power consumption is much larger than the static power consumption, the total power consumption, for the latch register design, ends up being roughly 1.5% larger.

Table 5. Power consumption results

	Reference	Latch register	Latch register as % of reference
Dynamic power consumption	1.112 mW	1.130 mW	101.6%
Static power consumption	1.924 uW	1.910 uW	99.3%
Total power consumption	1.114 mW	1.131 mW	101.5%

The increased dynamic power consumption results from the switching power consumption of the clock pins of the latches being higher compared to flip-flops and from the OR gates that were added to make the register latches transparent.

5.3 ATPG results

Cost is an important metric, along with area and power consumption, that should be considered when assessing the benefit of the latch replacement. Reduced fault coverage will lead to more defective products being shipped to customers, increasing returns and costs. Increased testing time increases the cost of the testing itself.

Test patterns for stuck-at, IDDQ and transition faults were generated with Tessent [25], made by Siemens AG. The patterns, for both the reference and latch register designs, were generated using the same tool settings.

5.3.1 Stuck-at fault results

The stuck-at fault ATPG results, for the reference and latch register designs, are presented in Table 6. With the help of the DFT methods presented in chapter 4.5, a stuck-at fault coverage of 99.06% was achieved for the latch register design, while the reference design achieves a 99.09% stuck-at fault coverage. The stuck-at fault coverage of the 4 register banks is similarly close between the reference and latch register designs.

Table 6. Stuck-at fault results

	Reference	**Latch register**
Test coverage	99.13%	99.10%
Fault coverage	99.09%	99.06%
ATPG effectiveness	100.00%	99.97%
Test patterns	920	1803
Basic patterns	918	400
Sequential patterns	2	1403
Register bank 1 fault coverage	98.31%	98.22%
Register bank 2 fault coverage	99.84%	99.83%
Register bank 3 fault coverage	99.95%	99.95%
Register bank 4 fault coverage	99.94%	99.94%

The main drawback of the latch register design is the increased number of test patterns. As there are roughly 97% more test patterns for the latch register design, the testing time is increased.

5.3.2 IDDQ fault results

The IDDQ fault ATPG results, for the reference and latch register designs, are presented in Table 7. The latch register design has a worse IDDQ fault coverage, for the register banks. This results from the ATPG tool being unable to generate sequential patterns to test the inputs of the latches, reporting them as blocked.

Table 7. IDDQ fault results

	Reference	**Latch register**
Test coverage	98.65%	98.59%
Fault coverage	92.85%	91.33%
ATPG effectiveness	100.00%	100.00%
Test patterns	519	587
Basic patterns	505	565
Sequential patterns	14	22
Register bank 1 fault coverage	92.78%	88.08%
Register bank 2 fault coverage	90.70%	84.93%
Register bank 3 fault coverage	93.65%	88.56%
Register bank 4 fault coverage	90.83%	83.79%

The latch replacement does not affect the number of IDDQ test patterns as significantly as the number of stuck-at test patterns. The number of IDDQ test patterns is roughly 13% higher for the latch register design.

5.3.3 Transition fault results

The transition fault ATPG results, for the reference and latch register designs, are presented in Table 8. The overall fault coverage and the register bank fault coverage is higher for the latch register design. The higher fault coverage results from the DFT measures of chapter 4.5.3 improving controllability. The ATPG tool manages to report fault coverage for the reset pins of the latches, while the reset pins of the scan flip-flops are not fully covered. This results from the control point connected to the resets pins of the latches. The ATPG tool also manages to report fault coverage for faults in the scan enable pins of the manually inserted clock gating cells, also caused by the control point that was inserted in the latch register design.

Table 8. Transition fault results

	Reference	**Latch register**
Test coverage	84.19%	85.31%
Fault coverage	82.46%	83.54%
ATPG effectiveness	99.65%	99.78%

Test patterns	3000	3300
Basic patterns	9	9
Sequential patterns	2991	3291
Register bank 1 fault coverage	79.36%	83.09%
Register bank 2 fault coverage	83.72%	86.64%
Register bank 3 fault coverage	87.23%	90.60%
Register bank 4 fault coverage	83.87%	87.25%

The latch replacement also increases the testing time of transition faults. The number of transition test patterns is 10% larger for the latch register design.

6 DISCUSSION

The main goal for the thesis was to decrease area of digital logic in a PMIC. The goal was reached with a total area decrease of 5% and a register bank area decrease of 15%, compared to the reference PMIC. The 5% area decrease is significant as it can already result in large profits when millions of ICs are manufactured.

The area benefit of the latch replacement is highly PMIC dependent. It is determined by the flip-flop area in read-write and OTP registers, compared to the rest of the digital area. Designs with more registers benefit more from this method. The average width of the read-write and OTP registers also effect the area benefit. As future work, studying how to fully populate all registers to be 8-bit wide could be considered to maximise the area benefit of the replacement.

Read-write and OTP registers are the most common source of flip-flops in the reference design, and they were also found the best suited for latch replacement due to their structure. This already resulted in roughly 25% of all the flip-flops to be replaced with latches. As future work, determining other suitable flip-flops to be replaced, and replacing them could be considered.

Although the latch replacement method was developed with PMICs particularly in mind, it is suited for all ICs containing flip-flop-based register banks. This is significant as Nordic Semiconductor also makes Bluetooth, Wi-Fi and cellular IoT ICs that most likely could use the latch replacement method. As future work, finding out which products at Nordic Semiconductor are suited for the latch replacement method could be considered.

The second goal of not significantly increasing the power consumption was also met, as the total power consumption increased by only 1.5%. The effect of the latch replacement on power consumption was contradictory, as the dynamic power consumption was increased by 1.6%, but the static power consumption was decreased by 0.7%. The results were taken from activity, where a register bank was continuously written to and read from. In a more realistic situation, the PMIC might be configured at start-up, but then the configuration registers would probably be seldomly accessed and the clock would be off most of the time. Because of that, the decreased static power consumption is significant.

The increase in the dynamic power consumption resulted from the OR gates, that were added to the clock three, and from the switching power consumption of the clock pins of latches unexpectedly being higher than of flip-flops according to the modelled library data. As future work, the reason for this unexpected switching power consumption could be researched, so that the power consumption could be decreased.

The third goal of maintaining the quality of manufacturing testing was also met. As the stuck-at fault model is a widely used fault model, and stuck-at test patterns have been shown to detect many of the faults based on other models, the most effort in this thesis was spent on improving the stuck-at fault coverage of the latch register design. The innovation of leaving one bit of each register as a flip-flop was significant in improving fault coverage, without increasing the silicon area significantly. The stuck-at fault coverage ended up being roughly the same compared to the reference design. The drawback is that the number of test patterns needed to achieve this coverage was roughly doubled.

Due to the reduced observability of the faults in the latch registers, most of the faults in the rest of the circuit are detected in the first test patterns and the later test patterns only detect a comparatively small number of faults in the latch registers. This means that the testing time could be significantly decreased by sacrificing some fault coverage in the latch registers. As future work, the optimal number of test patterns could be studied to optimize the costs from increased returns with the costs from decreased testing time.

The quality of the transition testing was improved, as the fault coverage was increased. The improvement caused by the improved controllability of the scan enable pins of the clock gates is deceptive and can be disregarded, as they are only used in scan testing and do not affect the normal operation of the circuit. The increased fault coverage for the reset pins of the latches compared to the reset pins of flip-flops is more significant. A slight downside is that the number of test patterns was increased by 10%.

The quality of the IDDQ testing was not completely maintained, as the fault coverage was decreased. The testing time was also slightly increased, with a 13% increase in the number of test pattern. The fault coverage decrease is not very significant, as the number of dedicated IDDQ test patterns is small compared to total number of patterns. Monitoring the quiescent supply current during stuck-at testing already gives quite good IDDQ fault coverage. As future work, the IDDQ fault coverage of the latch register design could be also improved closer to that of the reference design.

The LSSD scan style, covered in chapter 3.1.3, is the most well-known method of dealing with latches. The main downside with it is that a dummy latch needs to be added for each latch in the design. Another downside is that it requires 2 extra clock signals to be routed. The partial scan method developed in this thesis is much more area efficient, only having the first bit of each latch register scannable, only using one OR gate per latch register and having less routing complexity.

The most similar partial scan method, found during researching the theory part of the thesis, was the method of modelling the latches as combinational gates and preloading them with known values, of [12], presented in chapter 3.1.4. The method requires adding multiplexers to the data inputs of the latches and OR gating a force write signal to the clock inputs of the latches. Since the latches are also modelled as combinational gates, it requires modifying the gate level netlists. The partial scan method developed in this thesis is much more area efficient, only using a single OR gate per latch register to bring the latches into known states. The improved observability from the innovation of leaving one bit as a scannable flip-flop and the added control points most likely also results in much larger fault coverage. Another upside of the partial scan method developed in this thesis is that it is well supported by the existing design automation tools, not requiring any gate-level netlists to be manually modified.

7 SUMMARY

The main purpose of this thesis was to decrease the area of a flip-flop-based reference PMIC by replacing selected flip-flops with latches. The secondary goals for the thesis were to not increase the power consumption and not to decrease the manufacturing fault coverage, caused by the latch replacement method.

The thesis first provided some necessary background theory on the differences of latches and flip-flops. Latches are transparent on clock active level, whereas flip-flops are transparent only on clock active edge. The transparent nature of latches makes them ill-suited for shift registers and so ill-suited for the most common manufacturing testing method of scan testing. Pulsed latches and pulsed flip-flops behave similarly to flip-flops if the clock pulse is short enough. A disadvantage is the increased power consumption caused by the pulser.

The thesis then provided some necessary background theory on design for testability methods. Scan testing improves the controllability and observability of a design by replacing flip-flops in a design with scan flip-flops and connecting into scan chains. Partial scan reduces the area overhead of the scan cells by leaving some sequential elements as non-scannable. A disadvantage is that the fault coverage can decrease and the testing time increase, due to the reduced controllability and observability. BIST schemes incorporate testing features on the CUT itself. A disadvantage of the BIST methods is the area overhead of the BIST logic.

As the average register width of the reference PMIC was small, and the power consumption overhead of the pulser in a pulsed latch design is dependent on how many latches can share a pulser, it was decided to investigate decreasing the area with latches instead. Flip-flops in read-write and OTP registers, of the reference PMIC, were identified as suitable to be replaced with latches.

The latch register method developed in this thesis saves area, by sharing the primary latches for all the latch registers. The write data is loaded to shared negative level active primary latches on the negative level of the clock. The primary latch outputs are connected to several secondary latches. Clock gating is used to determine which latch register, acting as the secondary latches, the data is loaded to on the positive level of the clock.

The area overhead of the traditionally used DFT methods for testing latches, such as memory and logic BIST and LSSD scan style were determined to be suboptimal due to their large area overhead. Partial scan was selected as the DFT method instead, as it was estimated that the fault coverage could be maintained with the sequential depth only being 2.

The register latches were made transparent during the shift operation of scan testing with OR gates. This allowed the latches to be set into known states during shift and then sequentially tested during capture. Observability of clock gating logic was improved by keeping the first bit of each latch register as flip-flop, allowing the clock gating logic to be observed from the flip-flop directly. A sequential multiplexer control point was added to the asynchronous reset pins of the latches, to be able to stuck-at test them. A sequential OR control point was added to the shift enable pins of the latch clock gates, to improve the controllability of the clock gates. Another sequential OR control point was added to the pins of the OR gates, making the latches transparent, to make them stuck-at testable.

The latch replacement resulted in 5% total area decrease and 15% register bank area decrease. The price of the area decrease is a slight increase in power consumption. The latch replacement method manages to maintain stuck-at fault coverage, but results in roughly twice as many test patterns needed to achieve that coverage. Another disadvantage of the replacement is some IDDQ fault coverage being lost.

8 REFERENCES

[1] Harris, D. M. & Harris, S. L. (2007). Digital design and computer architecture. Morgan Kaufmann.

[2] Holdsworth, B. & Woods, R. C. (2002). Digital logic design (4th ed.). Newnes.

[3] Sinclair, I. R., & Dunton, J. (2007). Practical Electronics Handbook: Vol. 6th ed. Newnes.

[4] Mano, M. M., & Ciletti, M. (2013). Digital design: with an introduction to the Verilog HDL. Pearson.

[5] Harris, D. M. (2001). Skew-tolerant Circuit Design. Morgan Kaufmann.

[6] Wai Kai Chen. (2005). The Electrical Engineering Handbook. Academic Press.

[7] Laung-Terng W., Cheng-Wen W., & Xiaoqing W. (2006). VLSI Test Principles and Architectures : Design for Testability. Morgan Kaufmann.

[8] Williams, M.J., & Angell, J.B. (1973). Enhancing Testability of Large-Scale Integrated Circuits via Test Points and Additional Logic. In: IEEE Transactions on Computers, vol. C-22, no. 1, pp. 46-60, Jan. 1973, doi: 10.1109/T-C.1973.223600.

[9] Eichelberger, E.B., & Williams, T.W. (1977). A logic design structure for LSI testability. In: Proceedings of the 14th Design Automation Conference (DAC '77). IEEE Press, 462–468, doi: 10.1145/62882.62924.

[10] TestMAX DFT User Guide, version S-2021.06-SP5. Available: https://spdocs.synopsys.com/dow_retrieve/latest/home_public/tm_dft.html

[11] Beest, F.T., Peeters, A.M., Kerkhoff, H.G., & Berkel, K.V. (2002). Scan Chain Optimization for Asynchronous Circuits. In: Proceedings Prorisc workshop (pp. 288-294).

[12] Hui, M. M. Y., & Nadeau-Dostie, B. (1992). Scan testing of latch arrays. In: Digest of Papers. 1992 IEEE VLSI Test Symposium, Atlantic City, NJ, USA, 1992, pp. 31-36, doi: 10.1109/VTEST.1992.232720.

[13] Sheth, A. M., & Savir, J. (2003). Single-clock, single-latch, scan design. In: IEEE Transactions on Instrumentation and Measurement, vol. 52, no. 5, pp. 1455-1457, Oct. 2003, doi: 10.1109/TIM.2003.818550.

[14] Tsukisaka, M., Imai, M., & Nanya, T. (2004). Asynchronous scan-latch controller for low area overhead DFT. In: IEEE International Conference on Computer Design: VLSI in Computers and Processors, 2004. ICCD 2004. Proceedings., pp. 66-71, doi: 10.1109/ICCD.2004.1347901.

[15] Bushnell, M. L. & Agrawal, V. D. (2000). Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits. Kluwer Academic.

[16] Chickermane, V., Richter, S., & Barnhart, C. (2000). Integrating Logic BIST in VLSI Designs with Embedded Memories. In: Proceedings 18th IEEE VLSI Test Symposium, 2000, pp. 157-164, doi: 10.1109/VTEST.2000.843840.

[17] Van de Goor, A. J., & Tlili, I. B. (1998). March tests for word-oriented memories. In: Proceedings Design, Automation and Test in Europe, pp. 501-508, doi: 10.1109/DATE.1998.655905.

[18] Al-Harbi, S. M., & Gupta, S. K. (2001). An efficient methodology for generating optimal and uniform march tests. In: Proceedings 19th IEEE VLSI Test Symposium. VTS 2001, 2001, pp. 231-237, doi: 10.1109/VTS.2001.923444.

[19] Zarrineh, K., & Upadhyaya, S. J. (1999). On programmable memory built-in self test architectures. In: Design, Automation and Test in Europe Conference and Exhibition,

1999. Proceedings (Cat. No. PR00078), 1999, pp. 708-713, doi: 10.1109/DATE.1999.761207.

[20] Boutobza, S., Nicolaidis, M., Lamara, K. M., & Costa, A. (2005). Programmable memory BIST. In IEEE International Conference on Test, 2005, pp. 10 pp.-1164, doi: 10.1109/TEST.2005.1584083.

[21] Rajsuman, R. (1999). Testing a system-on-a-chip with embedded microprocessor. In: International Test Conference 1999. Proceedings (IEEE Cat. No. 99CH37034), 1999, pp. 499-508, doi: 10.1109/TEST.1999.805773.

[22] Tsai, C. H., & Wu, C. W. (2001). Processor-programmable memory BIST for bus-connected embedded memories. In: Proceedings of the 2001 Asia and South Pacific Design Automation Conference, 2001, pp. 325-330, doi: 10.1109/ASPDAC.2001.913327.

[23] Bodoni, M. L., Benso, A., Chiusano, S., Di Carlo, S., Di Natale, G., & Prinetto, P. (2000). An effective distributed BIST architecture for RAMs. In: Proceedings IEEE European Test Workshop, 2000, pp. 119-124, doi: 10.1109/ETW.2000.873788.

[24] Miyazaki, M., Yoneda, T., & Fujiwara, H. (2006). A memory grouping method for sharing memory BIST logic. In Proceedings of the 2006 Asia and South Pacific Design Automation Conference, 2006, pp. 671-676, doi: 10.1109/ASPDAC.2006.1594763.

[25] Tessent Scan and ATPG User's Manual, Software version 2021.4, document revision 23. Available: https://support.sw.siemens.com/en-US/product/852852118

[26] Mitić, M., & Stojčev, M. (2006). An overview of on-chip buses. In: Facta universitatis-series: Electronics and Energetics, 19(3), 405-428.

[27] AMBA Specification (Rev 2.0).

[28] AMBA APB Protocol Secification v2.0.

[29] Design Compiler User Guide, version T-2022.03. Available: https://spdocs.synopsys.com/dow_retrieve/latest/home_public/synthesis.html

[30] Paik, S., & Shin, Y. (2010). Pulsed-latch circuits to push the envelope of ASIC design. In: International SoC Design Conference, 2010, pp. 150-153, doi: 10.1109/SOCDC.2010.5682949.

[31] Bernard, S. (2014). Robust and energy-efficient explicit pulse-triggered flip-flops in 28nm FDSOI technology for ultra-wide voltage range and ultra-low power circuits.

[32] Chang, C. L., Jiang, I. H. R., Yang, Y. M., Tsai, E. Y. W., & Chen, A. S. H. (2012). Novel pulsed-latch replacement based on time borrowing and spiral clustering. In: Proceedings of the 2012 ACM international symposium on International Symposium on Physical Design, March 2012, pp. 121-128, doi: 10.1145/2160916.2160944.

[33] Why some patterns do not have capture clock pulse? (Accessed April 21, 2022). Siemens AG, URL: https://support.sw.siemens.com/knowledge-base/MG57862