TRABALHO DE GRADUAÇÃO

# COMPARISON BETWEEN THE IMPLEMENTATIONS OF A BCH DVB-S2X DECODER IN FPGA AND IN ASIC

**Thiago Queiroz Holanda**

**Brasília, dezembro de 2019**

ENGENHARIA
MECATRÔNICA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

# COMPARISON BETWEEN THE IMPLEMENTATIONS OF A BCH DVB-S2X DECODER IN FPGA AND IN ASIC

**Thiago Queiroz Holanda**

*Relatório submetido como requisito parcial de obtenção*
*de grau de Engenheiro de Controle e Automação*

Banca Examinadora

Prof. José Camargo da Costa, ENE/UnB _____
*Orientador*

Prof. José Edil Guimarães de Medeiros, ENE/UnB _____

Eng. Ana Ravena Alcântara da Costa _____

**Brasília, dezembro de 2019**

**FICHA CATALOGRÁFICA**


| |
|---|
| THIAGO, QUEIROZ HOLANDA<br>COMPARISON BETWEEN THE IMPLEMENTATIONS OF A BCH DVB-S2X DECODER IN FPGA AND IN ASIC,<br>[Distrito Federal] 2019.<br>x, 101p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2019). Trabalho de Graduação – Universidade de Brasília.Faculdade de Tecnologia.<br>1. BCH 2.ASIC<br>3. Comunicação Satelital<br>I. Mecatrônica/FT/UnB II. Título (Série) |


**REFERÊNCIA BIBLIOGRÁFICA**

HOLANDA, THIAGO, (2019). Comparison between the implementations of a BCH DVB-S2X decoder in FPGA and in ASIC. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-$n°$022, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 101p.

**CESSÃO DE DIREITOS**

AUTOR: Thiago Queiroz Holanda

TÍTULO DO TRABALHO DE GRADUAÇÃO: Comparison between the implementations of a BCH DVB-S2X decoder in FPGA and in ASIC.

GRAU: Engenheiro ANO: 2019



---


Thiago Queiroz Holanda

SQN 206, Bloco H, Apt 602

70844-080 Brasília – DF – Brasil.

## Aknoledgements

*First I need to thank my mother Elizabeth Queiroz who raised me into the person I am today. She taught me meaning of hardwork, dedication and perseverance, showed me that even in the toughest times there is a way and gave me every opportunity to grow. Then I need to thank my brother Matheus Queiroz Holanda who aided me whenever I was in need, who was an incredible role model and an extremely caring brother. Without the both of them I would not be the person I am today. I must also thank my father Adriano Holanda, my grandfathers Edelmo Queiroz and Nilson Holanda for helping my mother to give me these opportunities. I must thank my grandmothers Silvia, Iolanda and my aunt Elisângela for all the love and support that they have given me.*

*I must thank my advisor José Camargo da Costa for giving me such an amazing opportunity to learn and develop during this project. I must also thank my other advisor Guilherme Shimabuko who took the time to teach me everything I needed, who helped me through all the project, who advised me through every step. Without his help I could not have gotten so far and accomplished so much I am incredibly grateful for everything. I must also thank everyone in the LPCI, Laboratório de Projeto de Circuitos Integrados, for all the help and support they gave me during this period especially Rafael Lima who helped me learn and develop more in this area and many others.*

*Now I must thank my girlfriend Sarah Maria de Albuquerque Sousa who has been with me even before I started my graduation, who helped me through every step of the way, stood by me and supported me whenever I needed. You are the most important person in my life and I am incredibly lucky to have found someone so amazing as you to call my partner. I must also thank my incredible friends who helped me through so many things, Danilo Li, Jorge Simeão, Ricardo Bauchspiess, Daniel Ervilha and Ada Carine you are all my family and I love you very much.*

*Last but not least I must thank DROID for teaching me how to work, giving me the opportunity to learn and grow. I must thank Alexandre Crepory for accepting me into this incredible team, teaching me so much about development and leadership and being an amazing friend. I must also thank Rodrigo Werberich and Leticia Porto you are both incredible people, incredible role models, incredible friends and have shown me that through dedication and hard work I can accomplish so much. Another person I need to thank is Sara Gomes Cardoso, you are incredible and managed to make my experience in this team even more amazing. Artur, Camila, João, Gabriel, Abdullah, Lucas S, Rafael, Carlos, Daniel Bauchspiess, Rodrigo, Vanessa, Rebeca, Rosana, Isabella, Bianca, Thamires you were all very important for my development and growth.*

*Thiago Queiroz Holanda*

## RESUMO

Comunicação satelital é uma parte muito importante no sistema de comunicação global, portanto padrões que garantam seu funcionamento devem ser aplicados por meio de especificação de comportamentos e algoritmos. Entre esses padrões está o *Digital Video Broadcasting - Satellite Second Generation Extended*, DVB-S2X, uma evolução do *Digital Video Broadcasting - Satellite*, DVB-S. Nesse padrão, para corrigir as mensagens recebidas que podem sofrer erros durante a transmissão, são utilizados códigos de correção de erro tais como o *Low Density Parity Check*, LDPC, e o Bose-Chauduri-Hoquenghem, BCH. A implementação desses algoritmos em um sistema digital é realizada através de uma plataforma apropriada. Para isso podem ser utilizados *Field Programmable Gate Arrays*, FPGAs, ou um sistema *Application Specific Integrated Circuit*, ASIC. Ao utilizar uma implementação ASIC, ferramentas *Electronic Design Automation*, EDA, e *Computer Aided Design*, CAD, são usadas para poder auxiliar o desenvolvimento. Neste trabalho, tomando como ponto de partida uma implementação em FPGA de um decodificador BCH previamente realizada, foi feita uma adaptação para fluxo de projeto de ASIC digital e sua validação por simulação com emprego do framework do Cadence Design Systems. Para isso utilizou uma especificação de 100MHz de clock buscando consumo menor do que a implementação da FPGA. Para auxiliar o desenvolvimento foi criado um script em Bash para automatizar a síntese lógica de várias tecnologias e configurações de metais. A comparação de consumo de potência, temporização e área foi feita utilizando os resultados da síntese lógica nas tecnologias de 180nm da XFAB. A partir dos dados analisados foi encontrada uma implementação em ASIC com uma frequência máxima de clock de 490MHz com um consumo abaixo de 960 mW ocupando uma área de aproximadamente 101 $mm^2$. O modelo em FPGA apresentou uma frequência nominal de 100MHz com um consumo de 620 mW. O modelo em ASIC para a mesma frequência teve um consumo de 175 mW. Comparando os dois modelos conclui-se que uma implementação em ASIC pode levar a ganhos consideravéis no consumo e em desempenho. Para a especificação todas as soluções em ASIC tiveram um consumo menor e todas as tecnologias permitem aumentar o desempenho total do sistema.



Palavras Chave: BCH, ASIC, Comunicação Satelital

## ABSTRACT

Satellite communication is a very important part of the global communication system. Therefore, standards that ensure its operation must be applied through the specification of behaviors and algorithms. Among these standards there is the Digital Video Broadcasting - Satellite Second Generation Extended, DVB-S2X, an evolution of the Digital Video Broadcasting - Satellite, DVB-S. Within this standard, to correct the received messages that can suffer errors during transmission, error correction codes such as the Low Density Power Converter, LDPC, and the Bose-Chauduri-Hoquenghem, BCH, are utilized. The implementation of these algorithms in a digital system is performed through an adequate platform. For that, Field Programmable Gate Arrays, FPGAs, can be utilized, or an Application Specific Integrated Circuit, ASIC, can be created. In the ASIC implementation, Eletronic Design Automation, EDA, tools and Computer Aided Design, CAD, are used to facilitate the development. In this work, utilizing a previously made implementation in FPGA of a BCH decoder, an adaptation for the ASIC digital design flow and itsvalidation through simulation with the use of the Cadence Design System framework. The comparison between power consumption, timing and area were made using the results of the logical synthesis in XFAB technologies. From the analysed data, an ASIC implementation with maximum frequency of 490MHz with a power consumption below 960 mW occupying an area of approximately 101 $mm^2$ was found. The FPGA model presented a nominal frequency of 100MHz and power consumption of 620 mW. The ASIC model for the same frequency presented power consumption of 175 mW. Comparing these two models, it can be seen that the ASIC implementation can lead to considerable gains in power consumption and performance.



Keywords: BCH, ASIC, Satellite Communication

# SUMMARY

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF SYMBOLS

### Acronyms

| | |
|---|---|
| ARIB | *Association of Radio Industries and Business* |
| ASIC | *Application Specific Integrated Circuits* |
| BCH | *Bose-Chaudhuri-Hocquenghem codes* |
| CAD | *Computer Aided Design* |
| CD | *Compact Disk* |
| CMOS | *Complementary Metal Oxide Semiconductor* |
| DTH | *Direct to Home* |
| DVB | *Digital Video Broadcasting* |
| DVB-S | *Digital Video Broadcasting - Satellite* |
| DVB-S2X | *Digital Video Broadcasting - Satellite Second Generation Extension* |
| DVB-S2 | *Digital Video Broadcasting - Satellite Second Generation* |
| EDA | *Electronic Design Automation* |
| ESL | *Eletronic System Level* |
| ETSI | *European Telecomunications Standards Institute* |
| FEC | *Forward Error Correction* |
| FIFO | *First In First Out* |
| FPGA | *Field Programmable Gate Array* |
| FSM | *Finite State Machine* |
| HAL | *HDL Analysis and Lynt* |
| HDL | *Hardware Description Language* |
| HV | *High Voltage* |
| IC | *Integrated Circuit* |
| IP | *Intelectual Property* |
| ISDB-S | *Integrated Services Digital Broadcasting - Satellite* |
| ISDB-S3 | *Integrated Services Digital Broadcasting - Satellite Version 3* |
| LDPC | *Low Density Parity Check* |
| LEF | *Library Exchange Format* |
| LPMOS | *Low Power MOS* |
| MHz | *Mega Hertz* |
| MOSLP | *MOS Low Power* |
| MOSST | *MOS Standard* |
| MPW | *Multi-Project Wafer* |
| mW | *MilliWatt* |
| NCVHDL | |
| RS | *Reed-Solomon* |
| RTL | *Register Transfer Level* |
| SDC | *Synopsys Design Constraints* |

# Chapter 1

# Introduction

*With the widespread use of wireless communication, the need for reliable sources of message correcting becomes more present. With these needs, many standards are established to guarantee the proper communication between various technologies.*

## 1.1 Contextualization

Satellite communication is an incredibly important part of the current communication system. This form of communication has two basic steps: the uplink of information and the downlink of the message. Both of these steps are susceptible to errors during the transmission. To ensure that the information is correctly sent and received, procedures and standards are put in place to guarantee the correct performance. One of the most recent additions to the standards of satellite communication is the Digital Video Broadcasting - Second Generation Extended (DVB-S2X).

Given that this standard is an improvement from past implementations, it is essential to create hardware able to accomplish all the demanding constraints. Therefore, each individual part demands a focused approach. For the downlink, decoding of the received message demands heavy processing algorithms like the Low Density Parity Check (LDPC) and the Bause-Chauduri-Hoquenghem (BCH).

An efficient way to use these algorithms is through the implementation of a digital system. For this, there are two different possibilities: the use of Field Programmable Gate Arrays (FPGA) or Application Specific Integrated Circuits (ASIC). To determine which of these is the implementation with the overall lowest power consumption and higher performance capabilities, a comparison between them must be made.

The current work focuses on the digital implementation with ASIC technology of a BCH decoder based on DVB-S2X. This is based on an implementation of the BCH decoder system in Very-High Speed Integrated Circuits Hardware Description Language (VHDL) with a focus on FPGA developed in another work. With this implementation, a clock frequency of 100MHz is specified with a processing capability of 5 megasymbols/second. With this specification it is inter-

esting to test different implementations to find the best alternatives regarding power consumption, cost and timing.

## 1.2 Objectives

This work seeks to adapt the BCH codes, previously implemented for FPGA, in ASIC technology to pass the Hardware Description Language Analysis and Lint. After this it is necessary to synthesize the codes to determine resource estimation and compare the results between different technologies and FPGA implementation.

# Chapter 2

# Literature Review

.

## 2.1 Satellite Communication Standards

To guarantee that satellite communications will behave correctly, many standards need to be put in place. To create these, Standard Development Organization (SDO) are necessary. Each region has their own SDO, as seen with the Association of Radio Industries and Business (ARIB) in Japan and with the European Telecommunications Standards Institute (ETSI) in Europe.

Among the many standards created by these SDOs, some of the most notable ones for satellite communication are the Integrated Services Digital Broadcasting - Satellite (ISDB-S) regulated by ARIB and the Digital Video Broadcasting - Satellite (DVB-S) regulated by ETSI.

The ISDB-S is a system that consists of a source coding section that converts video, audio and data signals into digital signals; a multiplexing section that multiplexes the digital signals; a conditional access section that scrambles the signals and distributes the unscrambling keys to subscribers; and a channel coding section that performs signal processing. [8] The overview of the channel coding can be viewed in Figure 2.1.

Originally for error correction, the ISDB-S utilized Reed-Solomon codes for outer coding correction [8]. In the current implementation, ISDB-S3, LDPC is utilized for inner coding and the BCH for outer coding. The BCH is capable of correcting up to 12 errors. As for the LDPC, the code length is of 44880 bits with 11 different code rates. [9]

The DVB-S became the most popular system for digital satellite television delivery in the late 1990's to the early 2000's. Over time, the system was replaced with the Digital Video Broadcasting - 2nd Generation Satellite (DVB-S2), that brought gains of around 30 percent to the physical layer for Direct To Home (DTH) transmission. The DVB-S2 system allows for quasi-error free operation at 0,6 to 1,2 dB from the modulation constrained Shannon limit. This is achieved through a system that utilizes LDPC codes, a large number of decoding iterations and a concatenated BCH outer code. [10]

The standard Digital Video Broadcasting - Satellite Second Generation Extension is a specification that provides additional technologies and features. It is based on the DVB-S2 and utilizes

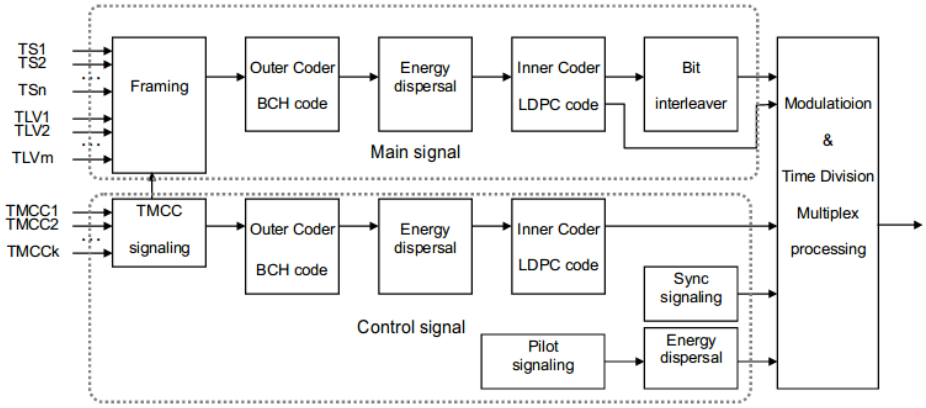


Figure 2.1: Basic Structure of Channel Coding of ISDB-S [1]

the same architectural system while adding finer modulation and coding to provide additional elements such as a finer gradation and extension of the number of modulation and coding modes.[11] An overview of the decoding process can be viewed in Figure 2.2. This image shows the path the symbols take until they are corrected. It can be seen that many different steps are necessary and that the LDPC and BCH codes are necessary. The BCH block is highlighted since it is the subject of this work.

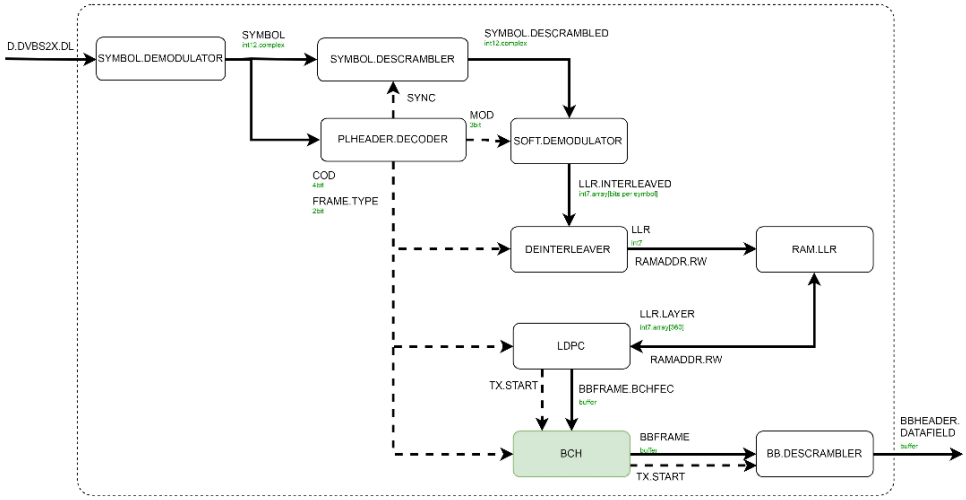


Figure 2.2: Overview of a DVB-S2X.

## 2.2 Error Correction Codes

Satellite communications are susceptible to errors, the change in value of information, during transmission. An error occurs when a bit of information becomes altered assuming the value of 1 when it is originally 0 or assuming 0 when it is originally 1 by an external factor, leading to misinformation. Errors can be classified into single bit errors and burst errors. Burst errors are

a number of bits in a sequence that suffer errors while single bit errors are individual errors. To guarantee the correct information, it is necessary to utilize algorithms to correct such errors.

Error correction codes are the means whereby errors can be corrected based on the data received. Error detection codes are the means whereby errors can be detected based on the data received. In conjunction, it leads to error control coding. The use of error control coding can provide the difference between an operating system and a dysfunctional one [12]. The most commonly used cyclic error correction codes are the BCH and the Reed-Solomon (RS) codes. These codes are related and their decoding algorithms are similar. [12]

The BCH codes have a few important properties that make them desirable for digital implementation. One important property is the fact that "For any positive integers $m$ and $t$, there is a code in this class that consists of blocks of length $2^m - 1$, that corrects $t$ errors, and that requires no more than $mt$ parity check digits." This means that, for any type of message and number of errors, it is possible to use BCH codes to correct the errors [13]. As seen in Figure 2.3 for the BCH algorithm a frame is received and with this frame syndromes are calculated. The syndromes are the remainder of the division of the received message by the generator polynomial. The syndromes then are used in the Berlekamp-Massey algorithm to determine the error locator polynomial. To find the error locations, the Chien search is used and with the locations determined the errors are corrected.

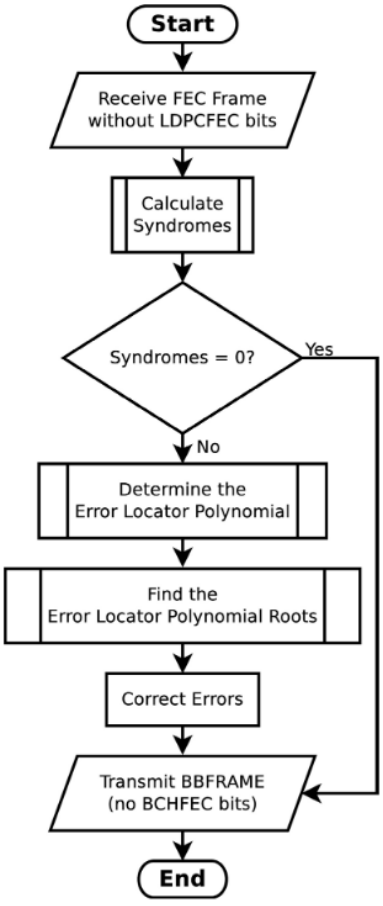


Figure 2.3: BCH Decoder Flowchart [2]

Reed-Solomon codes are also constructed and decoded through the use of finite fields. These codes can be constructed through three different ways: the original approach through primitive elements of finite fields, the generator polynomial approach and the Galois field Fourier transform approach. The behavior of a Reed-Solomon decoder can be seen in Figure 2.4. Some of the algorithms used in this process are the same as the ones used in BCH decoding, like the Chien search and the syndrome calculations. [14]

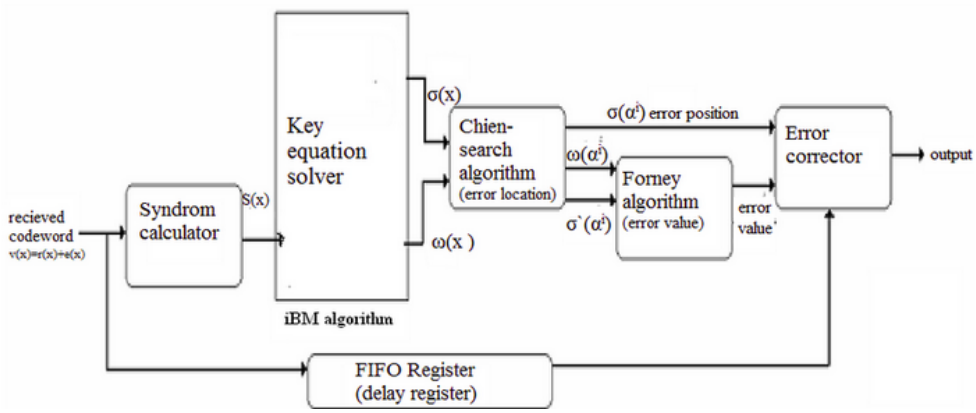


Figure 2.4: Reed-Solomon Decoder Flowchart [3]

LDPC codes have great performance due to the fact that they possess iterative decoding algorithms that are parallelizable in hardware while also being easy to implement. Also, LDPC codes of almost any rate and blocklength can be created simply by specifying the shape of the parity check matrix. [12]

## 2.3 IC Digital Design Flow

With complex systems, it is necessary to create dedicated hardware to accomplish specified demands. To aid and simplify these tasks, the design changed to resemble coding with the help of Hardware Description Languages. The use of these languages considerably reduces the necessary time to create a system. One of the most utilized languages is the Very-High Speed Integrated Circuit (VHSIC) Hardware Description Language, commonly referred to as VHDL.

For the digital design, it is necessary to specify the final platform to determine all the necessary workflow and methodology. One of the most common forms of hardware implementation is through the use of Field Programmable Gate Arrays, FPGA, in which there is a matrix of configurable logic blocks connected via programmable interconnects. FPGAs can be reprogrammed to accomplish different desired application or functionality requirements after manufacturing [15]. The other form of hardware implementation is with Application Specific Integrated Circuit, ASIC. ASIC projects are designed for a single purpose and cannot be repurposed after manufactured.

ASIC implementations are incredibly complex and time consuming. To implement such complex systems, the use of powerful design programs such as Electronic Design Automation (EDA) and Computer Aided Design (CAD) tools are essential. These tools end up being extremely

complex with elaborate workflows necessary to guarantee the final results. [16]

Each of these applications have different purposes, as well as advantages and disadvantages in their implementation. A design with FPGA can take a few weeks, while an ASIC design will take several months to accomplish all the necessary steps. Therefore, for fast prototyping, FPGAs are the recommended platform. With more demanding timing and power consumption applications, ASIC is regularly used due to the full control over the whole design, allowing for a more optimized final result. The last important factor is the monetary cost between both designs. For small scale designs, FPGAs are the desired design implementation since the cost is equivalent to the number of FPGAs being used in the design. The cost for ASIC applications is based on the cost of manufacturing a batch of chips. Therefore, large scale production is necessary to reduce the cost of each individual chip. This means that, if it is necessary to create a large scale production of an application, the FPGA implementation would have its cost increased for each product, while with ASIC, the cost would be the same whether ten thousand or twenty thousand products are manufactured.

After defining the platform for implementation, it is necessary to define the design methodology. There are many different ways for a digital design project to be implemented, with the top-down and the bottom-up being the most commonly used. In bottom-up design, the designer starts by building the basic gates for the technology being used. After these are built, they build basic units (such as adders, flip-flops) with the gates. With the basic units, they build generic modules that will be used in the project. With all of these components, the designer puts together all the modules and checks to see if the resulting architecture is reasonable. This methodology is usually used only in full custom designs with the use of no external Intellectual Properties (IP), and is considered an old fashioned design methodology.

In contrast, the top-down design methodology starts by choosing the algorithm to be used and implementing it. After the algorithm is implemented, the designer starts making the architecture and optimizing it. This leads to a guaranteed desirable final architecture, given that the designer has complete control of how it is going to turn out, while on the bottom-up, the architecture depends on the modules previously created. After the architecture is defined, the designer starts determining the functional modules and the design hierarchy. With these, the project is divided into smaller blocks until the specification is met. When this is complete with the aid of EDA and CAD tools, such as Genus and Innovus, the blocks are verified and the logical synthesis is made. In the logical synthesis, the software determines the gates that will be used in the blocks created. When this step is done, the layout is created from the physical synthesis [17].

## 2.4 IC EDA Tools

With the constant rise of hardware development complexity, there is an ever increasing need to utilize EDA and CAD tools to aid in the process [16]. Given an electronic system modeled at the electronic system level (ESL), an EDA automates the design and test processes of verifying the correctness of the design against the specifications. The EDA will take the system through various

synthesis and verification steps, finally testing the manufactured electronic system to ensure that it meets the specifications and quality requirements [18].

EDA comprises of hardware and software co-design, synthesis, verification, and test tools that check the ESL design, translate to the register transfer level (RTL), and then takes the RTL design through the synthesis and verification stages at the gate level to produce a physical design ready for fabrication and manufacturing test [18].

There are many different distributors for EDA and CAD tools and they act at many different parts of the design workflow. Among them, two of the biggest ones are Cadence Design Systems and Synopsys. Cadence Design Systems is a company that provides tools, IPs and hardwares necessary for the entire electronics design chain, from chip design to chip packaging to boards and systems [19].

As mentioned above, to run the digital design workflow with Cadence Digital Systems, the Genus and Innovus are utilized. Genus is the tool responsible for the synthesis for RTL. To run the synthesis, a script is written in Tool Command Language (TCL). TCL is a standard language syntax for writing tool scripts that is used by most CAD tools [16]. The TCL script contains all the necessary commands for the tool to run the logical synthesis. To accomplish a complete timing analysis, constraints need to be specified in the Synopsys Design Constraints (SDC) file. This file contains the clock definitions, the delays that need to be accounted for in the chip as well as many other timing restraints.

## 2.5 IC Fabrication Technology

IC fabrication can be separated in five major steps as seen in Figure 2.5. The first step is purchasing the starting substrate wafer. The second step is the process fabrication of the IC on the wafer. Third step is the wafer sort and test (in this step, bad die is discarded). Fourth step is the packaging, where the good die are cut and packaged. The fifth step is the mark and final test to guarantee that the ICs were not damaged during packaging [4].

To go through this process, first it is necessary to define the technology to be used in the implementation. This is dependent on the type of application and the foundry that will produce the hardware. XFAB is the largest analog/mixed signal foundry group, with a large variety of technologies ranging from 1um to 0.13 um [20]. Among the many available technology libraries, two were focused to test the applications XH018 and XC018.

The XH018 series is X-FAB's 0.18 micron Modular Mixed Signal High Voltage (HV) Complementary Metal Oxide Semiconductor (CMOS) Technology. Based upon the industrial standard single poly with up to six metal layers 0.18-micron drawn gate length Nwell process. This technology operates with low power 1.8V, 3.3V and can operate at temperatures of up to 175 Celsius [21].

The XC018 series is XFAB's 0.18 micrometer Modular Mixed Signal Technology. Based on the industrial standard one-poly-three-metal process baseline, wide selection of options such as

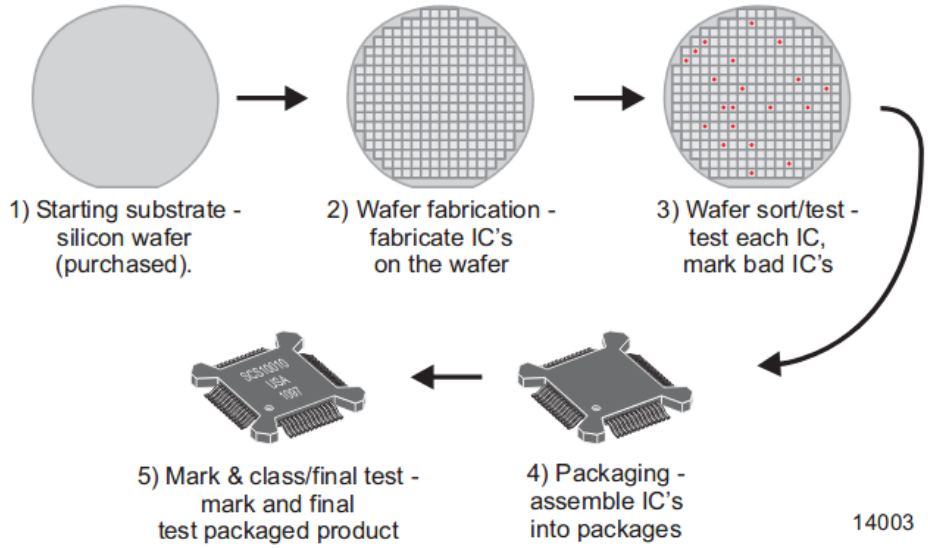


Figure 2.5: Steps in IC Fabrication [4]

standard or low power, 1.8V core voltage with either 3.3V or 5.0V I/O voltage are easily integrated through the modular approach [22].

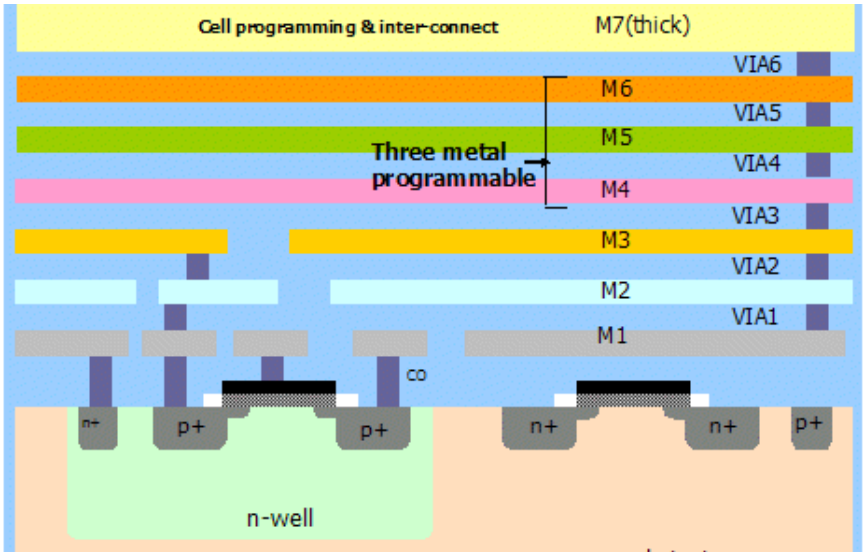


Figure 2.6: Camadas de Metais para roteamento [5]

These technologies contain various different metal configurations differing in the number and types of metal used. These metal layers are the metals that are available for routing the ASIC system. A represententation of these metals can be viewed in Figure 2.6. The configurations available in the XH018 and XC018 are MET2_METMID utilizing the metal layers up to M2, MET3_METMID utilizing utilizing the metals up to M3, MET4_METMID utilizing up to M4, MET5_METMID utilizing up to M6, MET2_METMID_METTHK utilizing up to M2 and the M7 layer, MET3_METMID_METTHK utilizing up to M3 and the M7 layers and MET4_METMID_METTHK utilizing up to M4 and M7.

## 2.6 System Specifications

Timing is utilized to describe the speed of a digital system. In timing, the delay in the circuit and the effects in the system are both analysed. Usually in the system, there will be a number of paths that require attention, called critical paths [23].

To be able to define the timing, a few definitions are necessary. The first definition is the rise time, which is the time for the waveform to rise from 20% to 80% of the steady-state value. The fall time is the time it takes the waveform to fall from 80% to 20% of the steady-state time. The propagation delay time is the maximum time for the input crossing 50% to the output crossing 50%. The contamination delay time is the minimum time from the input crossing 50% to the output crossing 50% [23].

One of the most important timing definitions in digital design is the slack. The slack is the remaining time after the information passes through the whole system. It is calculated through equation

$$Slack = LaunchEdge - ArrivalTime \tag{2.1}$$

With this, we can see that the larger the slack the more the system the system remains in idle and therefore the system can be improved in regards to the timing. Given a negative slack, we can determine that the system is not completing the desired tasks in time before a new task arrives, resulting in possible faulty behavior.

In the worst case scenario, clock skew is added. Clock skew, also called uncertainty, is the difference between the arrival of clock signals at different registers in a clock domain, as seen in Figure 2.7. The uncertainty can also be used to help adjust conservative constraints into more realistic ones by utilizing a negative uncertainty. This affects the propagation time since the skew and the flip-flop setup time are subtracted from the clock period to arrive at the time left for propagation [7].

After defining these timing constraints in regards to the clock, it is necessary to define the constraints in regards to the data. For this, we define input delays and output delays. "The input delay is the arrival time of external paths at an input port relative to a clock edge. Meanwhile, output delay represents the delay of an external timing path from an output port to a register input." [7]. Input delay can be seen in Figure 2.8 and output delay can be seen in Figure 2.9.

Data required time for a setup is the minimum time required for the data to get latched into the destination register and can be described by

$$DataRequiredTime(setup) = ClockArrivalTime - Tsu \tag{2.2}$$

The setup time, or the $Tsu$, is a fundamental requirement of all flip-flopped devices. The data required time for hold is the minimum time required for the data to get latched into the

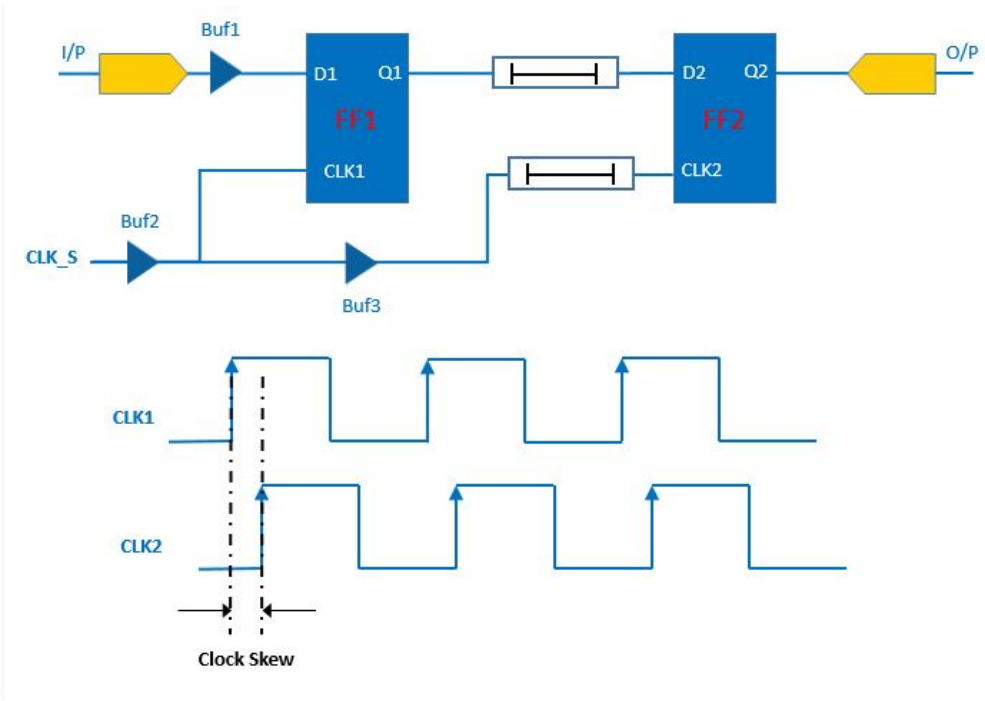


Figure 2.7: Clock Skew [6]

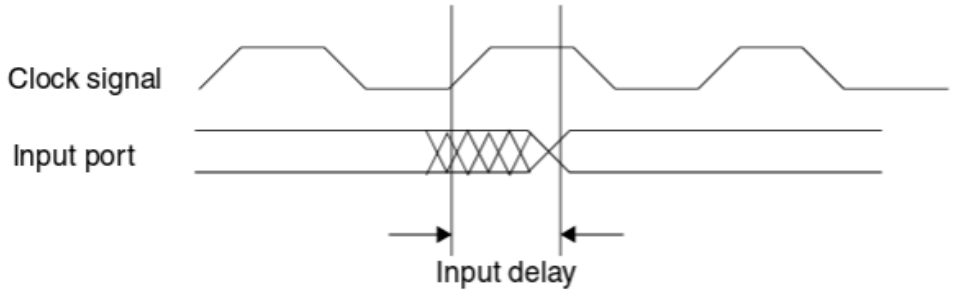


Figure 2.8: Input Delay [7]

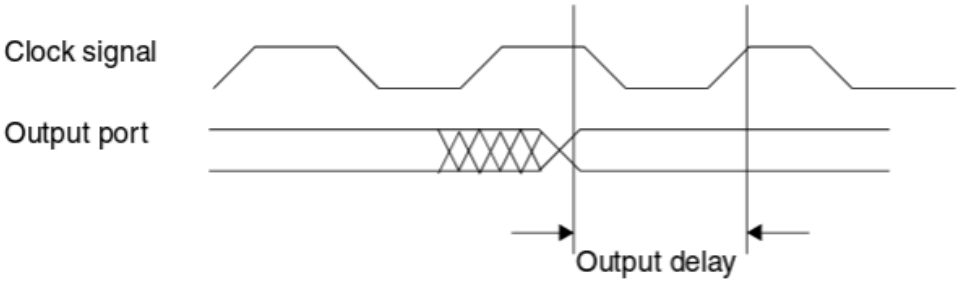


Figure 2.9: Output Delay [7]

destination register [24].

$$DataRequiredTime(hold) = ClockArrivalTime + Th \tag{2.3}$$

CMOS technology is very power-efficient given that they dissipate nearly zero power when idle. With a high number of transistors and a high clock frequency in a system, the power consumption becomes an important factor to be analysed. To analyse power consumption, we need to take into account the static dissipation and the dynamic dissipation.[23]

The static dissipation is due to secondary effects of the CMOS transistors. Some of these effects are the subthreshold conduction through off transistors, tunneling current through gate oxide and leakage through reverse-biased diodes. [23]

When a transistor is off, there is still a small amount of subthreshold current. This current is exponentially dependent on the threshold current. Leakage current through the gate is dependent on the oxide thickness. This becomes more important with gate oxides of 20A or thinner. There is also a static dissipation from reverse biased diode leakage between diffusion regions, wells and the substrate. In modern processes, diode leakage is much smaller than the subthreshold current or the gate leakage [23].

The dynamic dissipation is due to charging and discharging load capacitance. The primary component is the charging of load capacitance. To describe an equation where $\alpha$ is the activity factor is utilized [23].

$$P = \alpha C V_{dd}^2 f \tag{2.4}$$

The third specification, IC area, is based on fabrication cost. Given that the fabrication process is long, demands a controlled enviroment and expensive machinery to accomplish, the cost for fabrication and prototyping is high. To have an estimate for prototyping costs, an Europractice schedule schedule was utilized and costs for Multi Project Wafers (MPW) from 2019 [25]. There were 50 runs for XFAB technology and the cost for the XH018 technology is 1450 euros per millimeter squared. Therefore, it is necessary to fit a large number of ICs in a single wafer to dilute the price of each chip, which creates a demand for the smallest possible design.

Considering the timing and power consumption specifications, the current works seek to compare the implementation of a BCH decoder in FPGA with ASIC, as described in the following chapter.

# Chapter 3

# Methodology

*The chosen methodology implemented in this work is the top-down. As such, the process of implementation followed a few work flows to guarantee greater efficiency and better end results within the project.*

## 3.1 Design Flow

The current work seeks to develop an ASIC implementation of a BCH decoder in accordance with the DVB-S2X standard and compare it to the FPGA implementation developed in another work. In this study, the minimum clock frequency specification was 100MHz, due to the maximum processing of the LDPC of 5 megasymbols per second, with the smallest possible power consumption. This BCH decoder embedded in FPGA utilized VHDL representation and was simulated in software. This utilized many different aspects that couldn't be transported for ASIC, such as the use of IPs and base blocks with no direct equivalent. All of these codes were validated, and they were all partially synthesizable, so the decision was made to bypass a revalidation of these codes at the beginning of the project. This implementation, with adaptations, was applied in ASIC with the help of the Cadence framework.

The VHDL codes implemented in FPGA were verified to see if they could be synthesizable in the IC XFAB technology. After that verification, made with the Cadence - Incisive tool, it was determined that the previous implementation was incompatible with Cadence guidelines. Corrections on these codes were then required and alterations were made so that they could be synthesized.

The project workflow is presented on Figure 3.1. Starting with the FPGA VHDL files, the NCVHDl tool, was used to check for syntax errors. After this step, the code went through elaboration with the NCElab tool and through Linting checks with the Cadence tool Hardware Description Languag (HDL) Analysis and Lint (HAL). The HAL tool checks the codes for design consistency, reusability, portability, synthesizability, testability and semantic correctness [26]. This step is specially important due to the reusability and portability, since chip designs are usually used in many different technologies and projects.

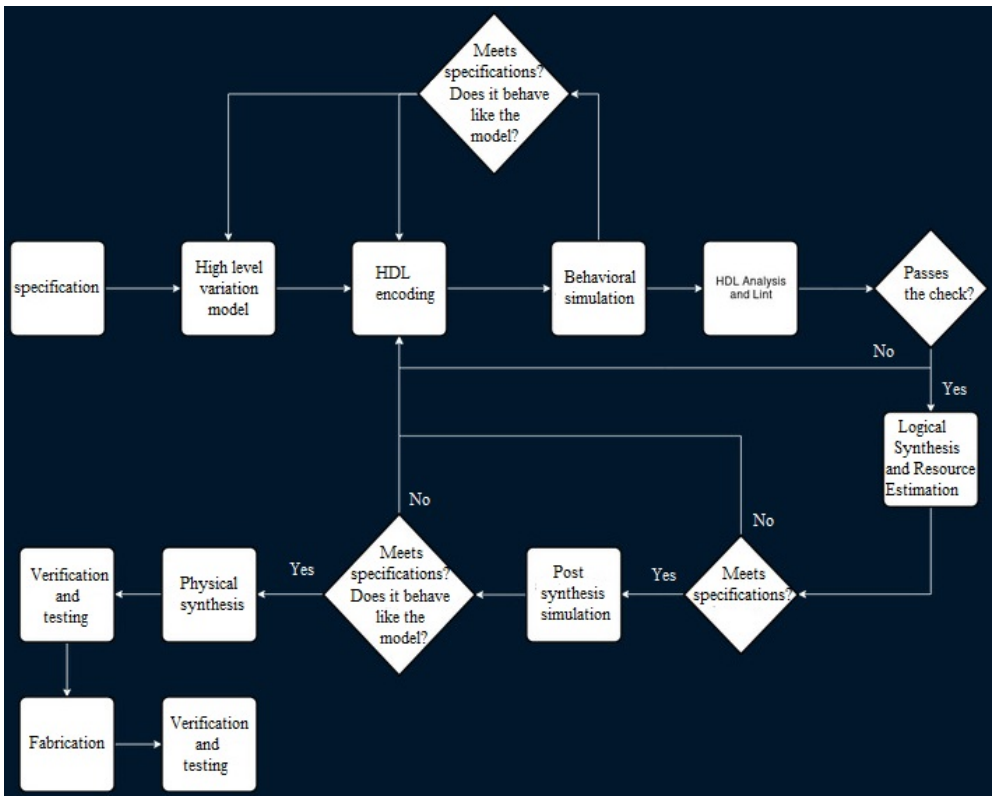


Figure 3.1: Digital Design Work Flow.

To have a thorough HAL analysis, another workflow is used, as seen in Figure 3.2, one that is more focused in this part of the design. After the usage of HAL, the Genus tool was employed to start performing an iterative process that runs a logical synthesis with the codes and checks if they meet the specifications. In this step, XFABs technologies XC018[22] and XH018[21] were utilized.

For each XFAB technology, there are many different types of metal configurations and cell types. The technologies have high speed cells with low power consumption and contain a specific library with all the available cells to implement in digital circuits. In the XC018, there were two different types of process versions (MOSST and MOSLP), while the XH018 only holds one (LPMOS). In the XC018, each process modulation has seven different possibilities for metal configuration. In the XH018, there are only five. This adds up to a wide number of possible configurations with varied sizes and power consumption, dependent on the basic cells implemented in each of these technologies process version.

To aid in the logic synthesis, a script was created in this work to automate this process. Utilizing Bash[27], a script was created that accessed the folders for the specified technology and gathered the LEF, tech and lib files. The Bash script then puts all the file paths in the template TCL file, Template_Genus_Synthesis. The script then runs the TCL file for all the possible metal configurations available in the technologies library. For each of these possibilities, detailed reports for resource estimation are created in the log folder. The script creates an additional file with the most relevant information to help identify and organize the results from the three detailed reports.

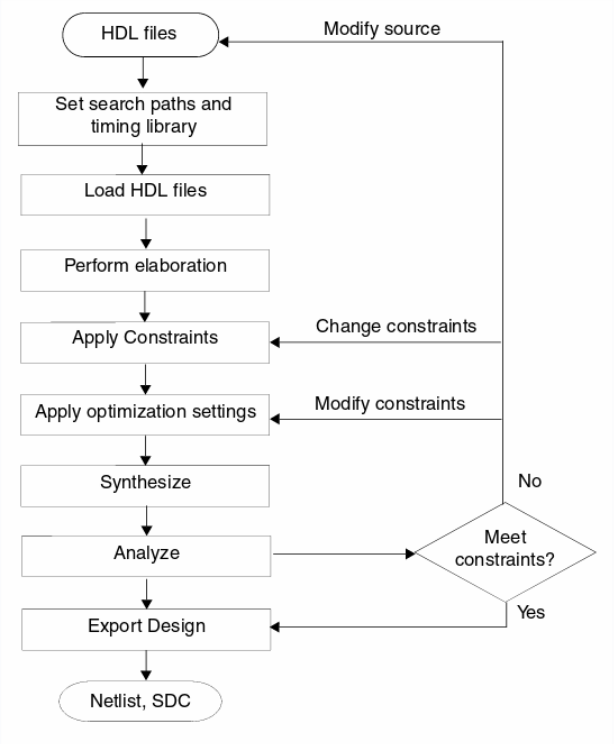


Figure 3.2: Generic Genus Work Flow. [7]

The gathered data for the same technology is stored in the files named timing report, power report and cell report. In the timing report, the arrival time of the clock, the required time for the whole system and the final slack are all informed. In the power report, the number of cells, the leakage power, the dynamic power and the total power consumption in the system are informed. Lastly, for the cell report, the instances of cells and the total area of the chip are informed. All these files are in the .csv file format to facilitate the use of this information in softwares such as Octave[28].

The automation script was implemented due to the fact that each synthesis could take between ten to forty minutes, and each technology holds five to seven different metal configurations, which could amount to almost five hours of simulation. Through the automation, it is possible to run these simulations on the background or during periods of time no other task is running. It also aids in organizing desired results for different applications, and facilitates the process for the logical synthesis. For the present work, the main focus was the comparison between the final resource estimations. Therefore, the previously mentioned information was stored together. This script can also be adapted to gather other information, depending on the application.

To specify the timing constraints, an evaluation of the system and standard requirements is necessary. Given the clock specification of 100MHz, an input and output delay of 1% of the clock was initially utilized. After verifying that the constraint was met, these delays were changed to have a minimum of 0.1 ps and a maximum of 1 ps. With the delays fixed, the clock's frequency was then increased to reach the maximum that the system could accomplish.

To have a complete comparison with the FPGA implementation the same procedures need to

be utilized. The FPGA implementation did not take into account possible delays in the system. Therefore the delays were discarded and only the clock was utilized to see the systems performance. With this, different values for the clock were tested to see which was the highest possible clock that fulfilled the timing constraints.

During the timing analysis, different implementations for the First In First Out (FIFO) memory block were created since the original FPGA implementation utilized an IP. After these implementations were validated, they were synthesized with the rest of the system. For this, the packages of each of these FIFO blocks had to be inserted to the original package blocks. After this was completed, a few of these blocks were synthesized and a comparison between them was made to determine which would be the most efficient block.

To optimize the FIFO, many five different architectures were implemented and verified. All of these architectures considered that the message could be written eight bits at a time. The first one was the fifo_v1 this architecture follows a combinational logic with two processes, but this version led to high power consumption. The second implementation, fifo_v2, it also has combinational logic but has three processes that help paralyze the tasks. The other three implementations are based on finite state machines. The fifo_FSM_PAR allows the FIFO to be written and read at the same time. The fifo_FSM allows only one action at a time, either writing or reading, and has two different states. Meanwhile fifo_FSM2 can only perform one action at a time but has three different states, one state to guarantee that the information will not be written in an inaccessible location. All of these were validated and the fifo_FSM2 has the correct behavior and the lowest power consumption.

Finally, after this part of the workflow is complete, and the synthesized codes meet the constraints, the resource estimation results are compared to the previously established results of the FPGA design, to determine the benefits from using either design and determine which should be implemented.

The final validations of the adapted codes were made in parallel with the resource estimation step. This was due to the fact that the original codes were already validated, and the implemented changes would be small enough that they would not imply in great changes to the system's behavior. Another reason for this was the necessity to acquire results that verified if there would be considerable improvements through the implementation in ASIC.

# Chapter 4

# Experimental Results

*The results are presented according to the previously stated objectives, and are organized in relation to code adaptation, timing, power consumption and area. Given that the adapted code is synthesizable, the resource estimation can be gathered and analysed.*

## 4.1 ASIC Code Adaptation

With the modification of the codes to fit the Cadence guidelines, four packages referenced by the VHDL blocks remain. The GlobalConstantsPkg that holds all constants, the BchDecAlphaPkg that holds the alpha values for future calculations, the BchDecPkg that holds the types, subtypes and entity references for all blocks and the Dvbs2xPkg that contains types used in the toplevel of the design.

Originally for the BCH decoder, many blocks were implemented and tested. The BchDecAlphaRomsCs is a memory that holds all the alpha values. The BchDecGalMult is the block that is responsible for creating the polynomials base for future processing. The BchDecSyndPar and the BchDecSyndGen are used together to calculate the syndromes. The BchDecSyndGenPar generates all the syndromes in parallel. The BchDecRiBM_fsm implements the Berlekamp-Massey algorithm to decode the BCH code. The BchDecChienSearch implements the Chien search for error-correction procedures. The fifo_FSM2 block implements the memory. Lastly, the Dvbs2xBchDec is the toplevel of the design and currently works as a controller for the FIFO.

During development, some of these blocks were altered or discarded. The toplevel was also separated into two different files: Fifocontroler_8bit, that contains the logic for controlling the FIFO, and the Dvbs2xBchDecLogic_8bit, that has only the toplevel logic. Figure 4.1 shows the block diagram and the function of the implemented blocks. The circular blocks are memory, and rectangle blocks are the necessary decoding algorithm blocks.

All the implemented blocks passed through the HAL and were synthesized. A few errors that the HAL points out remained due to the fact that the block could still be synthesized. Most of these errors refer to the types of ports used and signals that aren't the standard types(std_logic,

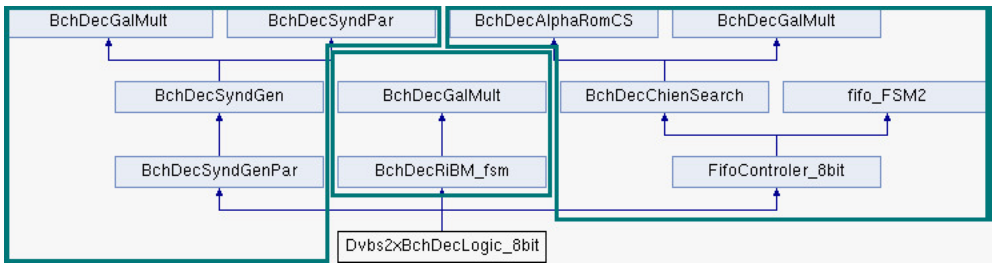


Figure 4.1: Implemented BCH Decoder Block Diagram

| Number | Metal Configuration | Available Libraries |
|---|---|---|
| 1 | MET2_METMID | MOSLP, MOSST |
| 2 | MET3_METMID | LPMOS, MOSLP, MOSST |
| 3 | MET4_METMID | LPMOS, MOSLP, MOSST |
| 4 | MET5_METMID | LPMOS, MOSLP, MOSST |
| 5 | MET2_METMID_METTHK | MOSLP, MOSST |
| 6 | MET3_METMID_METTHK | LPMOS, MOSLP, MOSST |
| 7 | MET4_METMID_METTHK | LPMOS, MOSLP, MOSST |

Table 4.1: Number to Metal Configuration

std_logic_vector).In later stages of the project, these errors should be eliminated because they could create portability issues. Warnings from the HAL were ignored due to the fact they do not influence in the final synthesized result.

## 4.2 ASIC Timing

Through the timing analysis, three clock frequencies were specified with the use of delays, and two clock frequencies without the use of delay. For the two clock frequencies without delay, a toplevel block that could only alter one bit at a time of the FIFO was utilized. With that block, came an analysis of 100MHz, the minimum system requirement, and 200MHz, the maximum frequency. For the clock frequencies with the use of delay, as described in chapter 3, the toplevel logic was the final implementation that could alter the FIFO eight bits at a time. With this, there was an analysis of 100MHz, 300MHz, the maximum frequency for LPMOS, and 490MHz, the maximum frequency that the MOSST technology could accomplish. The LPMOS was given extra attention in this analysis, due to the fact that it would be possible to produce a prototype in this technology.

The gathered data can be fully viewed in the tables located in the appendix. In the graphs displayed in Figures 4.2 to 4.14, each number corresponds to one type of metal configuration as shown in the Table 4.1.

For a clock frequency of 100MHz without delays for the LPMOS technology, all metal configurations had a positive slack. The worst configurations were MET3_METMID and MET4_METMID,

with 17 ps of slack, while the best were the MET5_METMID, with a slack of 80 ps. For the MOSLP technology all metal had a positive slack. The worst configuration was MET4_METMID, with a slack of 4 ps. The best configuration was MET5_METMID, with a slack of 126 ps. For the MOSST technology, all metal had a positive slack. The worst configuration was MET5_METMID, with a slack of 2686 ps. The best configuration was MET2_METMID, with a slack of 4047 ps. Given that all technologies had positive slack, there is a possibility of clock frequency increase, the technology with the highest margin for improvement in timing is the MOSST. A direct comparison between the slack of each technology can be seen in Figure 4.2.

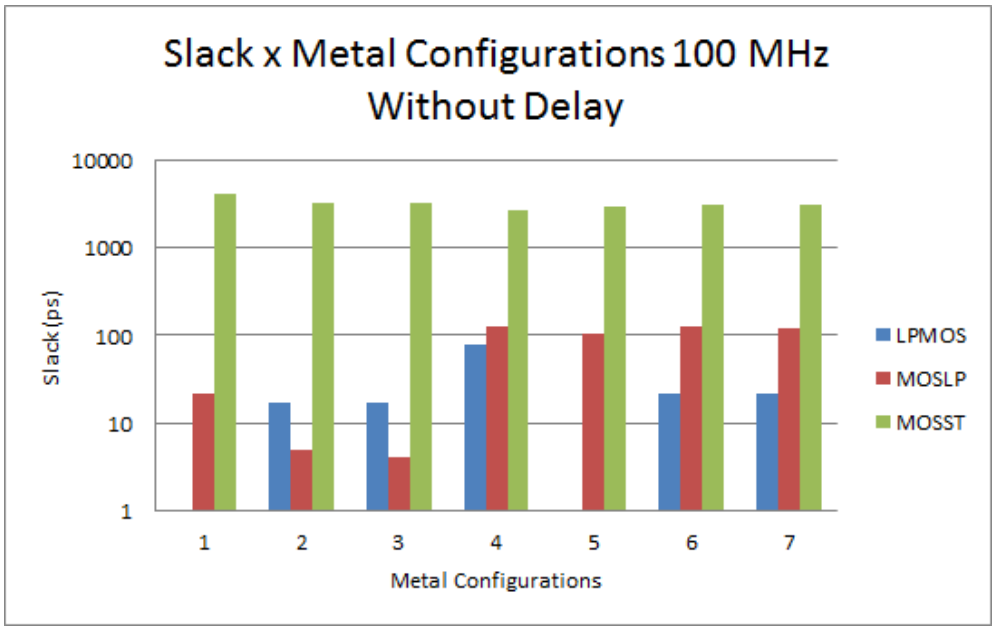


Figure 4.2: Timing 100MHz without delay

Given the results presented for 100MHz, the clock frequency was then increased to 200MHz. For this clock frequency, all the technologies had positive or null slack. Given that the highest slack among them was of 2 ps, there was very little room for improvement, making this the maximum frequency for this implementation.

For a clock frequency of 100 MHz, with the addition of delay, all metal configurations of the LPMOS technology had positive slack. The worst configuration was, MET4_METMID with slack of 5 ps, while the best configuration was MET5_METMID, with a slack of 49 ps.For the MOSLP technology, all metal configurations had positive slack, with the MET4_METMID having the smallest slack, 16 ps, and MET5_METMID having the largest, 186 ps. For the MOSST technology, all metal configurations had a positive slack, with a much larger margin for improvement. The smallest slack, 2631 ps, was from MET5_METMID, and the largest, 3849 ps, from MET4_METMID. The comparison can be viewed in Figure 4.3

Given the clock frequency of 300 MHz, with delay, all technologies had positive or null slack. This frequency is specified as being the maximum frequency for the LPMOS technology. All technologies had a slack of 0 ps.

The largest clock frequency that could be achieved by one of the three technologies with delay

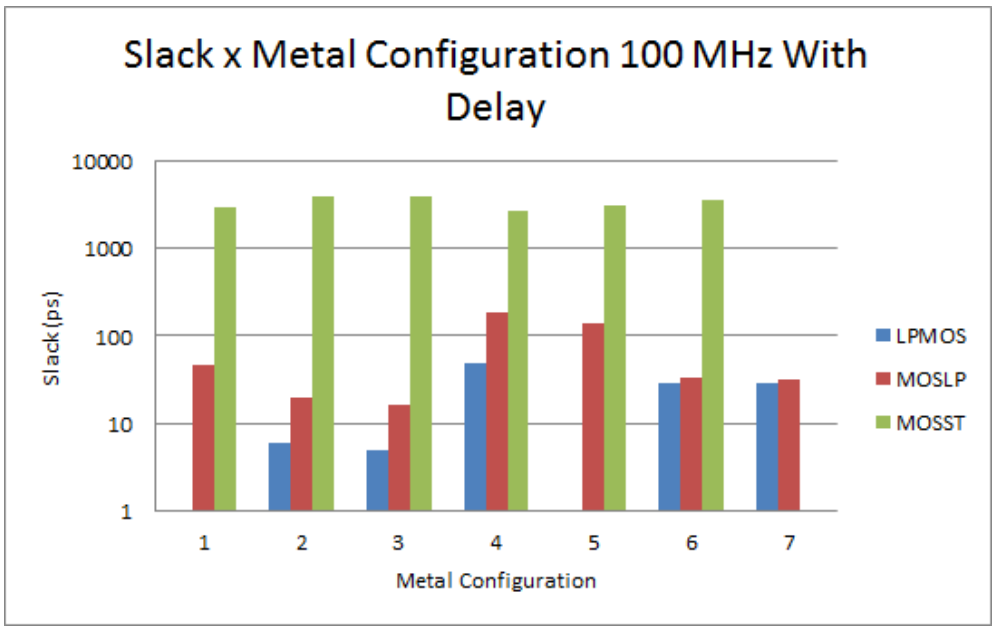


Figure 4.3: Timing 100MHz with delay

was 490MHz. In this frequency, only the MOSST technology maintained a null slack. The LPMOS and MOSLP technologies had large negative slacks that can be viewed in Figure 4.4. This was then specified as the maximum frequency achieved by designed system. With this, it can be seen that the MOSST is the best for high speed designs.

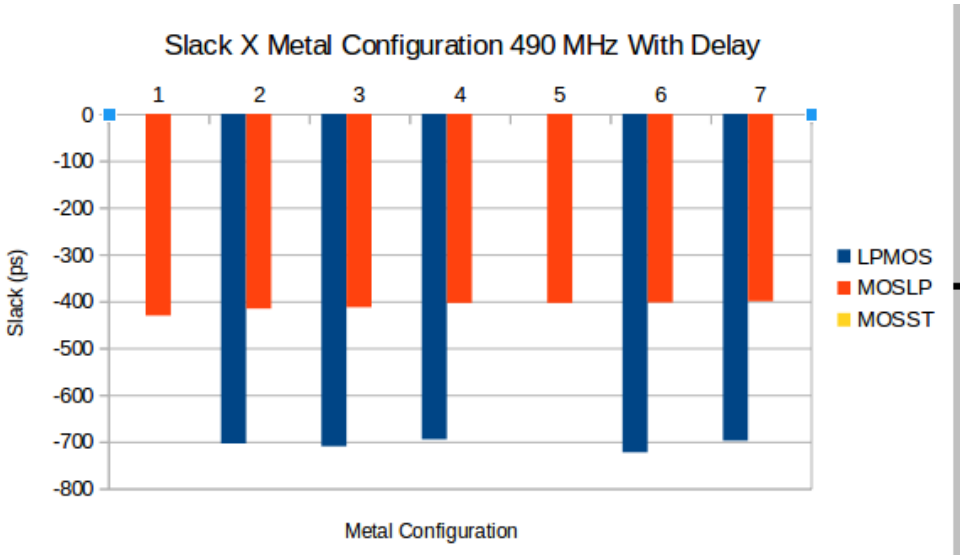


Figure 4.4: Timing 490MHz with delay

## 4.3 ASIC Power Consumption

With the clock frequency of 100MHz, without delay, the best overall technology was the MOSLP with the MET5_METMID with a power consumption of 177 mW. The MOSST was the

second best, with the same metal configuration consuming 182 mW. LPMOS technology had the worst power consumption, with MET5_METMID consuming 202 mW. This data can be seen in Figure 4.5

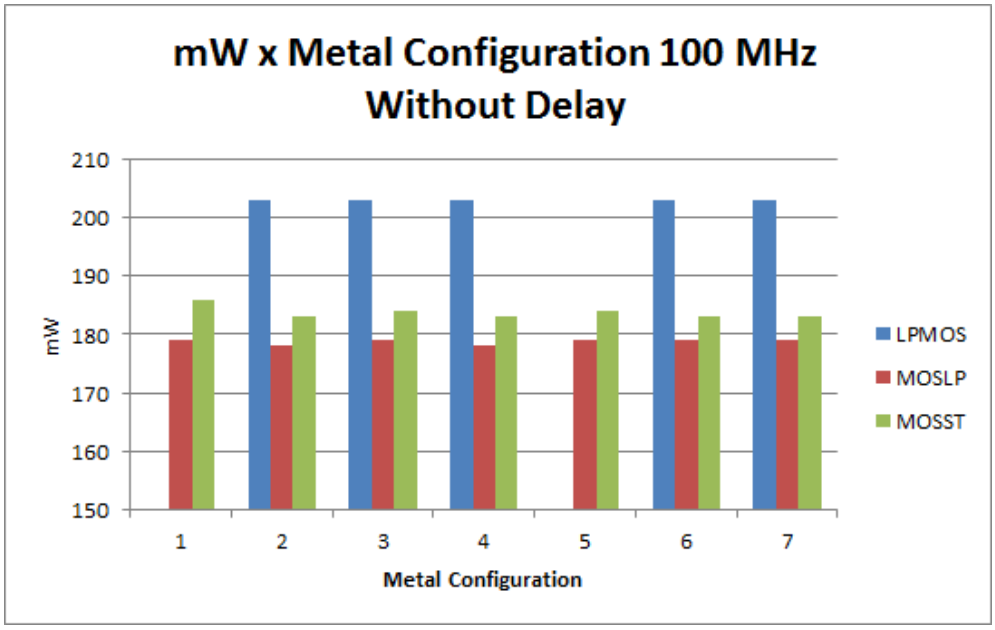


Figure 4.5: Power 100MHz without delay

For 200MHz, the best technology became the MOSST with MET5_METMID, and a power consumption of 143 mW. The MOSLP was the second best implementation, with MET4_METMID and a power consumption of 144 mW. The LPMOS technology had a power consumption of 175 mW with the MET3_METMID configuration. The comparison can be viewed in Figure 4.6

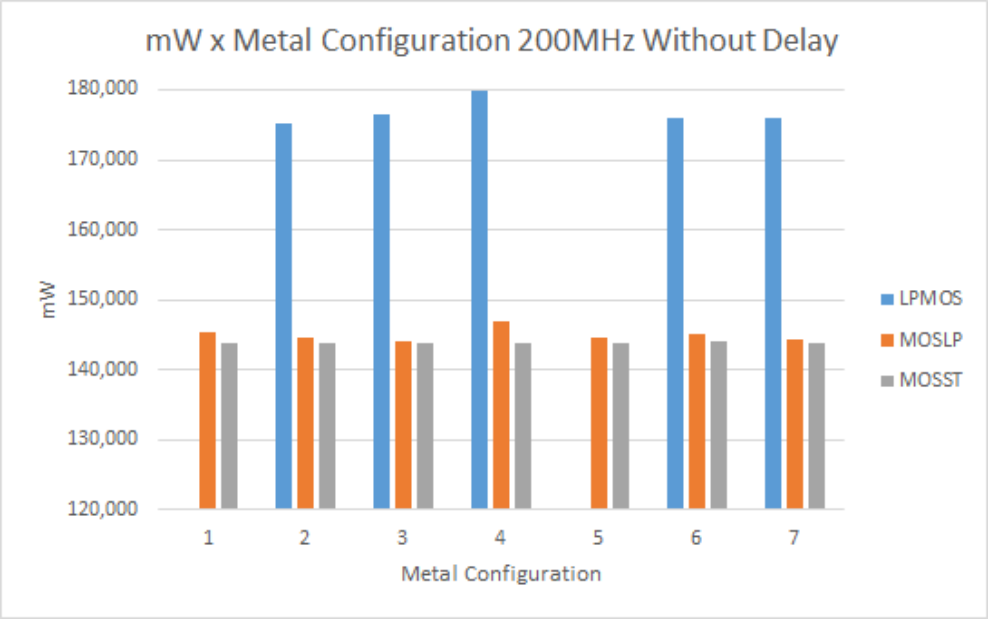


Figure 4.6: Power 200MHz without delay

For 100MHz with delay, there is an overall increase in the power consumption for all the technologies. The best technology was MOSLP with MET5_METMID, consuming 183 mW. The MOSST with MET3_METMID_METTHK consumed 188 mW, while LPMOS, with MET5_METMID, consumed 208 mW. This can be seen in Figure 4.7.

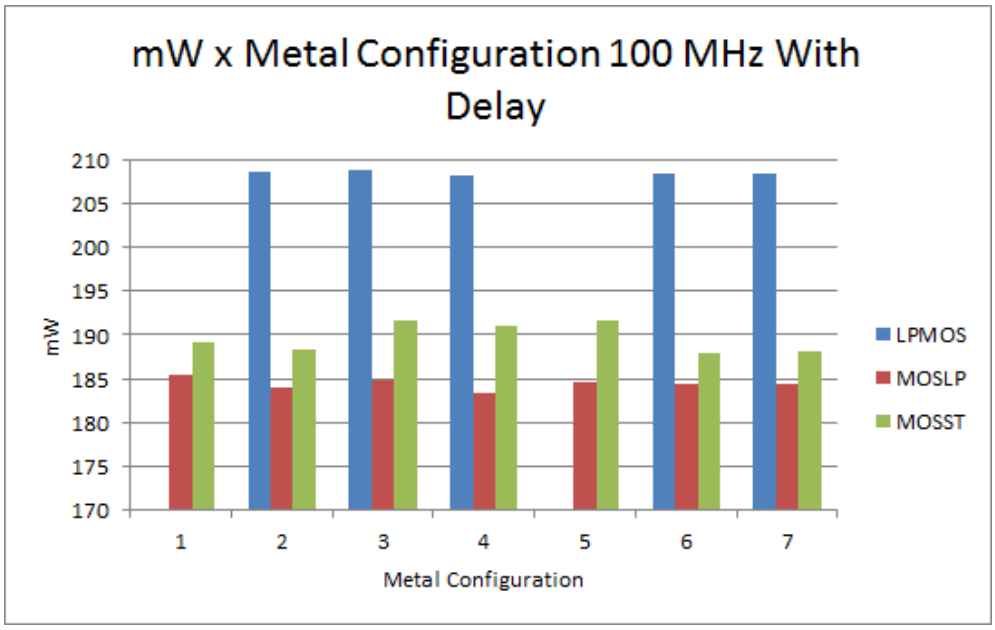


Figure 4.7: Power 100MHz with delay

With 300MHz, the MOSST is the technology with the lowest power consumption, as seen in Figure 4.8,with MET5_METMID consuming 554 mW. The second best technology is the MOSLP with MET4_METMID_METTHK, consuming 573 mW. The LPMOS had a considerably larger power consumption of 677 mW in the MET4_METMID_METTHK.

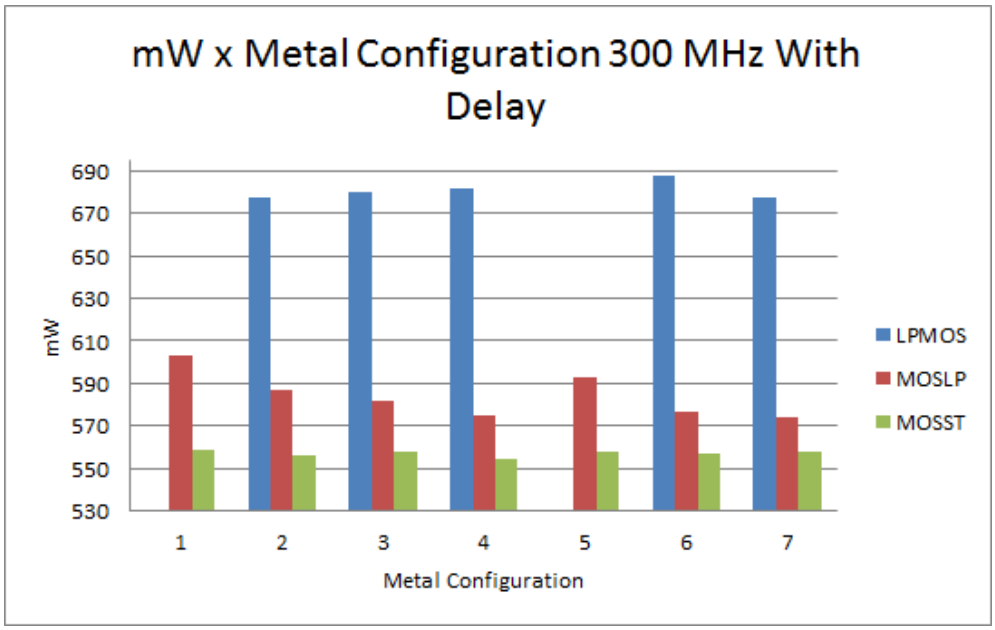


Figure 4.8: Power 300MHz with delay

With the clock frequency of 490MHz, with delay, both LPMOS and MOSLP could not complete the tasks in this frequency, so only the MOSST is analysed for power consumption. For MOSST, the best configuration was MET3_METMID_METTHK, with 940 mW of power consumption. The comparison between these results can be seen in Figure 4.9

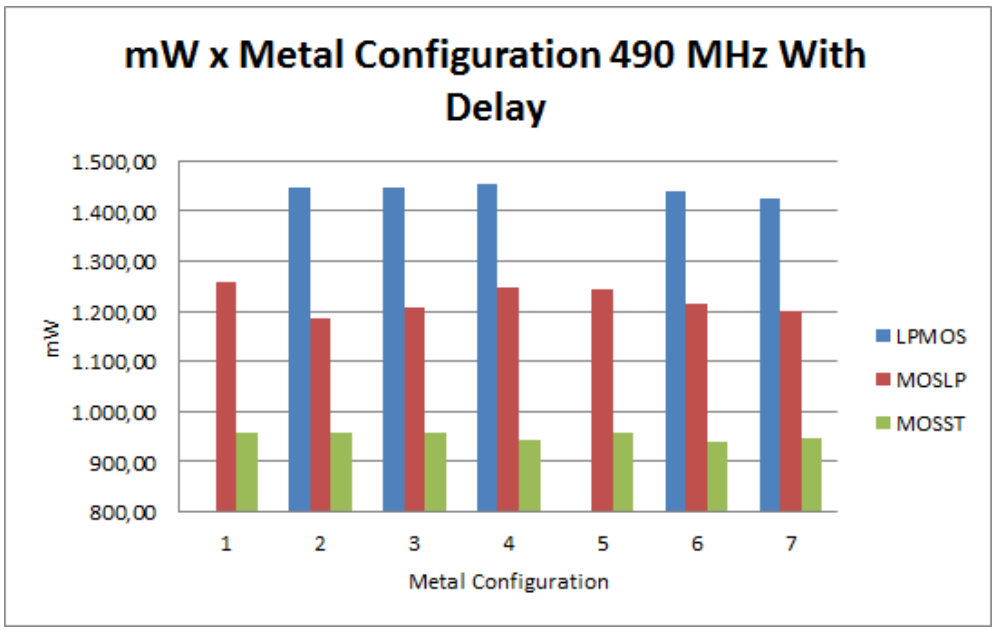


Figure 4.9: Power 490MHz with delay

As the power consumption increases, the viability of the LPMOS and MOSLP reduces. Both of these technologies are focused on low power, while the MOSST is focused on high speed circuits. Therefore, it is easier to optimize this technology.

## 4.4 ASIC Area

For 100MHz, the best technology was MOSST with MET5_METMID, occupying an area of 957048 $\mu m^2$. The second best technology was the LPMOS with MET5_METMID, occupying 958711 $\mu m^2$. The MOSLP occupies the largest area, 958966 $\mu m^2$, with MET3_METMID.

For 200MHz, the best technology remained the MOSST with MET5_METMID, occupying 958511 $\mu m^2$. The second best technology was MOSLP with MET3_METMID 968266 $\mu m^2$. The LPMOS with MET4_METMID_METTHK occupies 974280 $\mu m^2$.

For 100MHz, with delay, the metal configuration of MET5_METMID was the best for all the technologies. The MOSST occupied the least area, 957128 $\mu m^2$, then the LPMOS, 958717 $\mu m^2$, and the MOSLP occupies the largest area, 958954 $\mu m^2$.

For 300MHz, the best technology was MOSST with MET5_METMID, occupying 967489 $\mu m^2$. The second best technology is MOSLP with MET5_METMID, occupying 1004701 $\mu m^2$. The LPMOS occupies 1014013 $\mu m^2$ with MET3_METMID.

As previously stated, the LPMOS and MOSLP implementations in 490 MHz can not ac-

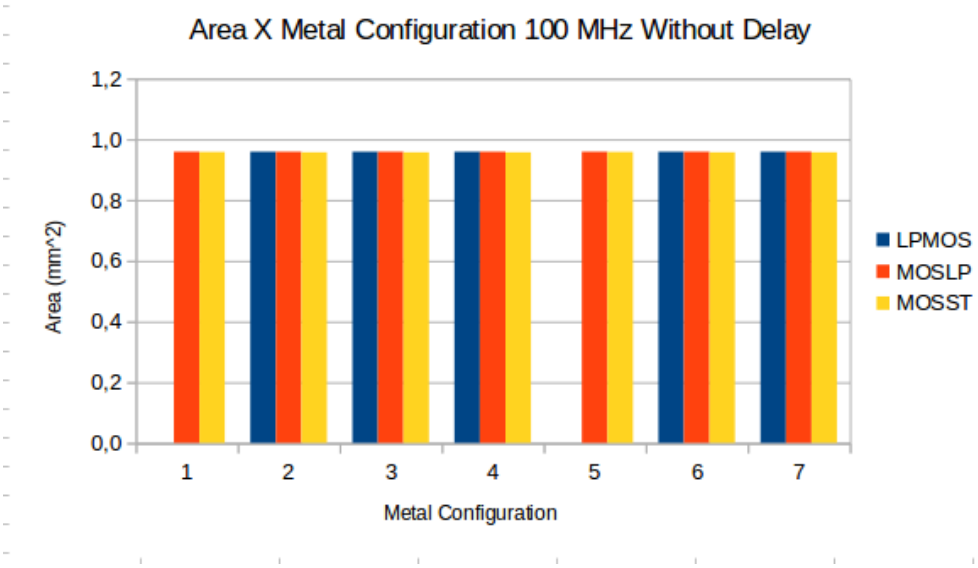


Figure 4.10: Area 100MHz without delay

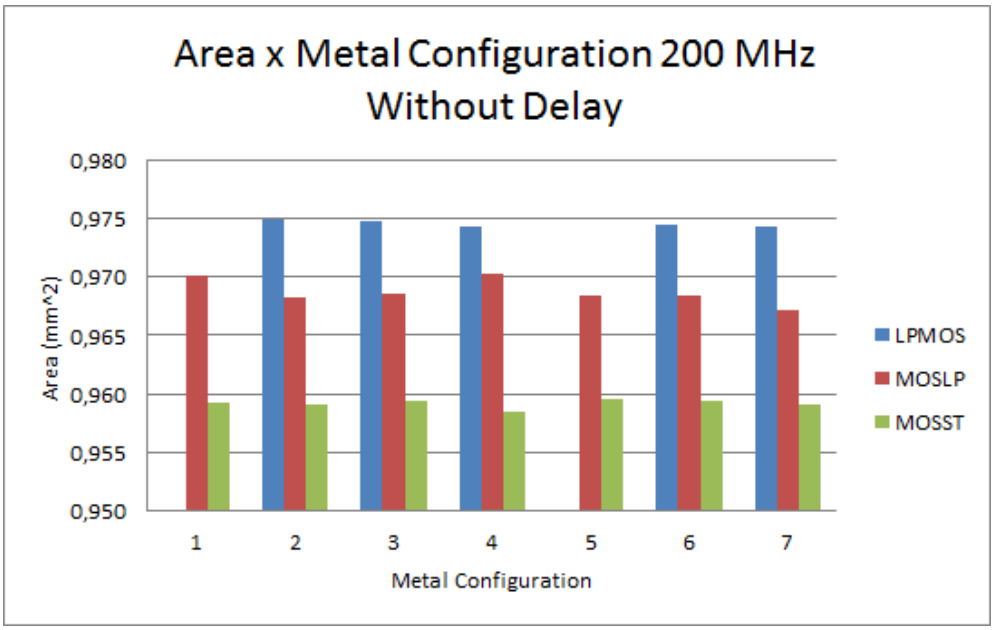


Figure 4.11: Area 200MHz without delay

complish the timing constraints. Therefore only the MOSST will be analysed. The best metal configuration is MET5_METMID, occupying 1009177 $\mu m^2$, which is a small increase from the other synthesized blocks.

For the system specification, clock frequency of 100 MHz, the best technology in regards to the power would be the MOSLP with the MET5_METMID. On the other hand if the power consumption became a secondary factor compared to the area the best implementation would be with MOSST with MET5_METMID. The LPMOS consistently had the worst results among the technologies making it the least desirable to implement the final system.

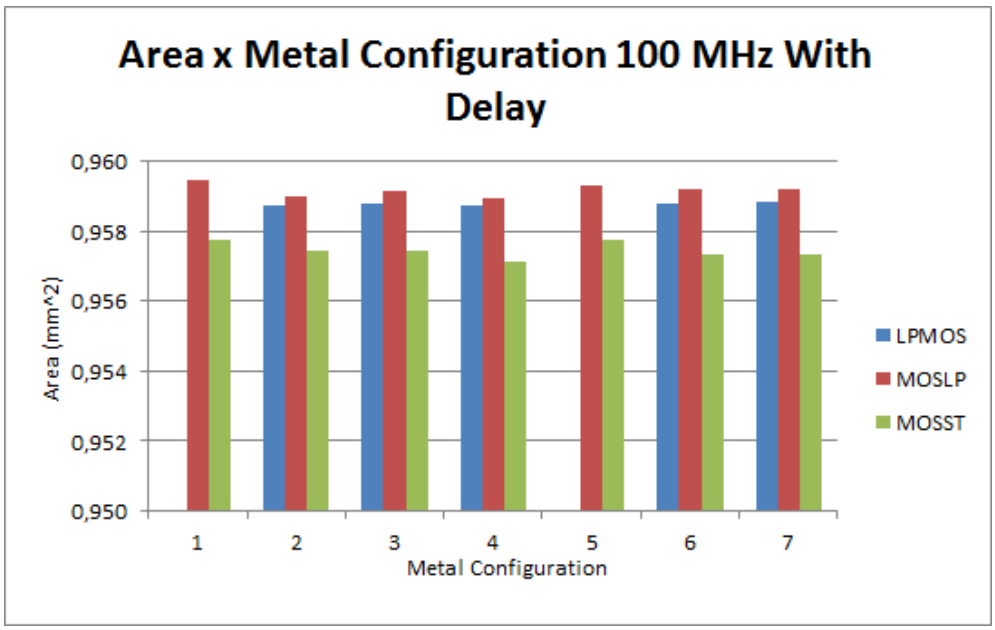


Figure 4.12: Area 100MHz with delay

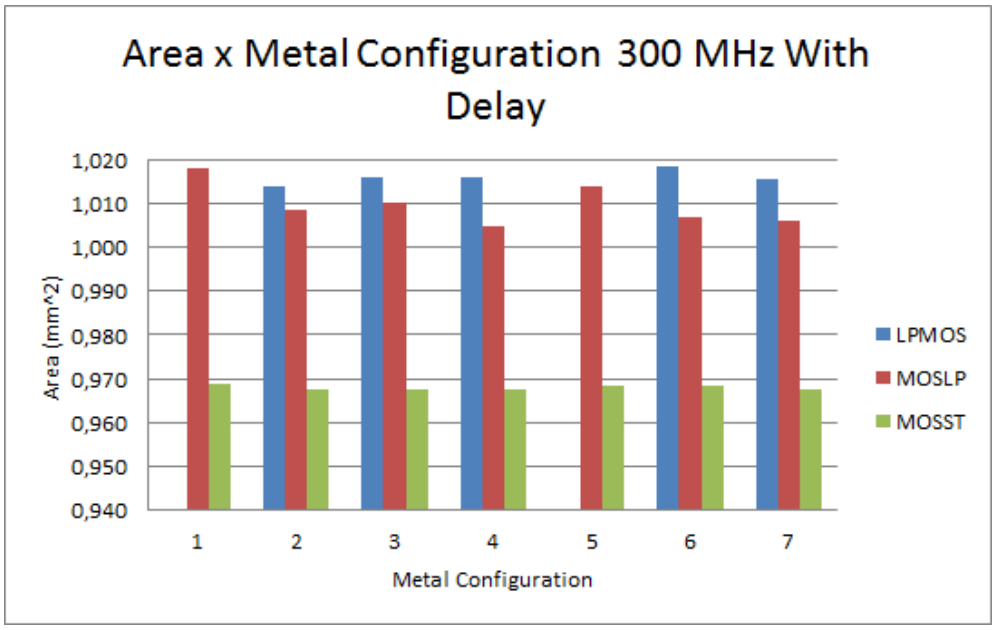


Figure 4.13: Area 300MHz with delay

## 4.5 ASIC x FPGA Comparison

Two FPGA implementations were synthesized, one embedded with the original FIFO IP, FPGA IP, and another with the FIFO developed for the ASIC system, FPGA FIFO. These implementations utilized a clock frequency of 100 MHz and both had slack of 0. The maximum frequency for the FPGA implementations were of 120 MHz with delay. Comparing these results to the ASIC implementations, there is a large difference between each maximum frequency.

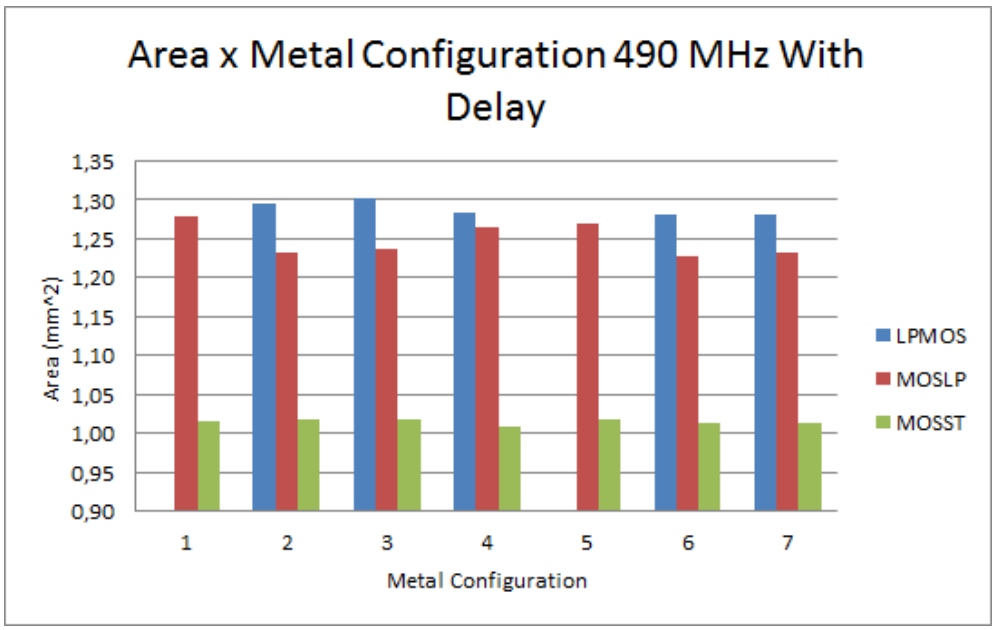


Figure 4.14: Area 490MHz with delay

| Technology | Library/Model | Maximum Frequency | Power Consumption (100 MHz) | Area (100MHz) |
|---|---|---|---|---|
| xh018 | LPMOS | 300 MHz | 208.18 mW | 958717 $\mu m^2$ |
| xc018 | MOSLP | 300 MHz | 183.44 mW | 958954 $\mu m^2$ |
| xc018 | MOSST | 490 MHz | 188.01 mW | 957128 $\mu m^2$ |
| FPGA IP | Zynq Ultrascale | 120 MHz | 601 mW | N/A |

Table 4.2: Comparisson Between FPGA and ASIC Implementations

For the implementation with the FPGA IP the power consumption was of 601 mW while for the implementation FPGA FIFO the power consumption was of 630 mW, this shows these architectures are comparable with the implemented FIFO having room for improvement. Comparing these results to the worst of the ASIC implementations for the same frequency, the MOSLP consumed 208 mW. This shows that there is considerable improvements in power consumption with the implementation in ASIC.

Lastly, for the occupied area the FPGA has a constant area which is much larger than any of the ASIC implementations. A table with this information is presented in Table 4.2

The results gathered from the synthesis demonstrated that it is possible to use XFABs technologies to implement this system. With the comparison between the ASIC and FPGA implementation, we see that the system can be improved with the use of ASIC.

For higher frequencies, the MOSST technology presented better overall results, as well as being the only technology capable of operating in 490 MHz. For smaller frequencies, such as the 100 MHz specification, the MOSLP has better power consumption. The LPMOS had worse results than the other technologies, which is expected considering it is an older technology. Even though the LPMOS had the worse results, for lower frequencies, the results remained close to the other

options. Therefore, it is still a valid option for prototyping.

In terms of the metal configuration, the MET5_METMID had the better overall results, consistently being one of the best options for the implementation. Other metal configurations, such as MET3_METMID and MET4_METMID_METTHK, had the best results in specific frequencies. Therefore, we can not specify one single metal configuration as being the ideal for the system.

Analysing the system with the DVB-S2X standard specifications, we see that there needs to be improvement in timing. The DVB-S2X standard requires a maximum clock frequency of 640 MHz, while the implemented system could only reach 490 MHz. The power consumption also increased drastically with the increase of the clock period, which could be a liability in systems that require low power.

# Chapter 5

# Conclusions and Future Works

## 5.1 Conclusions

The use of the workflows aided in the adaptation of the FPGA codes to the Cadence framework. With the final implementation, all the codes fit the Cadence requirements and were synthesizable. There remained warnings and errors in the implementation, the correction of these could lead to better final design, with better portability and reusability.

The final FIFO can be implemented utilizing two different ways, either through the use of vectors or arrays. Utilizing arrays, the overall power consumption and area occupied was larger, but the time required to synthesize was drastically reduced. The vector implementation is harder to code, which demands longer periods of time to formulate the codes and to synthesize. Therefore, there is a possibility of improvement in the synthesis with the implementation of vectors.

With the proposed changes to the architecture the implemented system was synthesized in the 180 $nm$ technology and led to better results than the FPGA implementations. To reach the same power consumption as the FPGA the system needs to run at frequencies higher than 300 MHz. The ASIC system can also operate in higher clock frequencies than the FPGA implementation.

## 5.2 Future Works

For further works the first logical step towards prototyping would be the physical synthesis. After this step it would be necessary to extract the circuit with the parasitics and run the Layout Versus Schematic (LVS). With these steps concluded a prototype could be manufactured.

A second line of work would be the reevaluateion of the implemented system determine if it can be optimized to meet the different specifications, such as DVB-S2X timing requirement or low power applications focusing only on power consumption.

Another possible line of work would be the implementation of the other parts of the decoding process, such as the LDPC, in ASIC. The LDPC would be the first block to evaluate, since it is the largest block in the decoding process.

# REFERENCES


[1] TRANSLATION, E. Transmission system for advanced wide band digital satellite broadcasting association of radio industries and businesses. *Jul*, v. 31, p. 86, 2014.

[2] CHAVES, C. G.; LIMA, E. R. de; MERTES, J. G. A synthesizable bch decoder for dvb-s2 satellite communications. *Journal of Integrated Circuits and Systems*, v. 10, n. 3, p. 174–180, 2015.

[3] DAS, A. S.; DAS, S.; BHAUMIK, J. Design of rs (255, 251) encoder and decoder in fpga. *international journal of soft computing and engineering*, v. 2, n. 6, p. 2231–2307, 2013.

[4] INTRODUCTION to Integrated Circuit Technology. `shorturl.at/qwCFI`. Accessed: 2019-11-12.

[5] THE Platform Based SOC Design that Utilizes Structured ASIC Technology. `https://www.design-reuse.com/articles/9566/the-platform-based-soc-design-that-utilizes-structured-asic-technology.html`. Acessed: 2019-12-10.

[6] INTRODUCTION and Source of Clock Skew. `http://www.vlsi-expert.com/2016/01/skew.html`. Acessed: 2019-11-28.

[7] SYSTEMS, I. C. D. *Genus Timing Analysis Guide for Legacy UI*. [S.l.]: Cadence Design Systems, Inc., December 2017.

[8] KATOH, H. Transmission system for isdb-s. *Proceedings of the IEEE*, IEEE, v. 94, n. 1, p. 289–295, 2006.

[9] INDUSTRIES, A. of R.; BUSINESSES. Transmission system for advanced wide band digital satellite broadcasting. *Jul*, v. 31, p. 143, 2014.

[10] WHAT is DVB-S2? `https://www.dvb.org/about`. Acessed: 2019-08-21.

[11] EXTENDING DVB-S2 What is DVB-S2X? `https://www.dvb.org/standards/dvb-s2x`. Acessed: 2019-08-21.

[12] MOON, T. K. *Error correction coding: mathematical methods and algorithms*. [S.l.]: John Wiley & Sons, 2005.

[13] PETERSON, W. W.; BROWN, D. T. Cyclic codes for error detection. *Proceedings of the IRE*, IEEE, v. 49, n. 1, p. 228–235, 1961.

[14] WICKER, S. B.; BHARGAVA, V. K. *Reed-Solomon codes and their applications*. [S.l.]: John Wiley & Sons, 1999.

[15] WHAT is an FPGA? `https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html`. Accessed: 2019-11-12.

[16] BRUNVAND, E. *Digital VLSI chip design with Cadence and Synopsys CAD tools*. [S.l.]: Addison-Wesley, 2010.

[17] CHRISTIANSEN, J. *IC design methodology and related tools*. [S.l.]: CERN, November 2006.

[18] WANG, L.-T.; CHANG, Y.-W.; CHENG, K.-T. T. *Electronic design automation: synthesis, verification, and test*. [S.l.]: Morgan Kaufmann, 2009.

[19] TRANSFORMING the Way People Design Next-Generation Systems. `https://www.cadence.com/content/cadence-www/global/en_US/home/company.html`. Accessed: 2019-11-12.

[20] ABOUT X-FAB. `https://www.xfab.com/about-x-fab/`. Acessed: 2019-11-12.

[21] 0.18 Micron Modular HT HV CMOS Technology. `https://www.xfab.com/technology/cmos/018-um-xh018/`. Accessed: 2019-11-12.

[22] 0.18 Micron Modular CMOS Technology. `https://www.xfab.com/technology/cmos/018-um-xc018/`. Accessed: 2019-11-12.

[23] WESTE, N. H.; HARRIS, D. *CMOS VLSI design: a circuits and systems perspective*. [S.l.]: Pearson Education India, 2004.

[24] TIMING Analysis with Time Quest I. `https://www.coursera.org/learn/intro-fpga-design-embedded-systems/lecture/YE1wc/9-timing-analysis-with-time-quest-i`. Accessed: 2019-11-12.

[25] 2019GENERAL Europractice MPW runsSchedule and Prices. `http://europractice-ic.com/wp-content/uploads/2019/10/191007_MPW2019-general-v13.pdf`. Accessed: 2019-11-27.

[26] SYSTEMS, I. C. D. *Incisive Enteprise Specman Elite Testbench e Linting with HAL*. [S.l.]: Cadence Design Systems, Inc., December 2016.

[27] GNU Bash. `https://www.gnu.org/software/bash/`. Acessed: 2019-11-28.

[28] GNU Octave. `https://www.gnu.org/software/octave/`. Acessed: 2019-11-28.

# APPENDIX

Table 1 to 15 contain timing results,arrival time, required time and slack, for each possible metal configuration. Table 16 to 30 contain power consumption results, leakage, dynamic and total power, for each metal configuration. Table 31 to 45 contain area results, cell instances and total area occupied, for each metal configuration.

| Metal Configuration | Arrival | Required Time | Slack |
|---|---|---|---|
| xh018_xx31_MET3_METMID | 10000 | 9088 | 17 |
| xh018_xx33_MET3_METMID_METTHK | 10000 | 9089 | 22 |
| xh018_xx41_MET4_METMID | 10000 | 9088 | 17 |
| xh018_xx43_MET4_METMID_METTHK | 10000 | 9089 | 22 |
| xh018_xx51_MET5_METMID | 10000 | 9100 | 80 |

Table 1: LPMOS Timing fifo_FSM2 100MHz without delay

| Metal Configuration | Arrival | Required Time | Slack |
|---|---|---|---|
| xc018_xx10_MET2_METMID | 10000 | 9226 | 22 |
| xc018_xx11_MET3_METMID | 10000 | 9218 | 5 |
| xc018_xx12_MET4_METMID | 10000 | 9218 | 4 |
| xc018_xx13_MET5_METMID | 10000 | 9219 | 126 |
| xc018_xx14_MET2_METMID_METTHK | 10000 | 9227 | 103 |
| xc018_xx15_MET3_METMID_METTHK | 10000 | 9219 | 124 |
| xc018_xx16_MET4_METMID_METTHK | 10000 | 9218 | 119 |

Table 2: MOSLP Timing fifo_FSM2 100MHz without delay

| Metal Configuration | Arrival | Required Time | Slack |
|---|---|---|---|
| xc018_xx10_MET2_METMID | 10000 | 9472 | 4047 |
| xc018_xx11_MET3_METMID | 10000 | 9472 | 3306 |
| xc018_xx12_MET4_METMID | 10000 | 9472 | 3304 |
| xc018_xx13_MET5_METMID | 10000 | 9481 | 2686 |
| xc018_xx14_MET2_METMID_METTHK | 10000 | 9472 | 2961 |
| xc018_xx15_MET3_METMID_METTHK | 10000 | 9473 | 3128 |
| xc018_xx16_MET4_METMID_METTHK | 10000 | 9473 | 3128 |

Table 3: MOSST Timing fifo_FSM2 100MHz without delay

| Metal Configuration | Arrival | Required Time | Slack |
|---|---|---|---|
| xh018_xx31_MET3_METMID | 5000 | 4553 | 0 |
| xh018_xx33_MET3_METMID_METTHK | 5000 | 4412 | 0 |
| xh018_xx41_MET4_METMID | 5000 | 3840 | 0 |
| xh018_xx43_MET4_METMID_METTHK | 5000 | 4400 | 0 |
| xh018_xx51_MET5_METMID | 5000 | 4550 | 0 |

Table 4: LPMOS Timing fifo_FSM2 200MHz without delay

| Metal Configuration | Arrival | Required Time | Slack |
|---|---|---|---|
| xc018_xx10_MET2_METMID | 5000 | 4618 | 0 |
| xc018_xx11_MET3_METMID | 5000 | 4683 | 1 |
| xc018_xx12_MET4_METMID | 5000 | 4613 | 0 |
| xc018_xx13_MET5_METMID | 5000 | 4684 | 0 |
| xc018_xx14_MET2_METMID_METTHK | 5000 | 4765 | 1 |
| xc018_xx15_MET3_METMID_METTHK | 5000 | 4286 | 0 |
| xc018_xx16_MET4_METMID_METTHK | 5000 | 4717 | 0 |

Table 5: MOSLP Timing fifo_FSM2 200MHz without delay

| Metal Configuration | Arrival | Required Time | Slack |
|---|---|---|---|
| xc018_xx10_MET2_METMID | 5000 | 4615 | 0 |
| xc018_xx11_MET3_METMID | 5000 | 4477 | 2 |
| xc018_xx12_MET4_METMID | 5000 | 4723 | 2 |
| xc018_xx13_MET5_METMID | 5000 | 4513 | 0 |
| xc018_xx14_MET2_METMID_METTHK | 5000 | 4620 | 1 |
| xc018_xx15_MET3_METMID_METTHK | 5000 | 4618 | 0 |
| xc018_xx16_MET4_METMID_METTHK | 5000 | 4513 | 1 |

Table 6: MOSST Timing fifo_FSM2 200MHz without delay

| Metal Configuration | Arrival | Required Time | Slack |
|---|---|---|---|
| xh018_xx31_MET3_METMID | 10000 | 9357 | 6 |
| xh018_xx33_MET3_METMID_METTHK | 10000 | 9101 | 29 |
| xh018_xx41_MET4_METMID | 10000 | 9357 | 5 |
| xh018_xx43_MET4_METMID_METTHK | 10000 | 9101 | 29 |
| xh018_xx51_MET5_METMID | 10000 | 9102 | 49 |

Table 7: LPMOS Timing fifo_FSM2 100MHz with delay

| Metal Configuration | Arrival | Required Time | Slack |
|---|---|---|---|
| xc018_xx10_MET2_METMID | 10000 | 9228 | 47 |
| xc018_xx11_MET3_METMID | 10000 | 9228 | 20 |
| xc018_xx12_MET4_METMID | 10000 | 9228 | 16 |
| xc018_xx13_MET5_METMID | 10000 | 9228 | 186 |
| xc018_xx14_MET2_METMID_METTHK | 10000 | 9228 | 137 |
| xc018_xx15_MET3_METMID_METTHK | 10000 | 9229 | 33 |
| xc018_xx16_MET4_METMID_METTHK | 10000 | 9229 | 31 |

Table 8: MOSLP Timing fifo_FSM2 100MHz with delay

| Metal Configuration | Arrival | Required Time | Slack |
|---|---|---|---|
| xc018_xx10_MET2_METMID | 10000 | 9472 | 2936 |
| xc018_xx11_MET3_METMID | 10000 | 9473 | 3848 |
| xc018_xx12_MET4_METMID | 10000 | 9472 | 3849 |
| xc018_xx13_MET5_METMID | 10000 | 9481 | 2631 |
| xc018_xx14_MET2_METMID_METTHK | 10000 | 9472 | 3081 |
| xc018_xx15_MET3_METMID_METTHK | 10000 | 9473 | 3655 |
| xc018_xx16_MET4_METMID_METTHK | 10000 | 9473 | 3654 |

Table 9: MOSST Timing fifo_FSM2 100MHz with delay

| Metal Configuration | Arrival | Required Time | Slack |
|---|---|---|---|
| xh018_xx31_MET3_METMID | 3330 | 2926 | 0 |
| xh018_xx33_MET3_METMID_METTHK | 3330 | 2901 | 0 |
| xh018_xx41_MET4_METMID | 3330 | 3012 | 0 |
| xh018_xx43_MET4_METMID_METTHK | 3330 | 3002 | 0 |
| xh018_xx51_MET5_METMID | 3330 | 3001 | 0 |

Table 10: LPMOS Timing fifo_FSM2 300MHz with delay

| Metal Configuration | Arrival | Required Time | Slack |
|---|---|---|---|
| xc018_xx10_MET2_METMID | 3330 | 2953 | 0 |
| xc018_xx11_MET3_METMID | 3330 | 3000 | 0 |
| xc018_xx12_MET4_METMID | 3330 | 3044 | 0 |
| xc018_xx13_MET5_METMID | 3330 | 3000 | 0 |
| xc018_xx14_MET2_METMID_METTHK | 3330 | 2923 | 0 |
| xc018_xx15_MET3_METMID_METTHK | 3330 | 3000 | 0 |
| xc018_xx16_MET4_METMID_METTHK | 3330 | 2950 | 0 |

Table 11: MOSLP Timing fifo_FSM2 300MHz with delay

| Metal Configuration | Arrival | Required Time | Slack |
|---|---|---|---|
| xc018_xx10_MET2_METMID | 3330 | 2753 | 0 |
| xc018_xx11_MET3_METMID | 3330 | 3044 | 0 |
| xc018_xx12_MET4_METMID | 3330 | 2962 | 0 |
| xc018_xx13_MET5_METMID | 3330 | 2984 | 0 |
| xc018_xx14_MET2_METMID_METTHK | 3330 | 2984 | 0 |
| xc018_xx15_MET3_METMID_METTHK | 3330 | 3019 | 0 |
| xc018_xx16_MET4_METMID_METTHK | 3330 | 2984 | 0 |

Table 12: MOSST Timing fifo_FSM2 300MHz with delay

| Metal Configuration | Arrival | Required Time | Slack |
|---|---|---|---|
| xh018_xx31_MET3_METMID | 2040 | 1737 | -704 |
| xh018_xx33_MET3_METMID_METTHK | 2040 | 1736 | -723 |
| xh018_xx41_MET4_METMID | 2040 | 1767 | -710 |
| xh018_xx43_MET4_METMID_METTHK | 2040 | 1767 | -698 |
| xh018_xx51_MET5_METMID | 2040 | 1716 | -695 |

Table 13: LPMOS Timing fifo_FSM2 490MHz with delay

| Metal Configuration | Arrival | Required Time | Slack |
|---|---|---|---|
| xc018_xx10_MET2_METMID | 2040 | 1750 | -431 |
| xc018_xx11_MET3_METMID | 2040 | 1676 | -416 |
| xc018_xx13_MET5_METMID | 2040 | 1746 | -404 |
| xc018_xx14_MET2_METMID_METTHK | 2040 | 1751 | -404 |
| xc018_xx15_MET3_METMID_METTHK | 2040 | 1755 | -403 |
| xc018_xx16_MET4_METMID_METTHK | 2040 | 1786 | -400 |

Table 14: MOSLP Timing fifo_FSM2 490MHz with delay

| Metal Configuration | Arrival | Required Time | Slack |
|---|---|---|---|
| xc018_xx10_MET2_METMID | 2040 | 1790 | 0 |
| xc018_xx11_MET3_METMID | 2040 | 1841 | 0 |
| xc018_xx12_MET4_METMID | 2040 | 1862 | 0 |
| xc018_xx13_MET5_METMID | 2040 | 1867 | 0 |
| xc018_xx14_MET2_METMID_METTHK | 2040 | 1829 | 0 |
| xc018_xx15_MET3_METMID_METTHK | 2040 | 1848 | 0 |
| xc018_xx16_MET4_METMID_METTHK | 2040 | 1826 | 0 |

Table 15: MOSST Timing fifo_FSM2 490MHz with delay

| Metal Configuration | Leakage | Dynamic | Total |
|---|---|---|---|
| xh018_xx31_MET3_METMID | 190.431 | 203185535.479 | 203185725.910 |
| xh018_xx33_MET3_METMID_METTHK | 190.488 | 202867719.429 | 202867909.917 |
| xh018_xx41_MET4_METMID | 190.451 | 203238161.723 | 203238352.175 |
| xh018_xx43_MET4_METMID_METTHK | 190.488 | 202909843.154 | 202910033.642 |
| xh018_xx51_MET5_METMID | 190.449 | 202685382.954 | 202685573.402 |

Table 16: LPMOS Power fifo_FSM2 100MHz without delay

| Metal Configuration | Leakage | Dynamic | Total |
|---|---|---|---|
| xc018_xx10_MET2_METMID | 242.545 | 179224949.115 | 179225191.660 |
| xc018_xx11_MET3_METMID | 242.446 | 178294568.507 | 178294810.953 |
| xc018_xx12_MET4_METMID | 242.393 | 178911951.890 | 178912194.282 |
| xc018_xx13_MET5_METMID | 242.468 | 177865163.906 | 177865406.374 |
| xc018_xx14_MET2_METMID_METTHK | 242.539 | 179022387.152 | 179022629.691 |
| xc018_xx15_MET3_METMID_METTHK | 242.420 | 178588318.018 | 178588560.438 |
| xc018_xx16_MET4_METMID_METTHK | 242.388 | 178622225.728 | 178622468.116 |

Table 17: MOSLP Power fifo_FSM2 100MHz without delay

| Metal Configuration | Leakage | Dynamic | Total |
|---|---|---|---|
| xc018_xx10_MET2_METMID | 671.839 | 186427255.618 | 186427927.457 |
| xc018_xx11_MET3_METMID | 671.120 | 183484953.124 | 183485624.245 |
| xc018_xx12_MET4_METMID | 671.120 | 183548164.120 | 183548835.240 |
| xc018_xx13_MET5_METMID | 670.759 | 182949046.418 | 182949717.177 |
| xc018_xx14_MET2_METMID_METTHK | 670.925 | 183527967.116 | 183528638.041 |
| xc018_xx15_MET3_METMID_METTHK | 670.772 | 183047528.828 | 183048199.599 |
| xc018_xx16_MET4_METMID_METTHK | 670.799 | 183100815.650 | 183101486.449 |

Table 18: MOSST Power fifo_FSM2 100MHz without delay

| Metal Configuration | Leakage | Dynamic | Total |
|---|---|---|---|
| xh018_xx31_MET3_METMID | 180.716 | 175281796.995 | 175281977.711 |
| xh018_xx33_MET3_METMID_METTHK | 179.287 | 175848463.742 | 175848643.029 |
| xh018_xx41_MET4_METMID | 180.254 | 176465843.133 | 176466023.388 |
| xh018_xx43_MET4_METMID_METTHK | 179.264 | 176078540.259 | 176078719.523 |
| xh018_xx51_MET5_METMID | 179.778 | 179775480.835 | 179775660.612 |

Table 19: LPMOS Power fifo_FSM2 200MHz without delay

| Metal Configuration | Leakage | Dynamic | Total |
|---|---|---|---|
| xc018_xx10_MET2_METMID | 239.802 | 145421394.659 | 145421634.460 |
| xc018_xx11_MET3_METMID | 240.180 | 144608337.632 | 144608577.812 |
| xc018_xx12_MET4_METMID | 240.225 | 144000902.441 | 144001142.666 |
| xc018_xx13_MET5_METMID | 239.245 | 147056931.933 | 147057171.178 |
| xc018_xx14_MET2_METMID_METTHK | 240.119 | 144666322.517 | 144666562.636 |
| xc018_xx15_MET3_METMID_METTHK | 240.095 | 145093419.443 | 145093659.538 |
| xc018_xx16_MET4_METMID_METTHK | 240.143 | 144469032.705 | 144469272.848 |

Table 20: MOSLP_Power_fifo_FSM2 200MHz without delay

| Metal Configuration | Leakage | Dynamic | Total |
|---|---|---|---|
| xc018_xx10_MET2_METMID | 684.357 | 143883745.251 | 143884429.608 |
| xc018_xx11_MET3_METMID | 684.513 | 143854459.003 | 143855143.517 |
| xc018_xx12_MET4_METMID | 685.114 | 143894263.448 | 143894948.562 |
| xc018_xx13_MET5_METMID | 683.191 | 143745076.631 | 143745759.821 |
| xc018_xx14_MET2_METMID_METTHK | 685.217 | 143849705.194 | 143850390.411 |
| xc018_xx15_MET3_METMID_METTHK | 685.317 | 143969504.978 | 143970190.294 |
| xc018_xx16_MET4_METMID_METTHK | 684.841 | 143809345.434 | 143810030.276 |

Table 21: MOSST_Power_fifo_FSM2 200MHz without delay

| Metal Configuration | Leakage | Dynamic | Total |
|---|---|---|---|
| xh018_xx31_MET3_METMID | 190.688 | 208724184.167 | 208724374.855 |
| xh018_xx33_MET3_METMID_METTHK | 190.749 | 208386361.715 | 208386552.464 |
| xh018_xx41_MET4_METMID | 190.706 | 208765773.771 | 208765964.477 |
| xh018_xx43_MET4_METMID_METTHK | 190.749 | 208429055.147 | 208429245.896 |
| xh018_xx51_MET5_METMID | 190.719 | 208179057.602 | 208179248.321 |

Table 22: LPMOS Power fifo_FSM2 100MHz with delay

| Metal Configuration | Leakage | Dynamic | Total |
|---|---|---|---|
| xc018_xx10_MET2_METMID | 242.267 | 185435463.877 | 185435706.144 |
| xc018_xx11_MET3_METMID | 242.177 | 183953056.525 | 183953298.702 |
| xc018_xx12_MET4_METMID | 242.107 | 184729182.947 | 184729425.054 |
| xc018_xx13_MET5_METMID | 242.227 | 183435639.517 | 183435881.744 |
| xc018_xx14_MET2_METMID_METTHK | 242.196 | 184686381.022 | 184686623.218 |
| xc018_xx15_MET3_METMID_METTHK | 242.128 | 184332198.315 | 184332440.442 |
| xc018_xx16_MET4_METMID_METTHK | 242.130 | 184375841.673 | 184376083.803 |

Table 23: MOSLP Power fifo_FSM2 100MHz with delay

| Metal Configuration | Leakage | Dynamic | Total |
|---|---|---|---|
| xc018_xx10_MET2_METMID | 671.860 | 189097045.290 | 189097717.149 |
| xc018_xx11_MET3_METMID | 671.327 | 188403244.287 | 188403915.614 |
| xc018_xx12_MET4_METMID | 671.411 | 191695068.913 | 191695740.324 |
| xc018_xx13_MET5_METMID | 671.069 | 191088332.018 | 191089003.087 |
| xc018_xx14_MET2_METMID_METTHK | 671.322 | 191734681.526 | 191735352.848 |
| xc018_xx15_MET3_METMID_METTHK | 671.068 | 188012062.359 | 188012733.427 |
| xc018_xx16_MET4_METMID_METTHK | 671.061 | 188051010.352 | 188051681.413 |

Table 24: MOSST Power fifo_FSM2 100MHz with delay

| Metal Configuration | Leakage | Dynamic | Total |
|---|---|---|---|
| xh018_xx31_MET3_METMID | 197.412 | 677582479.479 | 677582676.892 |
| xh018_xx33_MET3_METMID_METTHK | 197.563 | 687321079.475 | 687321277.037 |
| xh018_xx41_MET4_METMID | 198.022 | 679800316.824 | 679800514.846 |
| xh018_xx43_MET4_METMID_METTHK | 197.532 | 677360403.347 | 677360600.879 |
| xh018_xx51_MET5_METMID | 197.349 | 681871723.316 | 681871920.665 |

Table 25: LPMOS Power fifo_FSM2 300MHz with delay

| Metal Configuration | Leakage | Dynamic | Total |
|---|---|---|---|
| xc018_xx10_MET2_METMID | 241.005 | 602666714.792 | 602666955.797 |
| xc018_xx11_MET3_METMID | 241.300 | 587025414.892 | 587025656.192 |
| xc018_xx12_MET4_METMID | 241.703 | 581760058.850 | 581760300.553 |
| xc018_xx13_MET5_METMID | 240.484 | 574629499.099 | 574629739.582 |
| xc018_xx14_MET2_METMID_METTHK | 240.690 | 593096073.103 | 593096313.793 |
| xc018_xx15_MET3_METMID_METTHK | 241.298 | 576237550.804 | 576237792.103 |
| xc018_xx16_MET4_METMID_METTHK | 242.048 | 573654990.185 | 573655232.233 |

Table 26: MOSLP Power fifo_FSM2 300MHz with delay

| Metal Configuration | Leakage | Dynamic | Total |
|---|---|---|---|
| xc018_xx10_MET2_METMID | 698.869 | 558739806.635 | 558740505.504 |
| xc018_xx11_MET3_METMID | 695.334 | 556104682.431 | 556105377.764 |
| xc018_xx12_MET4_METMID | 695.688 | 557859677.026 | 557860372.715 |
| xc018_xx13_MET5_METMID | 694.334 | 554591021.872 | 554591716.206 |
| xc018_xx14_MET2_METMID_METTHK | 697.529 | 557950770.337 | 557951467.865 |
| xc018_xx15_MET3_METMID_METTHK | 696.610 | 556721213.463 | 556721910.073 |
| xc018_xx16_MET4_METMID_METTHK | 695.256 | 557640926.249 | 557641621.505 |

Table 27: MOSST Power fifo_FSM2 300MHz with delay

| Metal Configuration | Leakage | Dynamic | Total |
|---|---|---|---|
| xh018_xx31_MET3_METMID | 261.713 | 1448446587.004 | 1448446848.717 |
| xh018_xx33_MET3_METMID_METTHK | 258.896 | 1439149595.924 | 1439149854.820 |
| xh018_xx41_MET4_METMID | 264.140 | 1448661184.294 | 1448661448.433 |
| xh018_xx43_MET4_METMID_METTHK | 255.188 | 1426753837.242 | 1426754092.430 |
| xh018_xx51_MET5_METMID | 261.318 | 1453849138.751 | 1453849400.069 |

Table 28: LPMOS Power fifo_FSM2 490MHz with delay

| Metal Configuration | Leakage | Dynamic | Total |
|---|---|---|---|
| xc018_xx10_MET2_METMID | 290.363 | 1258801431.775 | 1258801722.138 |
| xc018_xx11_MET3_METMID | 282.218 | 1187433811.353 | 1187434093.571 |
| xc018_xx12_MET4_METMID | 282.238 | 1209002868.574 | 1209003150.812 |
| xc018_xx13_MET5_METMID | 287.752 | 1248255705.441 | 1248255993.193 |
| xc018_xx14_MET2_METMID_METTHK | 290.150 | 1244974923.197 | 1244975213.348 |
| xc018_xx15_MET3_METMID_METTHK | 278.612 | 1214219715.034 | 1214219993.646 |
| xc018_xx16_MET4_METMID_METTHK | 283.003 | 1198988106.355 | 1198988389.357 |

Table 29: MOSLP Power fifo_FSM2 490MHz with delay

| Metal Configuration | Leakage | Dynamic | Total |
|---|---|---|---|
| xc018_xx10_MET2_METMID | 813.544 | 958922427.528 | 958923241.072 |
| xc018_xx11_MET3_METMID | 818.323 | 956583006.600 | 956583824.923 |
| xc018_xx12_MET4_METMID | 812.287 | 958984858.300 | 958985670.587 |
| xc018_xx13_MET5_METMID | 790.621 | 941948232.152 | 941949022.772 |
| xc018_xx14_MET2_METMID_METTHK | 814.065 | 956136488.495 | 956137302.560 |
| xc018_xx15_MET3_METMID_METTHK | 793.050 | 940944730.370 | 940945523.420 |
| xc018_xx16_MET4_METMID_METTHK | 791.593 | 945695928.688 | 945696720.281 |

Table 30: MOSST Power fifo_FSM2 490MHz with delay

| Metal Configuration | Instances | Area |
|---|---|---|
| xh018_xx31_MET3_METMID | 28746 | 958991.443 |
| xh018_xx33_MET3_METMID_METTHK | 28845 | 958899.211 |
| xh018_xx41_MET4_METMID | 28743 | 959040.634 |
| xh018_xx43_MET4_METMID_METTHK | 28845 | 958899.211 |
| xh018_xx51_MET5_METMID | 28839 | 958711.673 |

Table 31: LPMOS Area fifo_FSM2 100MHz without delay

| Metal Configuration | Instances | Area |
|---|---|---|
| xc018_xx11_MET3_METMID | 28752 | 958966.848 |
| xc018_xx12_MET4_METMID | 28752 | 959182.056 |
| xc018_xx13_MET5_METMID | 28769 | 959034.485 |
| xc018_xx14_MET2_METMID_METTHK | 28747 | 959474.124 |
| xc018_xx15_MET3_METMID_METTHK | 28753 | 959268.139 |
| xc018_xx16_MET4_METMID_METTHK | 28752 | 959169.758 |

Table 32: MOSLP Area fifo_FSM2 100MHz without delay

| Metal Configuration | Instances | Area |
|---|---|---|
| xc018_xx10_MET2_METMID | 28695 | 957654.079 |
| xc018_xx11_MET3_METMID | 28816 | 957334.342 |
| xc018_xx12_MET4_METMID | 28816 | 957334.342 |
| xc018_xx13_MET5_METMID | 28803 | 957048.422 |
| xc018_xx14_MET2_METMID_METTHK | 28804 | 957777.055 |
| xc018_xx15_MET3_METMID_METTHK | 28805 | 957395.830 |
| xc018_xx16_MET4_METMID_METTHK | 28806 | 957411.202 |

Table 33: MOSST Area fifo_FSM2 100MHz without delay

| Metal Configuration | Instances | Area |
|---|---|---|
| xh018_xx31_MET3_METMID | 29623 | 974932.207 |
| xh018_xx33_MET3_METMID_METTHK | 29707 | 974507.940 |
| xh018_xx41_MET4_METMID | 29661 | 974741.594 |
| xh018_xx43_MET4_METMID_METTHK | 29652 | 974280.434 |
| xh018_xx51_MET5_METMID | 29580 | 974384.964 |

Table 34: LPMOS Area fifo_FSM2 200MHz without delay

| Metal Configuration | Instances | Area |
|---|---|---|
| xc018_xx10_MET2_METMID | 29460 | 970142.292 |
| xc018_xx11_MET3_METMID | 29448 | 968266.908 |
| xc018_xx12_MET4_METMID | 29464 | 968525.158 |
| xc018_xx13_MET5_METMID | 29372 | 970286.789 |
| xc018_xx14_MET2_METMID_METTHK | 29388 | 968405.256 |
| xc018_xx15_MET3_METMID_METTHK | 29397 | 968371.438 |
| xc018_xx16_MET4_METMID_METTHK | 29319 | 967132.454 |

Table 35: MOSLP Area fifo_FSM2 200MHz without delay

| Metal Configuration | Instances | Area |
|---|---|---|
| xc018_xx10_MET2_METMID | 28789 | 959200.502 |
| xc018_xx11_MET3_METMID | 28775 | 959142.089 |
| xc018_xx12_MET4_METMID | 28779 | 959471.050 |
| xc018_xx13_MET5_METMID | 28774 | 958511.837 |
| xc018_xx14_MET2_METMID_METTHK | 28804 | 959495.645 |
| xc018_xx15_MET3_METMID_METTHK | 28823 | 959351.148 |
| xc018_xx16_MET4_METMID_METTHK | 28821 | 959114.419 |

Table 36: MOSST Area fifo_FSM2 200MHz without delay

| Metal Configuration | Instances | Area |
|---|---|---|
| xh018_xx31_MET3_METMID | 28706 | 958748.566 |
| xh018_xx33_MET3_METMID_METTHK | 28802 | 958776.235 |
| xh018_xx41_MET4_METMID | 28705 | 958763.938 |
| xh018_xx43_MET4_METMID_METTHK | 28810 | 958819.277 |
| xh018_xx51_MET5_METMID | 28800 | 958717.822 |

Table 37: LPMOS Area fifo_FSM2 100MHz with delay

| Metal Configuration | Instances | Area |
|---|---|---|
| xc018_xx10_MET2_METMID | 28708 | 959452.603 |
| xc018_xx11_MET3_METMID | 28739 | 958979.146 |
| xc018_xx12_MET4_METMID | 28739 | 959154.386 |
| xc018_xx13_MET5_METMID | 28747 | 958954.550 |
| xc018_xx14_MET2_METMID_METTHK | 28715 | 959292.734 |
| xc018_xx15_MET3_METMID_METTHK | 28735 | 959206.651 |
| xc018_xx16_MET4_METMID_METTHK | 28735 | 959197.428 |

Table 38: MOSLP Area fifo_FSM2 100MHz with delay

| Metal Configuration | Instances | Area |
|---|---|---|
| xc018_xx10_MET2_METMID | 28670 | 957718.642 |
| xc018_xx11_MET3_METMID | 28806 | 957445.020 |
| xc018_xx12_MET4_METMID | 28795 | 957417.350 |
| xc018_xx13_MET5_METMID | 28791 | 957128.357 |
| xc018_xx14_MET2_METMID_METTHK | 28776 | 957730.939 |
| xc018_xx15_MET3_METMID_METTHK | 28789 | 957315.895 |
| xc018_xx16_MET4_METMID_METTHK | 28789 | 957315.895 |

Table 39: MOSST Area fifo_FSM2 100MHz with delay

| Metal Configuration | Instances | Area |
|---|---|---|
| xh018_xx31_MET3_METMID | 32321 | 1014013.980 |
| xh018_xx33_MET3_METMID_METTHK | 32464 | 1018524.125 |
| xh018_xx41_MET4_METMID | 32463 | 1015852.471 |
| xh018_xx43_MET4_METMID_METTHK | 32306 | 1015677.230 |
| xh018_xx51_MET5_METMID | 32409 | 1015932.406 |

Table 40: LPMOS Area fifo_FSM2 300MHz with delay

| Metal Configuration | Instances | Area |
|---|---|---|
| xc018_xx10_MET2_METMID | 31470 | 1017921.542 |
| xc018_xx11_MET3_METMID | 31514 | 1008756.756 |
| xc018_xx12_MET4_METMID | 31466 | 1010386.188 |
| xc018_xx13_MET5_METMID | 31542 | 1004701.622 |
| xc018_xx14_MET2_METMID_METTHK | 31513 | 1014093.914 |
| xc018_xx15_MET3_METMID_METTHK | 31327 | 1007013.571 |
| xc018_xx16_MET4_METMID_METTHK | 31350 | 1006051.284 |

Table 41: MOSLP Area fifo_FSM2 300MHz with delay

| Metal Configuration | Instances | Area |
|---|---|---|
| xc018_xx10_MET2_METMID | 29256 | 968808.002 |
| xc018_xx11_MET3_METMID | 29315 | 967578.242 |
| xc018_xx12_MET4_METMID | 29334 | 967608.986 |
| xc018_xx13_MET5_METMID | 29334 | 967489.085 |
| xc018_xx14_MET2_METMID_METTHK | 29282 | 968500.562 |
| xc018_xx15_MET3_METMID_METTHK | 29374 | 968300.726 |
| xc018_xx16_MET4_METMID_METTHK | 29307 | 967630.507 |

Table 42: MOSST Area fifo_FSM2 300MHz with delay

| Metal Configuration | Instances | Area |
|---|---|---|
| xh018_xx31_MET3_METMID | 44821 | 1294073.374 |
| xh018_xx33_MET3_METMID_METTHK | 44684 | 1281994.056 |
| xh018_xx41_MET4_METMID | 44853 | 1302577.164 |
| xh018_xx43_MET4_METMID_METTHK | 44516 | 1280327.731 |
| xh018_xx51_MET5_METMID | 44894 | 1283669.604 |

Table 43: LPMOS Area fifo_FSM2 490MHz with delay

| Metal Configuration | Instances | Area |
|---|---|---|
| xc018_xx10_MET2_METMID | 42619 | 1278172.577 |
| xc018_xx11_MET3_METMID | 41314 | 1232016.610 |
| xc018_xx12_MET4_METMID | 41451 | 1236499.085 |
| xc018_xx13_MET5_METMID | 43037 | 1264516.092 |
| xc018_xx14_MET2_METMID_METTHK | 42702 | 1270234.476 |
| xc018_xx15_MET3_METMID_METTHK | 41524 | 1226910.031 |
| xc018_xx16_MET4_METMID_METTHK | 42241 | 1231294.126 |

Table 44: MOSLP Area fifo_FSM2 490MHz with delay

| Metal Configuration | Instances | Area |
|---|---|---|
| xc018_xx10_MET2_METMID | 31846 | 1016368.970 |
| xc018_xx11_MET3_METMID | 32022 | 1017623.326 |
| xc018_xx12_MET4_METMID | 31803 | 1016983.850 |
| xc018_xx13_MET5_METMID | 31931 | 1009177.949 |
| xc018_xx14_MET2_METMID_METTHK | 31888 | 1017211.356 |
| xc018_xx15_MET3_METMID_METTHK | 32097 | 1012894.898 |
| xc018_xx16_MET4_METMID_METTHK | 32094 | 1012984.056 |

Table 45: MOSST Area fifo_FSM2 490MHz with delay