CND 111: Introduction to Digital Design

**Course Project**

**Implementation of DSP Slice**

**Section #:** 14

**Submitted by Group 10:**

|  |  |
| --- | --- |
| **Student Name** | **ID** |
| Rahma Aly Mahmoud Magdy Aly | V23010308 |
| Mariz Gad Atta | V23009927 |
| Mariz Sabry Zaky Attia | V23010580 |

**Supervisor: Dr. Ahmed Saeed**

**Table of Contents**

[Chapter 1: Introduction 2](#_Toc153299221)

[1.1. Basic DSP Slice Components and Functionality 2](#_Toc153299224)

[1.2. Applications 2](#_Toc153299233)

[Chapter 2: System Design/Implementation 3](#_Toc153299234)

[2.1. Dual Registers (A, B) 3](#_Toc153299236)

[2.2. Pre-Adder 4](#_Toc153299237)

[2.3. Multiplier 5](#_Toc153299238)

[2.4. ALU (adder/subtractor and logic unit) 6](#_Toc153299239)

[2.5. Pattern Detector 9](#_Toc153299240)

[Chapter 3: Verification 9](#_Toc153299241)

[Test 1: Multiplication Operation 9](#_Toc153299243)

[Test 2: Arithmetic/Logic Operations 12](#_Toc153299244)

[Chapter 4: RTL Schematic and Synthesis Result 14](#_Toc153299246)

[4.1. Design Schematic 14](#_Toc153299248)

[4.1.1. Full System 15](#_Toc153299252)

[4.1.2. Multiplier (when USE\_MULT = “multiply”) 15](#_Toc153299253)

[4.1.3. ALU (when USE\_MULT = “multiply”) 15](#_Toc153299254)

[4.1.4. Pre-adder (when USE\_DPORT = 1) 16](#_Toc153299255)

[4.1.5. Dual Register A 17](#_Toc153299256)

[4.1.6. ALU (when USE\_MULT = “none”) 17](#_Toc153299257)

[4.2. Synthesis Result 18](#_Toc153299258)

[4.2.1. Resource Usage 18](#_Toc153299260)

[4.2.2. Resource utilization by entity 18](#_Toc153299261)

[4.2.3. Power Consumption 19](#_Toc153299262)

[4.2.4. Maximum Clock 19](#_Toc153299263)

[Conclusion 20](#_Toc153299264)

# Chapter 1: Introduction



FPGAs are efficient for digital signal processing (DSP) applications. DSP applications use many binary multipliers and accumulators that are best implemented in dedicated DSP slices. 7 Series FPGAs have low-power DSP slices, combining high speed with small size while retaining system design flexibility.

## Basic DSP Slice Components and Functionality: The DSP48E1 slice supports many independent functions including multiply, multiply accumulate (MACC), multiply add, three-input add, widebus multiplexing, magnitude comparator, bitwise logic functions, pattern detect, and wide counter. These functions can be implemented through the basic components:

## Inputs: There are 4 inputs to the slice: A (30 bits), B (18 bits), C (48 bits), D (25 bits).

## Pre-Adder: It takes 2 inputs A and D, adds or subtracts them according to control signals to provide one of the 2 inputs to the Multiplier.

## 25 x 18 two’s Complement Multiplier: It takes 2 asymmetric two’s complement inputs, multiplies them and outputs 2 partial products.

## ALU Unit: It performs 3-input Arithmetic and logic operations according to control signals.

## Pattern Detector: Equality check at the output of the ALU unit to detect if the output matches specific pattern.

## Optional Pipelining: Attributes are set to choose whether the pipelining registers will be used. It helps saving power and increasing performance.

## Outputs: There is one output to the slice: P (48 bits).

## Dedicated Buses for Cascading: It can be used for larger Multipliers and larger Pre-Adders.

## Applications: The DSP slices are used for many applications in Digital Signal Processing as it can be used to implement various types of filters, such as low pass, high pass and band pass filters that are used for audio equalization, image filtering, biomedical signals filtering. In addition to, adaptive filters that can be used in filtering radar signals and sensor data in industrial automation application. The cascade capabilities of the DSP slice are extremely efficient at implementing highspeed pipelined filters. Additionally, They enhance the speed and efficiency of many applications beyond digital signal processing, such as wide dynamic bus shifters, memory address generators, wide bus multiplexers, and memory-mapped I/O registers.

# Chapter 2: System Design/Implementation

A diagram of a machine

Description automatically generatedThe DSP slice, figure 1, consists of three main components: a 25-bit pre-adder, a 25-bit by 18-bit two’s complement multiplier followed by a three-input adder/subtractor or two-input logic unit. The inputs to the slice can be registered one or two times (pipelined), and the inputs to the ALU unit are supplied through a number of multiplexers. The pattern detector uses the output of the ALU unit (P) and checks for equality which a certain pattern.

Figure : DSP48E1 Slice block diagram



## Dual Registers (A, B)

The A and B input ports can have 0,1, or 2 pipeline stages in its datapath, figure 2. The different pipe stages are set using attributes. Attributes control the number of used pipeline stages, where:

* Attribute AREG and BREG are used to select the number of pipeline stages for A and B direct inputs. Both have a default value of 1 (1 pipeline stage) which indicates that the registers A2 and B2 are the ones to be used.
* Attributes ACASCREG and BCASCREG select the number of pipeline stages in the ACOUT and BCOUT cascade datapath. The value of these attributes must be less than or equal to AREG and BREG respectively.

The values used as input to the dual registers can either be direct (use the A or B input directly) or cascaded (use ACIN / BCIN). For the B Dual register, the “INMODE” input selects the value of BMULT, which is the value to be supplied to the multiplier input. The effect of the attributes and INMODE value of register B are displayed in the following tables, table 1-2.

For the A register, the “INMODE” input controls the value of the input A supplied to the pre-adder, it either supplies the value of A as it is or provides zero to the pre-adder.

*A diagram of a circuit

Description automatically generatedTable 1: Description of used attributes in the dual registers*

Figure : Dual Register B block diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | BREG | BCASCREG | INMODE[4] | B\_INPUT |
| Function | Sets the number of pipeline registers for input B | Sets the number of pipeline registers for output COUTB | Sets the value of BMULT | Selects the used input |
| Possible values | 0,1.2 | 0,1,2 | 0,1 | “Direct”, “Cascade” |
| Notes | BREG = 0, no pipelining.  BREG = 1, register B2 is used. | Must be less than or equal to BREG. | 0: B2\_MUX\_OUT.  1: B1. | “Direct”: use input B.  “Cascade”: use input BCIN |

Table 2: the effect of the attribute’s values of BCOUT, XMUX and BMULT

|  |  |  |  |
| --- | --- | --- | --- |
|  | BCOUT | XMUX | BMULT |
| BREG = BCASCREG = 1  INMODE = 1 | B2 | B2 | B1 |
| BREG = BCASCREG = 1 or 2  INMODE = 0 | B2 | B2 | B2 |
| BREG = 2  BCASCREG = 1  INMODE = 1 | B1 | B2 | B1 |
| BREG = 0  BCASCREG = 0  INMODE = 1 | B2 (B from input) | B (input) | B1 |

## Pre-Adder

It either adds or subtracts the inputs A and D, and its functionality is controlled by the attribute USE\_DPORT and the input “INMODE”. If the pre-adder, figure 3, is not used the value of USE\_DPORT is set to zero.

Hence, if USE\_DPORT is set to zero, the value of multiplier A port (AMULT) will either be zero or A (input) depending on the value of “INMODE”. Otherwise, the value of AMULT will follow the following table, where the pre-adder subtracts or adds the inputs based on the value of the third bit of the input “INMODE” (INMODE[3] = 0, the pre-adder adds the inputs else it subtracts them).

The pre-adder can also be bypassed, making D the new input path to the multiplier. When the D path is not used, the output of the A pipeline can be negated prior to driving the multiplier.

A diagram of a computer

Description automatically generatedTable 3: AMULT (Pre-adder output) value based on INMODE value

Figure : Dual A, D, Pre-adder logic

|  |  |  |  |
| --- | --- | --- | --- |
| INMODE[3] | INMODE[2] | INMODE[1] | AMULT |
| 0 | 0 | 0 | A |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | D + A |
| 0 | 1 | 1 | D |
| 1 | 0 | 0 | - A |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | D – A |
| 1 | 1 | 1 | D |

## Multiplier

The 2’s Complement Multiplier takes 2 asymmetric inputs, the first 25-bit input is from the pre-adder and the second 18-bit input is from the B register. The outputs are two 43-bit partial products. The two partial products together give an 86-bit result at the output of the multiplier. Cascading of multipliers to achieve larger products is supported with a 17-bit, right-shifted, cascaded output bus. An optional pipeline register (MREG) can be used for the output of the multiplier.

To get the 2 partial products, 2 multiplication operations are needed:

* **18 bits by 18 bits:** The 18-bit input is multiplied by the least significant 18 bits from the 25-bit input.
* **25 bits by 18 bits:** The 18-bit input is multiplied by the most significant 7 bits from the 25-bit input concatenated with 18 zeros.

The result of the multiplier is obtained from the ALU unit after sign extending the 2 partial products to 48 bits in the multiplixers then adding them in the ALU. When the multiplier is used, the ALU becomes 2-input adder. When using two-input logic unit, the multiplier cannot be used.

A diagram of a diagram

Description automatically generated

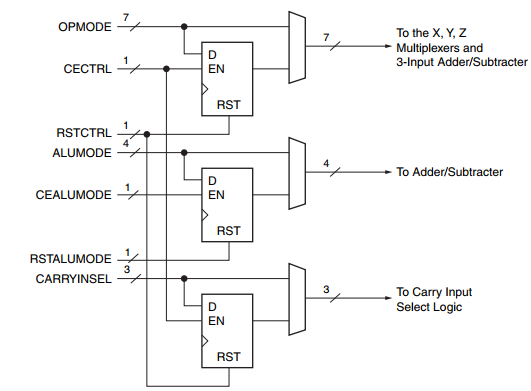
Figure : Multiplier block diagram with optional M Register

Table 4: Description of used attributes in the Multiplier

|  |  |  |  |
| --- | --- | --- | --- |
| Attributes | Description | Settings | Importance |
| USEMULT | Selects usage of the multiplier. | MULTIPLY (Default), NONE | * Saving power when the first stage multiplier is not being used (multiplier is bypassed). * To save the 48-bit 2’s complement operands. |
| MREG | Selects the number of M pipeline registers that are enabled by CEM, and reset synchronously by RSTM. | 0, 1(Default) | * Using the register provides increased performance with an increase of one clock latency and it saves power. |

## ALU (adder/subtractor and logic unit)

The data inputs to the adder/subtracter are selected by the OPMODE and the CARRYINSEL signals. The ALUMODE signals choose the function implemented in the adder/subtracter. Thus, the OPMODE, ALUMODE, and CARRYINSEL signals together determine the functionality of the embedded adder/subtracter/logic unit.

Figure OPMODE, ALUMODE, CARRYINSEL Port logic

The ALU inputs are the outputs of MUX X, MUX Y, MUX Z and Cin MUX

Table X MUX

|  |  |  |  |
| --- | --- | --- | --- |
| **OPMODE [6:4]** | **OPMODE [3:2]** | **OPMODE [1:0]** | **X MUX output** |
| Xxx | Xx | ‘b00 | O |
| Xxx | ‘b01 | ‘b01 | M |
| xxx | Xx | ‘b10 | P |
| xxx | xx | ‘b11 | A:B |

Table Y MUX

|  |  |  |  |
| --- | --- | --- | --- |
| **OPMODE [6:4]** | **OPMODE [3:2]** | **OPMODE [1:0]** | **Y MUX output** |
| Xxx | ‘b00 | xx | O |
| Xxx | ‘b01 | ‘b01 | M |
| xxx | ‘b10 | Xx | 48’FFFFFFFFFFF |
| xxx | ‘b11 | xx | C |

Table Z MUX

|  |  |  |  |
| --- | --- | --- | --- |
| **OPMODE [6:4]** | **OPMODE [3:2]** | **OPMODE [1:0]** | **Z MUX output** |
| ‘b000 | Xx | Xx | O |
| ‘b001 | xx | Xx | PCIN |
| ‘b010 | Xx | Xx | P |
| ‘b011 | Xx | xx | C |
| ‘b100 | ‘b10 | ‘b00 | P |
| ‘b101 | Xx | Xx | 17 bit shift PCIN |
| ‘b110 | Xx | Xx | 17 bit shift P |
| ‘b111 | Xx | xx | X |

Table CIN MUX

|  |  |
| --- | --- |
| **CARRYINSEL** | **Cin** |
| ‘b000 | CARRYIN |
| ‘b001 | ~PCIN[47] |
| ‘b010 | CARYYCASIN |
| ‘b011 | PCIN[47] |
| ‘b100 | CARRYCASCOUT |
| ‘b101 | ~P[47] |
| ‘b110 | A[24] XNOR B[17] |
| ‘b111 | P[47] |

The ALU outputs are the P (the result), carryout and multiplication sign out

The ALU has two modes which are controlled using the USE\_MULT attribute:

* When USE\_MULT is “none”: this means that the multiplier is not working and the ALU works as three inputs ALU which performs arithmetic operations or two input logic units which performs logical operations based on the ALUMODE and OPMODE inputs

Table Three input ALU Operation

|  |  |  |
| --- | --- | --- |
| **OPMODE [6:0]** | **ALUMODE [3:0]** | **Operation** |
| Any legal OPMODE | ‘b0000 | Z + X + Y + CIN |
| Any legal OPMODE | ‘b0011 | Z – (X + Y + CIN) |
| Any legal OPMODE | ‘b0001 | –Z + (X + Y + CIN) – 1 =  not (Z) + X + Y + CIN |
| Any legal OPMODE | ‘b0010 | not (Z + X + Y + CIN) =  –Z – X – Y – CIN - 1 |

Table Two inputs logic unit operation

|  |  |  |
| --- | --- | --- |
| **OPMODE [3:2]** | **ALUMODE [3:0]** | **Operation** |
| ‘b00 | ‘b0100 | X XOR Z |
| ‘b00 | ‘b0101 | X XNOR Z |
| ‘b00 | ‘b0110 | X XNOR Z |
| ‘b00 | ‘b0111 | X XOR Z |
| ‘b00 | ‘b1100 | X AND Z |
| ‘b00 | ‘b1101 | X AND (NOT Z) |
| ‘b00 | ‘b1110 | X NAND Z |
| ‘b00 | ‘b1111 | (NOT X) OR Z |
| ‘b10 | ‘b0100 | X XNOR Z |
| ‘b10 | ‘b0101 | X XOR Z |
| ‘b10 | ‘b0110 | X XOR Z |
| ‘b10 | ‘b0111 | X XNOR Z |
| ‘b10 | ‘b1100 | X OR Z |
| ‘b10 | ‘b1101 | X OR (NOT Z) |
| ‘b10 | ‘b1110 | X NOR Z |
| ‘b10 | ‘b1111 | (NOT X) AND Z |

* When USE\_MULT is “MULTIPLY”:

The multiplier is working and the ALU works as adder

## Pattern Detector

A pattern detector detects if the output P matches a specified pattern or the complement of the pattern. The PATTERNDETECT (PD) output goes High if the output of the adder matches a set pattern, it computes (~ (P ^ pattern) | mask) and then ANDs the results to a single output bit.

The PATTERNBDETECT (PBD) output goes High if the output of the adder matches the complement of the set pattern and computes the same expression but with the complement of the pattern.

A mask field can also be used to hide certain bit locations in the pattern detector. The pattern and the mask fields can each come from a distinct 48-bit configuration attribute called PATTERN and MASK respectively or from the (registered) C input.

A diagram of a mask

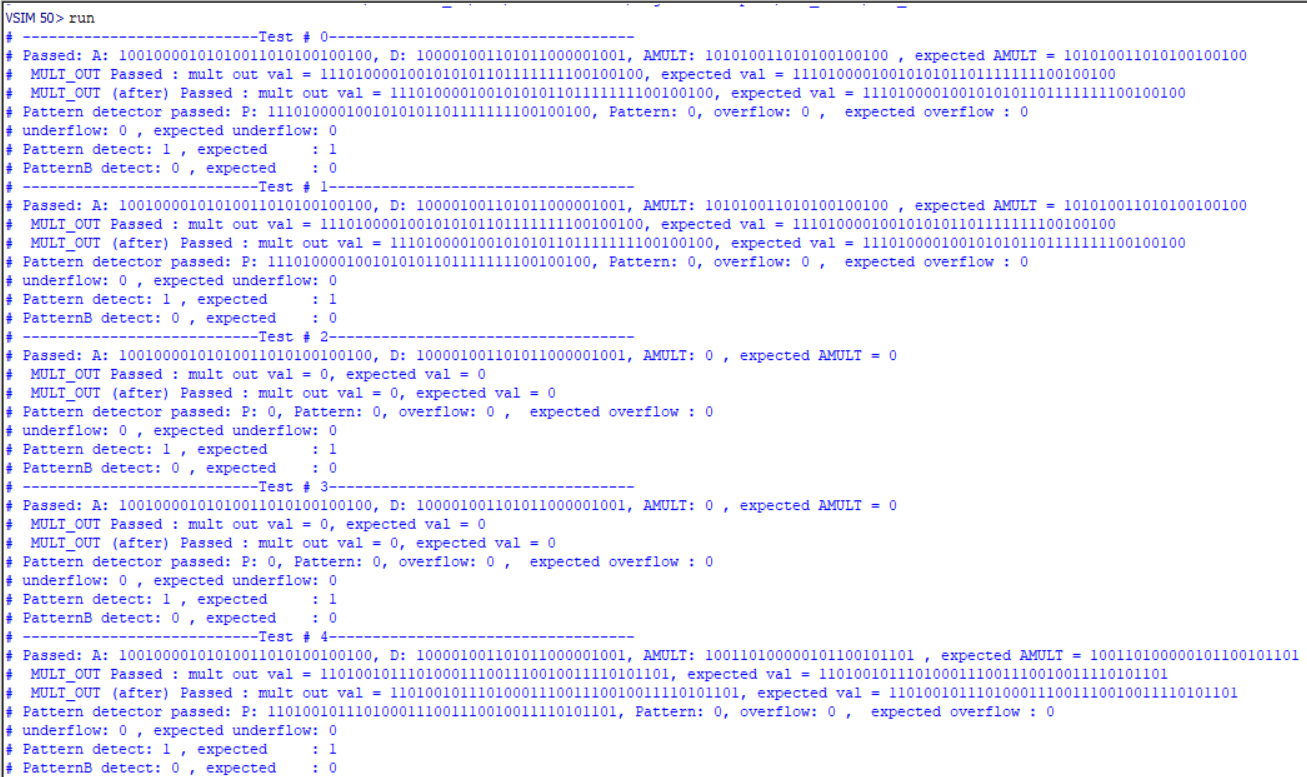
Description automatically generatedThe pattern detector also provides 2 flags: UNDERFLOW and OVERFLOW to determine if the operation in the DSP48E1 slice has overflowed beyond the P[N] bit where N is between 1 and 46. The P register must be enabled while using overflow and underflow ports. The overflow and underflow bits are implemented using an AND gate as shown in figure x.

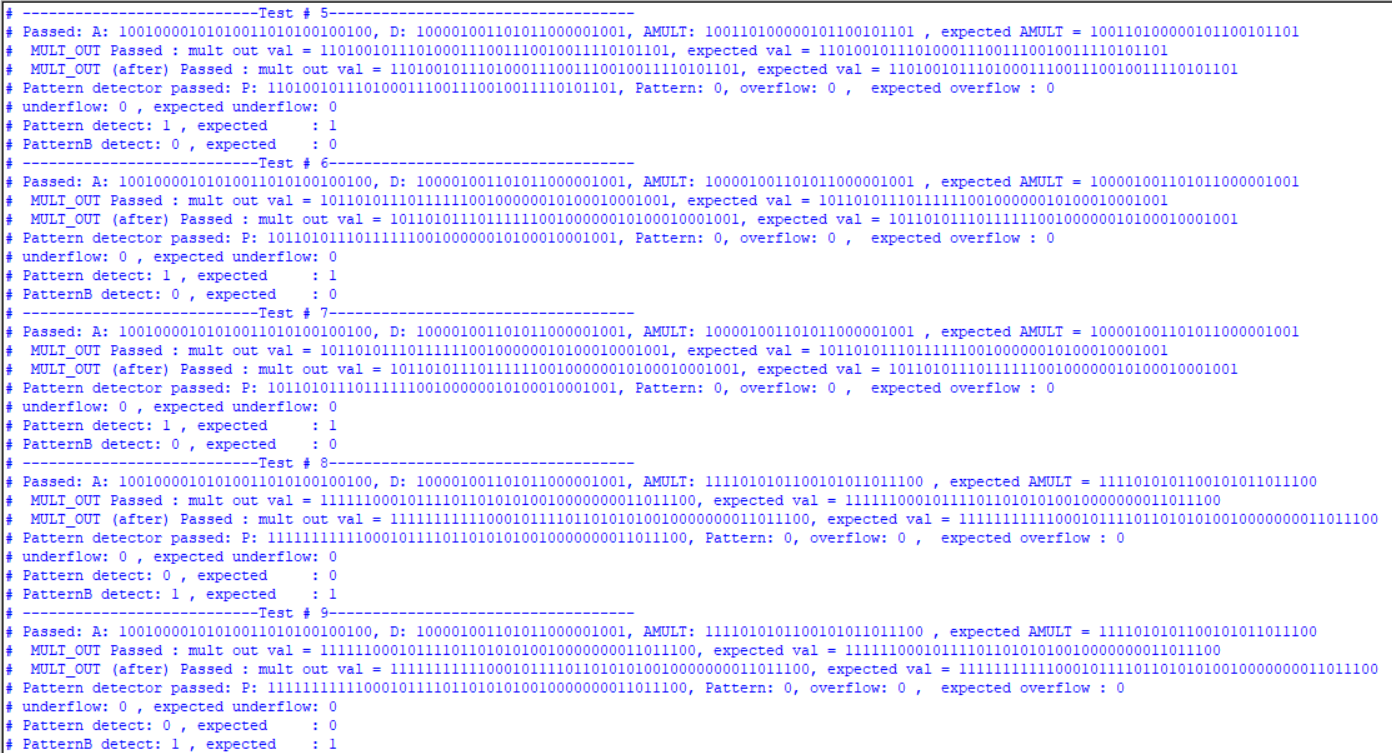
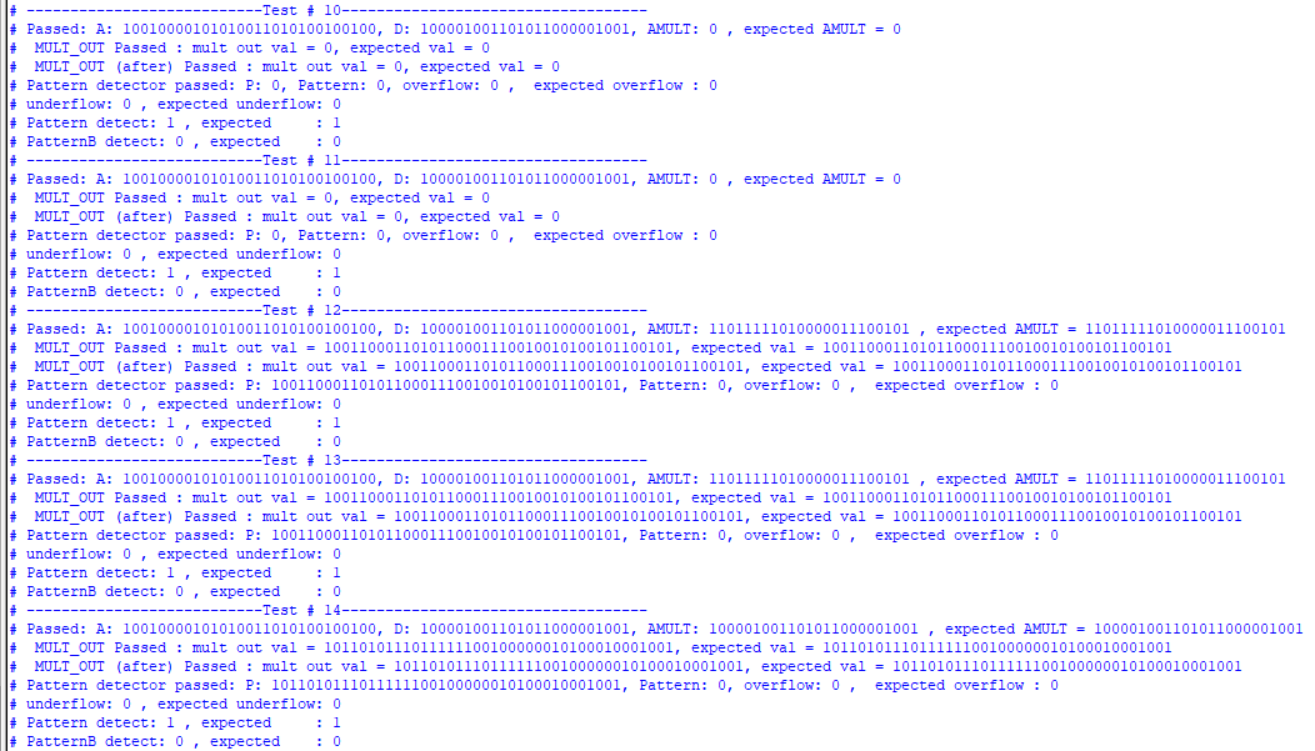
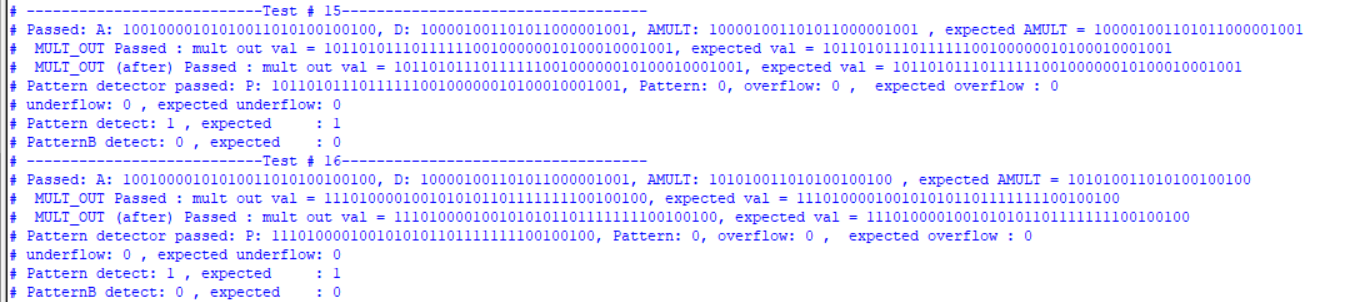
# Chapter 3: Verification

To verify our design, we have started by testing each of the main blocks by itself (unit test) and then we tested the whole system after integrating all the components by making sure that both the multiplication and the arithmetic/logic operations of the slice, including the pattern detection, are functioning properly.



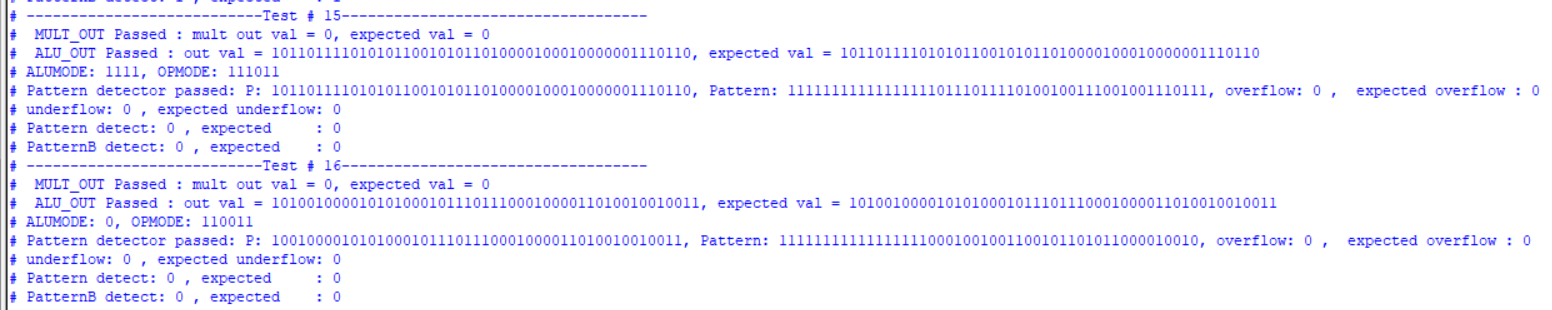
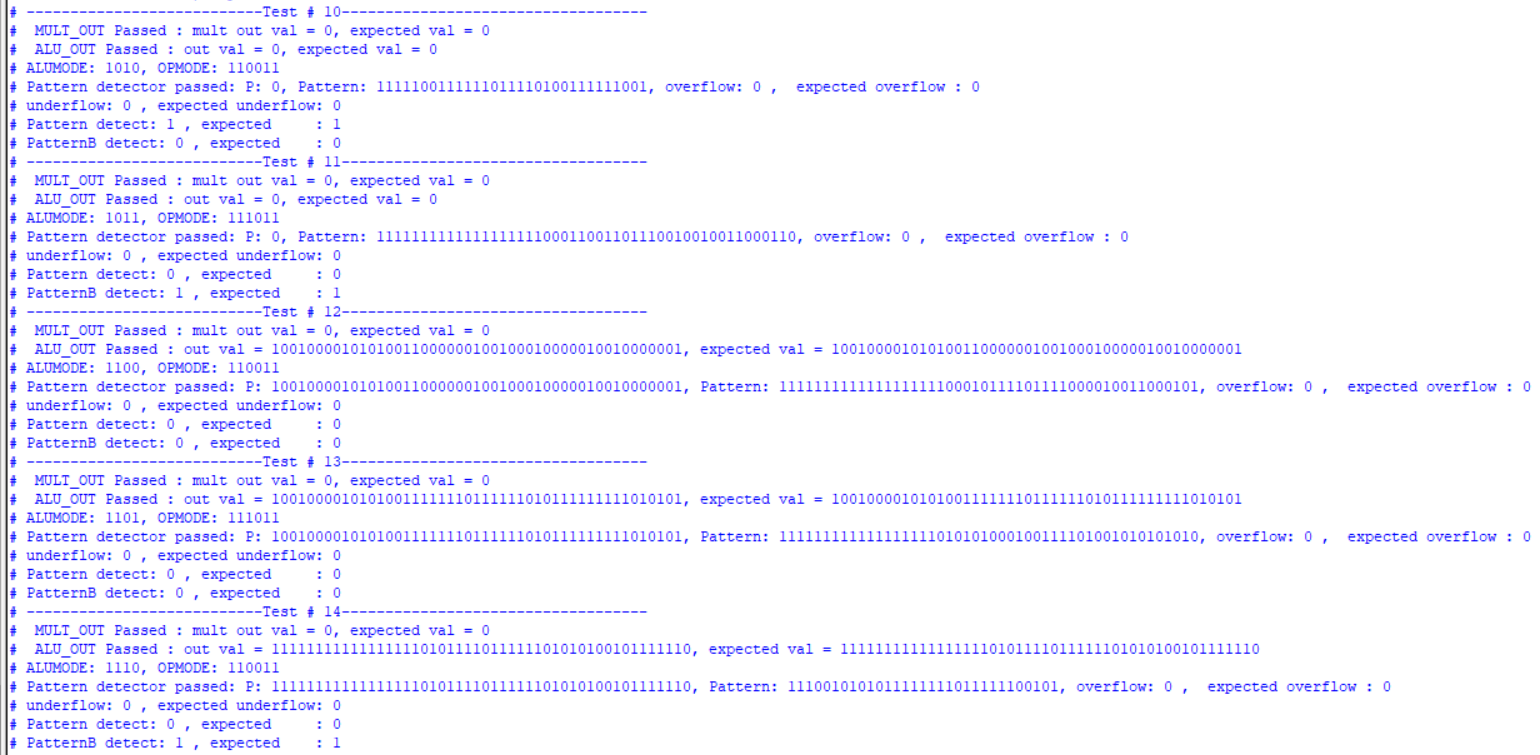
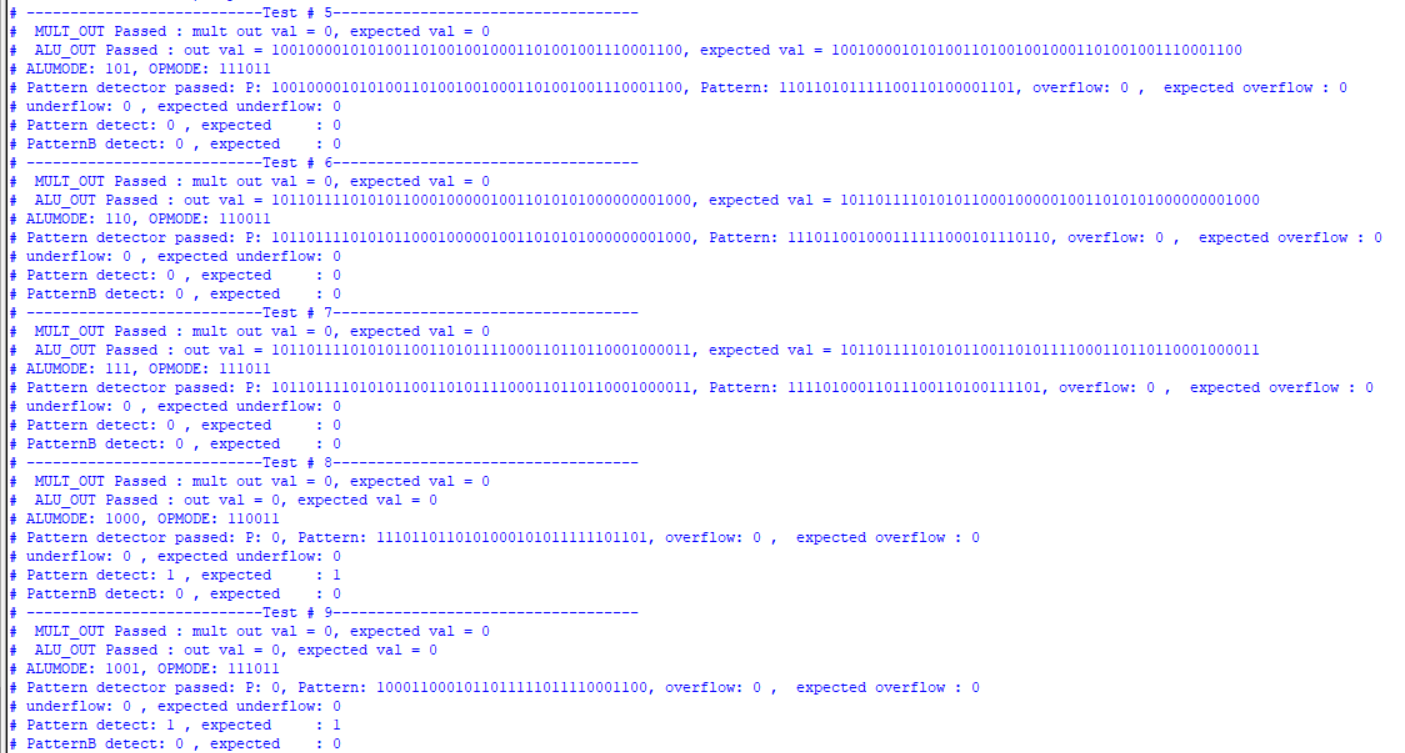
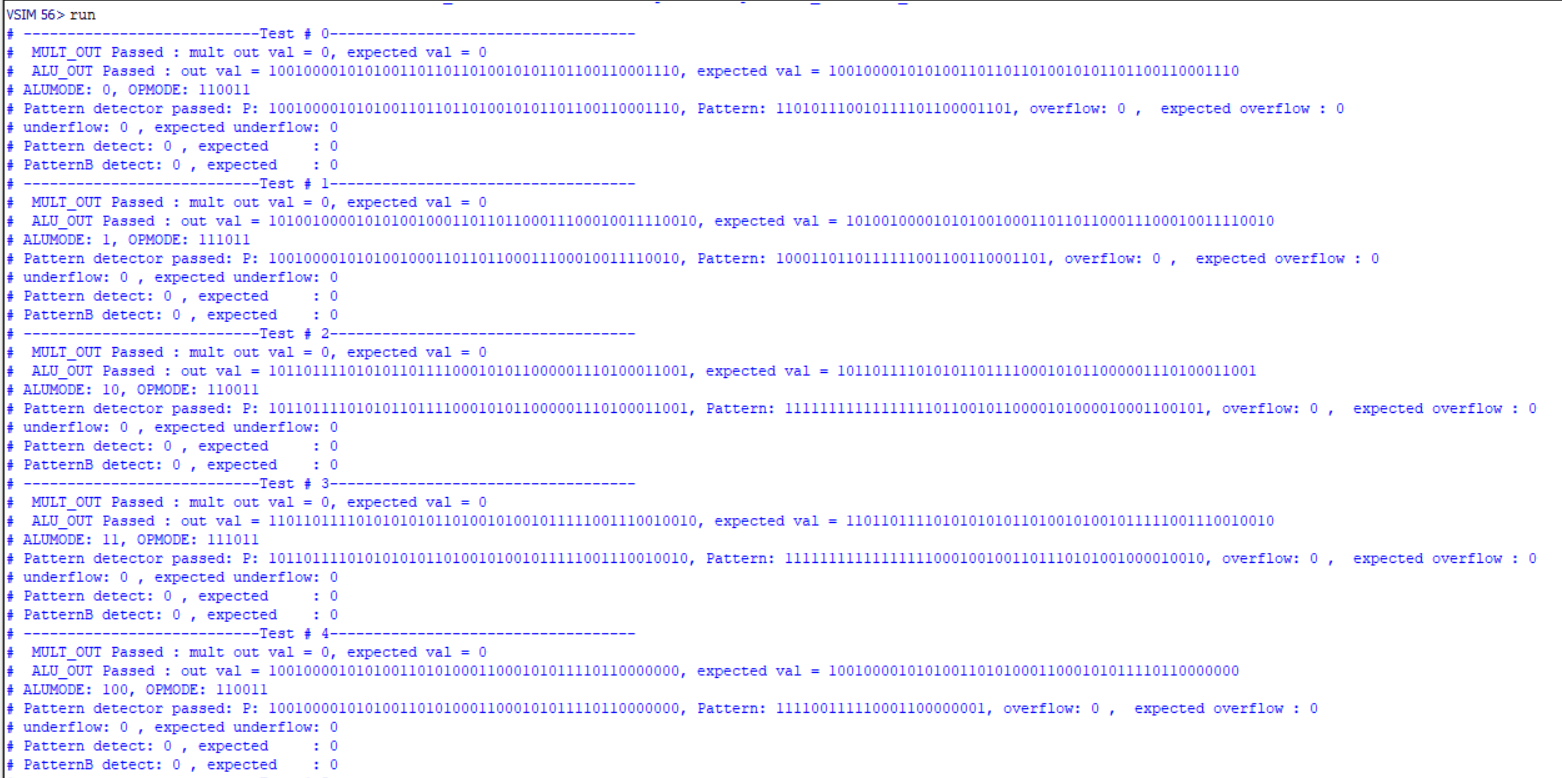
## Test 1: Multiplication Operation

All the possible combinations of the input “INMODE” were tested, random values for inputs A, D and B were used. All the tests passed successfully, figure . We have implemented 17 test cases corresponding to all the possible combinations of INMODE.



## Test 2: Arithmetic/Logic Operations

The same was applied to this test, and all of the cases passed successfully.





# Chapter 4: RTL Schematic and Synthesis Result

In this chapter, we will provide the schematic of the design and its elements and the obtained synthesis results (resource usage, power consumption and maximum clock). The synthesized design had the following attributes:

|  |  |
| --- | --- |
| Attribute | Value |
| BREG | 1 |
| AREG | 1 |
| DREG | 1 |
| CREG | 1 |
| ADREG | 1 |
| PREG | 1 |
| MREG | 1 |
| CARRYINREG | 1 |
| INMODEREG | 1 |
| ALUMODEREG | 1 |
| CARRYINSELREG | 1 |
| OPMODEREG | 1 |
| USE\_DPORT | 1 |
| ACASCREG | 1 |
| BCASCREG | 1 |
| A\_INPUT | "Direct” |
| B\_INPUT | "Direct” |
| USE\_MULT | "multiply”, “none” |
| USE\_SIMD | 0 |
| MASK | 48'h3FFFFFFFFFFF |
| PATTERN | 0 |
| SEL\_MASK | 0 (use default mask) |
| SEL\_PATTERN | PATTERN, input C |
| USE\_PATTERN\_DETECT | 1 |



## Design Schematic

A close-up of a table

Description automatically generated

1. A diagram of a computer program

   Description automatically generated

### Full System

### Multiplier (when USE\_MULT = “multiply”)

A diagram of a diagram

Description automatically generated

### ALU (when USE\_MULT = “multiply”)

A screenshot of a computer

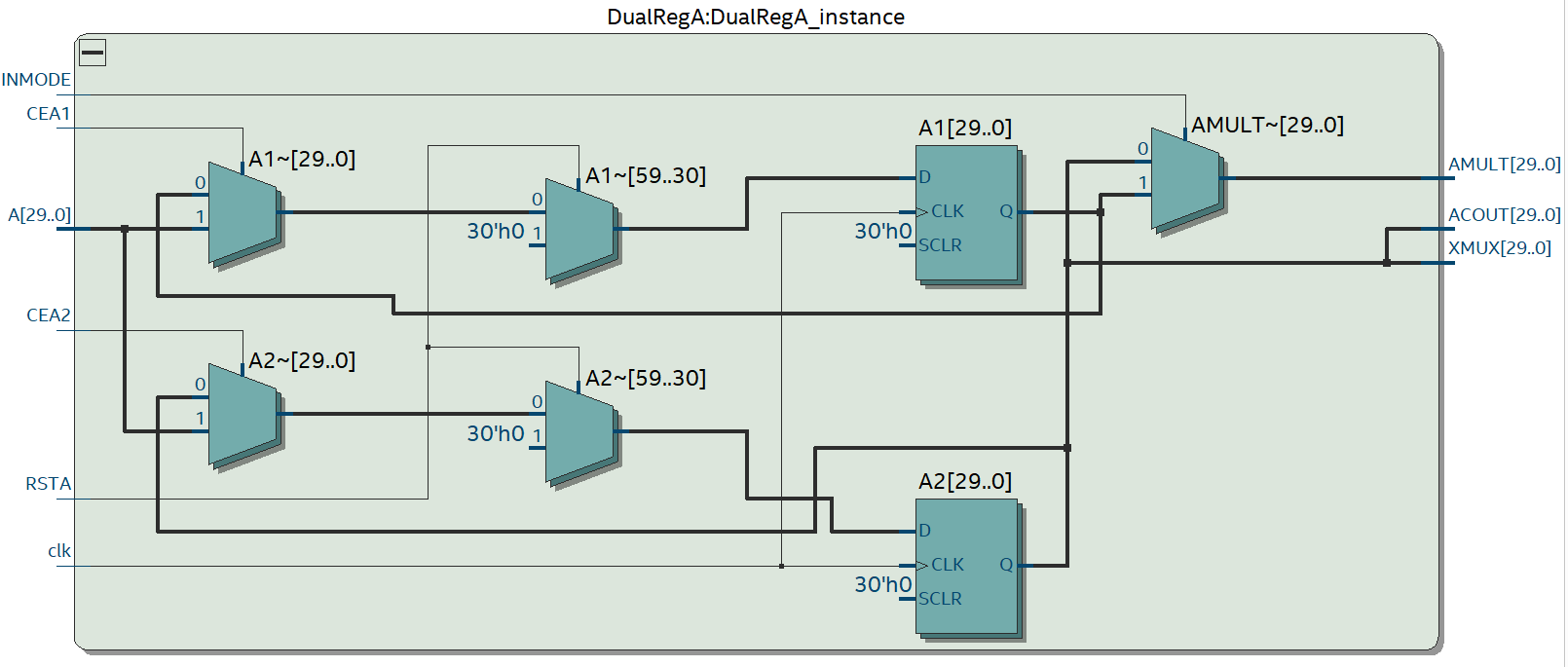
Description automatically generated

### Pre-adder (when USE\_DPORT = 1)

A diagram of a machine

Description automatically generated

### Dual Register A



### ALU (when USE\_MULT = “none”)

## Synthesis Result

To synthesize the DSP slice, we have selected the device “5CGTFD7D5F31C7” from the family Cyclone V, since it meets our design requirements size-wise (number of needed I/O pins is 417 and the number of available pins was 522).



### Resource Usage

The resources used depends on whether the multiplier was used or not, and as expected it increased when the multiplier was used (multiplier used 2 DSP blocks from the FPGA).

The following table presents the fitter resource usage summary for both cases.

|  |  |  |
| --- | --- | --- |
|  | Multiplier used | Multiplier not used |
| Logic Utilization (ALM needed/total) | 189/56480 | 484/56480 |
| ALM used for logic and registers | 72 | 75 |
| ALM used for LUT logic | 86 | 409 |
| ALM used for registers | 75 | 30 |
| Combinational ALUT usage for logic | 288 | 836 |
| Dedicated logic registers | 299 | 215 |
| I/O pins | 417 | 417 |
| Clock pins | 12 |  |
| DSP Blocks | 2 | 0 |
| Global clock | 1 | 1 |
| Maximum fan-out | 301 | 215 |
| Highest non-global fan-out | 79 | 107 |
| Total fan-out | 2979 | 4800 |
| Average fan-out | 2.01 | 2.52 |

### Resource utilization by entity

A screenshot of a computer

Description automatically generated(in caption : no multiplier)

(in caption: with multiplier )

A screenshot of a computer

Description automatically generated

### Power Consumption

The amount of consumed power depended on whether the multiplier was used or not, where the power increased when it was used as shown in the following table.

|  |  |  |
| --- | --- | --- |
|  | Multiplier used | Multiplier not used |
| Total thermal power dissipation | 437.92 mW | 375.17 mW |
| Core Dynamic thermal power dissipation | 8.07 mW | 0 |
| Core static thermal power dissipation | 349.29 mW | 349 mW |
| I/O thermal power dissipation | 80.56 mW | 26.17 mW |

### Maximum Clock

The maximum clock frequency for the design varied depending on whether the multiplier was used or not.

|  |  |  |
| --- | --- | --- |
|  | Multiplier used | Multiplier not used |
| Maximum clock frequency | 124.8 MHz | 110 MHz |

Hence, the input clock period was constrained to 10 nsec, and the design didn’t violate setup nor hold constraints for all operating conditions.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Fast Cold | Slow cold | Fast hot | Slow hot |
| Setup slack | 5.922 | 0.922 | 5.662 | 1.137 |
| Hold slack | 0.187 | 0.481 | 0.199 | 0.481 |

# Conclusion

DSP slices are used for many different applications in Digital Signal Processing. They have the ability to perform complex mathematical operations which make them ideal for implementing filters. The Cascading capapility helps in performing larger opertaions. DSP48E1 slice provides improved flexibility and utilization, improved efficiency of applications, reduced overall power consumption, and increased maximum frequency.   
As technology continues to evolve, DSP slices will continue to play a crucial role in various industries; DSP48E1 has wider functionality and unique features relative to prior generations.