

8-Bit ALU

Structural Vs Behavioral Design

Table of contents

1. Introduction

2. Project Implementation

- I. Stage 1: Top-Level Design
- II. Stage 2: Structural ALU
- III. Stage 3: Behavioral ALU

3. Conclusion



Introduction

This project report presents the design and implementation of an 8-bit Arithmetic Logic Unit (ALU), a fundamental building block of the Central Processing Unit (CPU) in a computer. The ALU performs various arithmetic and logical operations based on the inputs and the operation specified.

The ALU is a critical component in the digital processing of data. It comprises combinational logic that implements a variety of arithmetic operations such as addition, subtraction, and multiplication, as well as logical operations such as AND, OR, and inversion.

The primary goal of this project is to design an 8-bit ALU that can perform different arithmetic and logic operations on two 8-bit inputs. The design uses digital logic gates to implement the ALU operations, with the specific operation being selected by a 3-bit control signal. The output of the ALU is an 8-bit result, representing the outcome of the operation.

The design process involves breaking down the ALU into smaller, manageable components, implementing each component using digital logic gates, and then integrating these components to form the complete ALU. The design is then verified through simulation to ensure correct operation.

Introduction

Objectives:

- *Circuit design, and implementation using Verilog*
- *Simulation and testbench*
- *Design metrics (resources usage, power, operating frequency) evaluation*

Table of contents

1. Introduction

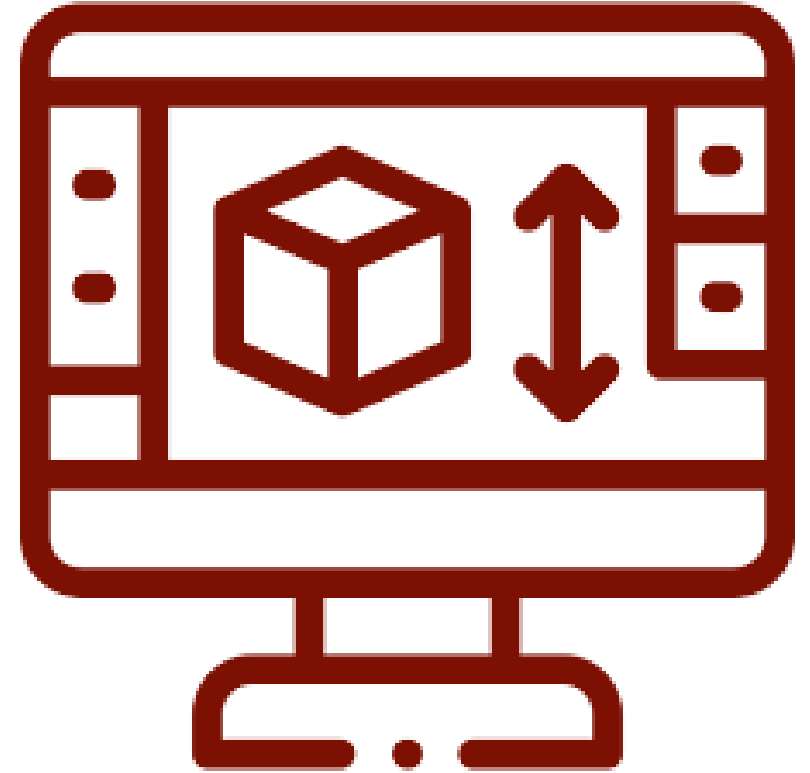
2. Project Implementation

I. Stage 1: Top-Level Design

II. Stage 2: Structural ALU

III. Stage 3: Behavioral ALU

3. Conclusion



Stage 1: Top-Level Design

Table 1: Operation Codes.

S0	S1	S2	S3	F	Description
0	0	0	0	A+B	Add
0	0	0	1	A-B	Subtract
0	0	1	1	B'+1	2's Complement
1	0	0	0	A AND B	AND
1	0	0	1	A XOR B	XOR
1	0	1	0	A OR B	OR
1	0	1	1	B'	1's Complement
1	1	0	0	A → →	RIGHT ROTATE
1	1	0	1	← ← A	LEFT ROTATE
1	1	1	0	A →	RIGHT SHIFT
1	1	1	1	← A	LEFT SHIFT

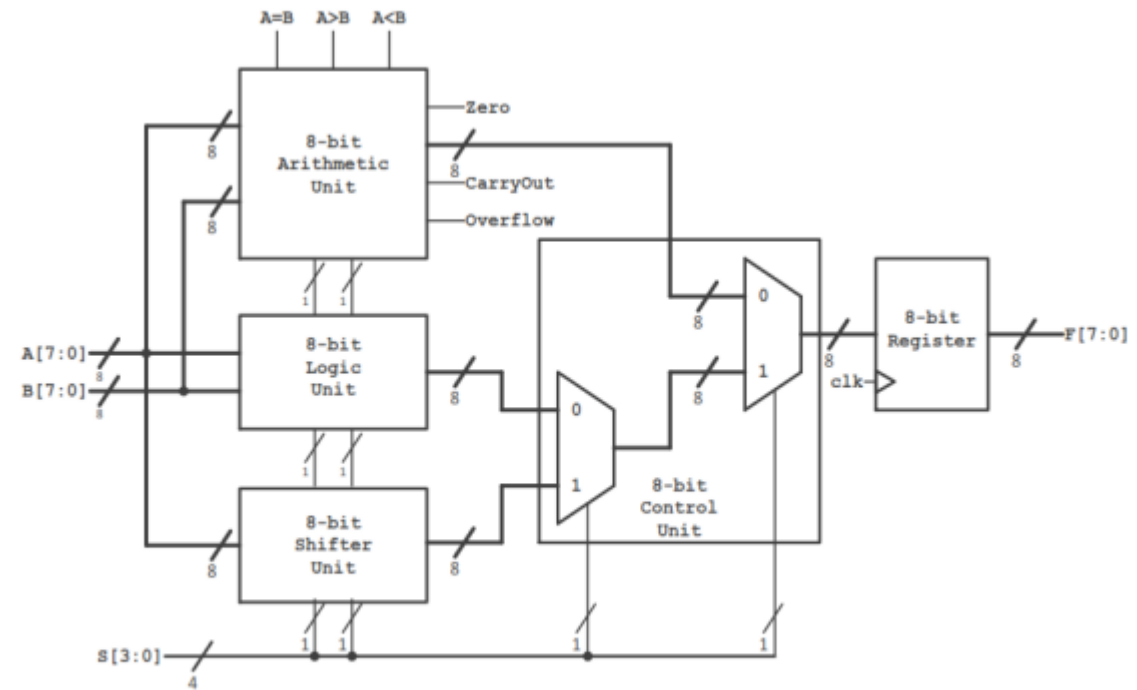


Figure 1: Block Diagram

Stage 1: Building The Main Modules

```
module Arithmetic_Unit(A,B,S,EQUAL,GT,LT,Zero,CarryOut,Overflow,AF);  
input [7:0]A,B;  
input[1:0] S;  
output [7:0] AF;  
output EQUAL,GT,LT,Zero,CarryOut,Overflow;  
endmodule
```

```
module Logic_Unit(A,B,S,LF);  
input [7:0]A,B;  
input[1:0] S;  
output [7:0] LF;  
endmodule
```

```
module Shifter_Unit(A,S,SHF);  
input [7:0]A;  
input[1:0] S;  
output [7:0] SHF;  
endmodule
```

```
module Control_Unit(AF,LF,SHF,S,F);  
input [7:0]AF,LF,SHF;  
input[1:0] S;  
output [7:0] F;  
endmodule
```

Stage 1: Wire them together in the main module

```
// Stage 1: Wire them together in the main module
module ALU(A,B,S,EQUAL,GT,LT,Zero,CarryOut,Overflow,F); //GT: Greater Than, LT: Less Than
input [7:0] A,B;
input [3:0] S;
output EQUAL,GT,LT,Zero,CarryOut,Overflow;
output [7:0] F;
wire [7:0] AF,LF,SHF; // you must declare any vector otherwise it will be declared automatically with a width of 1bit

Arithmetic_Unit AU(A,B,S[1:0],EQUAL,GT,LT,Zero,CarryOut,Overflow,AF);
Logic_Unit LU(A,B,S[1:0],LF);
Shifter_Unit SHU(A,S[1:0],SHF);
Control_Unit CONTU(AF,LF,SHF,S[3:2],F);

endmodule
```

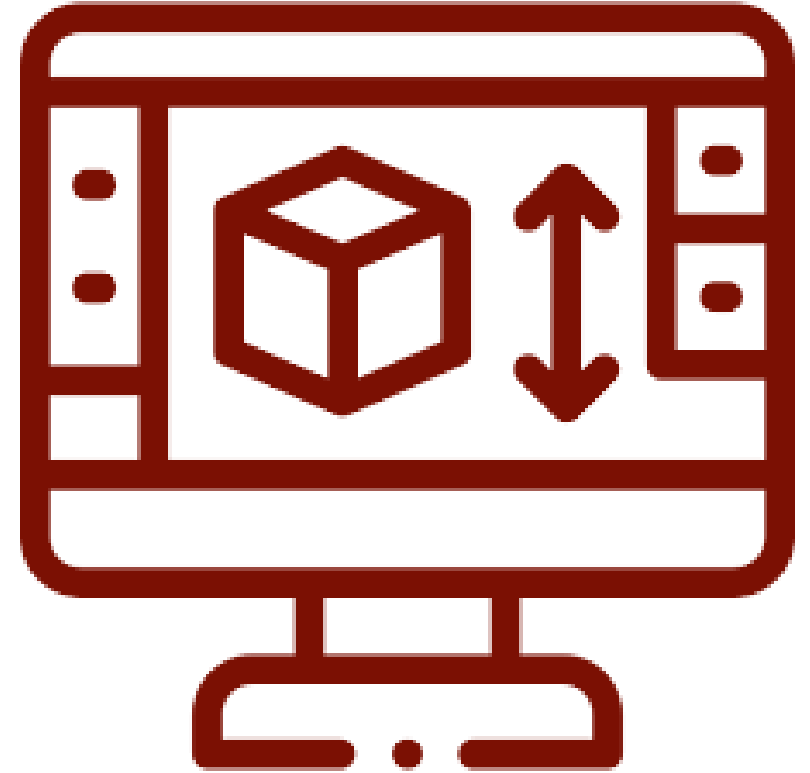

Table of contents

1. Introduction

2. Project Implementation

- I. Stage 1: Top-Level Design
- II. Stage 2: Structural ALU**
- III. Stage 3: Behavioral ALU

3. Conclusion



Stage 2: Structural ALU

Stage 2: Arithmetic Unit

```
module Arithmetic_Unit(A,B,S,EQUAL,GT,LT,Zero,CarryOut,Overflow,AF);
input [7:0]A,B;
input [1:0] S;
output [7:0] AF;
output EQUAL,GT,LT,Zero,CarryOut,Overflow;
wire [7:0]zeros8bit;
assign zeros8bit = {8{1'b0}};
//first input is either A or 0
wire [7:0] AX;
Mux2To1_8bit mux0(A,zeros8bit,S[1],AX);
ADDER_SUBTRACTOR A_S(AX,B,S[0],AF,CarryOut,Overflow,Zero);
//ADDER_SUBTRACTOR(A,B,S,SUM,COUT,OVERFLOW,ZERO);
//GT
FullComparator FC(A,B,EQUAL,GT,LT);
endmodule
```

Stage 2: Arithmetic Unit

```
//Arithmetic Blocks
module ADDER_SUBTRACTOR(A,B,S,SUM,COUT,OVERFLOW,ZERO);
input [7:0]A,B;
input S;
output [7:0]SUM;
output COUT,OVERFLOW,ZERO;
wire [7:0]BXOR;
wire [7:0]extend;
assign extend = {8{S}};
xor8 xor80(BXOR,B,extend);
wire [7:0] common,nott;
wire zeroBar;
FullAdder FA(A,BXOR,common,S,COUT,OVERFLOW);
not8 nt8(nott,common);
and8to1 a8t1(zeroBar,nott);
not(ZERO,zeroBar);
assign SUM = common;

endmodule
```

Stage 2: Arithmetic Unit

```
|  
module FullAdder(A,B,SUM,CIN,COU,OVERFLOW);  
input [7:0]A,B;  
input CIN;  
output [7:0]SUM;  
output COU,OVERFLOW;  
FullAdder1Bit FA0(A[0],B[0],SUM[0],CIN,c0);  
FullAdder1Bit FA1(A[1],B[1],SUM[1],c0,c1);  
FullAdder1Bit FA2(A[2],B[2],SUM[2],c1,c2);  
FullAdder1Bit FA3(A[3],B[3],SUM[3],c2,c3);  
FullAdder1Bit FA4(A[4],B[4],SUM[4],c3,c4);  
FullAdder1Bit FA5(A[5],B[5],SUM[5],c4,c5);  
FullAdder1Bit FA6(A[6],B[6],SUM[6],c5,c6);  
FullAdder1Bit FA7(A[7],B[7],SUM[7],c6,COU);  
xor xor0(OVERFLOW,c6,COU);  
endmodule  
module FullAdder1Bit(A,B,S,Cin,Co);  
input A,B,Cin;  
output S,Co;  
xor xor0 (w0,A,B);  
xor xor1 (S,w0,Cin);  
and and0 (w1,A,B);  
and and1 (w2,Cin,w0);  
or or0 (Co,w1,w2);  
endmodule
```

Stage 2: Arithmetic Unit

```
module Comparator(A,B,EQUAL,GT,LT);
input A,B;
output EQUAL,GT,LT;
not n0(BBAR,B);
not n1(ABAR,A);
xnor xn0(EQUAL,A,B);
and a0(GT,BBAR,A);
and a1(LT,ABAR,B);
endmodule
```

```
module Comparator2bit(A,B,EQUAL,GT,LT,E,G,L);
input [1:0] A,B;
input E,G,L;
output EQUAL,GT,LT;
wire [1:0] EQUALW,GTW,LTW;
genvar i;
generate
    for (i=0; i<2; i=i+1) begin : generate_comparator//
        Comparator CX (A[i],B[i],EQUALW[i],GTW[i],LTW[i]);
    end
    and an1(a0,EQUALW[1],GTW[0]);
    and ann1(a00,E,a0);
    and annn1(a11,E,GTW[1]);
    or o1(a1,a11,a00);
    or o2(GT,G,a1);

    and an2(b0,EQUALW[1],LTW[0]);
    and ann2(b00,E,b0);
    and annn2(b11,E,LTW[1]);
    or o3(b1,b11,b00);
    or o4(LT,L,b1);

    and an3(c1,EQUALW[0],EQUALW[1]);
    and an4(EQUAL,E,c1);
endgenerate
endmodule
```

Stage 2: Arithmetic Unit

```
module FullComparator(A,B,EQUAL,GT,LT);
input [7:0] A,B;

output EQUAL,GT,LT;
wire EQUALBAR;
//wire [7:0] EQUALW,GTW,LTW;
Comparator2bit c0(A[7:6],B[7:6],e0,g0,l0,1'b1,1'b0,1'b0);
Comparator2bit c1(A[5:4],B[5:4],e1,g1,l1,e0,g0,l0);
Comparator2bit c2(A[3:2],B[3:2],e2,g2,l2,e1,g1,l1);
Comparator2bit c3(A[1:0],B[1:0],EQUAL,GT,LT,e2,g2,l2);
//not n0(EQUAL,EQUALBAR);

endmodule
```

Stage 2: Logic Unit

```
module Logic_Unit(A,B,S,LF);  
  input [7:0]A,B;  
  input[1:0] S;  
  output [7:0] LF;  
  wire [7:0] AND_X,XOR_X,OR_X,NOT_X;  
  and88 and0(AND_X,A,B);  
  xor8 xor0 (XOR_X,A,B);  
  or88 or0 (OR_X,A,B);  
  not8 not0 (NOT_X,B);  
  Mux4To1_8bit mux0(AND_X,XOR_X,OR_X,NOT_X,S,LF);  
  
endmodule
```


Stage 2: Logic Unit

```
//Logic Blocks
module xor8(R,A,B);
input  [7:0]  A,B;
output [7:0]  R;
xor    xor0(R[0],A[0],B[0]);
xor    xor1(R[1],A[1],B[1]);
xor    xor2(R[2],A[2],B[2]);
xor    xor3(R[3],A[3],B[3]);
xor    xor4(R[4],A[4],B[4]);
xor    xor5(R[5],A[5],B[5]);
xor    xor6(R[6],A[6],B[6]);
xor    xor7(R[7],A[7],B[7]);
endmodule
module or88(R,A,B);
input  [7:0]  A,B;
output [7:0]  R;
or     or0(R[0],A[0],B[0]);
or     or1(R[1],A[1],B[1]);
or     or2(R[2],A[2],B[2]);
or     or3(R[3],A[3],B[3]);
or     or4(R[4],A[4],B[4]);
or     or5(R[5],A[5],B[5]);
or     or6(R[6],A[6],B[6]);
or     or7(R[7],A[7],B[7]);
endmodule
module and88(R,A,B);
input  [7:0]  A,B;
output [7:0]  R;
and    and0(R[0],A[0],B[0]);
and    and1(R[1],A[1],B[1]);
and    and2(R[2],A[2],B[2]);
and    and3(R[3],A[3],B[3]);
and    and4(R[4],A[4],B[4]);
and    and5(R[5],A[5],B[5]);
and    and6(R[6],A[6],B[6]);
and    and7(R[7],A[7],B[7]);
endmodule
```

```
module not8(R,X);
input  [7:0]  X;
output [7:0]  R;
not     not0(R[0],X[0]);
not     not1(R[1],X[1]);
not     not2(R[2],X[2]);
not     not3(R[3],X[3]);
not     not4(R[4],X[4]);
not     not5(R[5],X[5]);
not     not6(R[6],X[6]);
not     not7(R[7],X[7]);
endmodule
```

Stage 2: Shifter Unit

```
module Shifter_Unit(A,S,SHF);  
input [7:0]A;  
input[1:0] S;  
output [7:0] SHF;  
wire[7:0] RR,LR,RS,LS;  
RightRotate RRB(A,RR);  
LeftRotate LRB(A,LR);  
RightShift RSB(A,RS);  
LeftShift LSB(A,LS);  
Mux4To1_8bit mux0(RR,LR,RS,LS,S,SHF);  
endmodule
```

Stage 2: Shifter Unit

```
300 //Shifting Blocks
301 module RightRotate(X,R);
302 input  [7:0]  X;
303 output [7:0]  R;
304 assign R[0] = X[1];
305 assign R[1] = X[2];
306 assign R[2] = X[3];
307 assign R[3] = X[4];
308 assign R[4] = X[5];
309 assign R[5] = X[6];
310 assign R[6] = X[7];
311 assign R[7] = X[0];
312 endmodule
313
314 module LeftRotate(X,R);
315 input  [7:0]  X;
316 output [7:0]  R;
317 assign R[0] = X[7];
318 assign R[1] = X[0];
319 assign R[2] = X[1];
320 assign R[3] = X[2];
321 assign R[4] = X[3];
322 assign R[5] = X[4];
323 assign R[6] = X[5];
324 assign R[7] = X[6];
325 endmodule
326
```

Stage 2: Shifter Unit

```
326
327 module RightShift(X,R);
328   input  [7:0]  X;
329   output [7:0]  R;
330   assign R[0] = X[1];
331   assign R[1] = X[2];
332   assign R[2] = X[3];
333   assign R[3] = X[4];
334   assign R[4] = X[5];
335   assign R[5] = X[6];
336   assign R[6] = X[7];
337   assign R[7] = 0;
338   endmodule
339
340 module LeftShift(X,R);
341   input  [7:0]  X;
342   output [7:0]  R;
343   assign R[0] = 0;
344   assign R[1] = X[0];
345   assign R[2] = X[1];
346   assign R[3] = X[2];
347   assign R[4] = X[3];
348   assign R[5] = X[4];
349   assign R[6] = X[5];
350   assign R[7] = X[6];
351   endmodule
```

Stage 2: Control Unit

```
109 module Control_Unit(AF,LF,SHF,S,F);  
110   input [7:0]AF,LF,SHF;  
111   input [1:0] S;  
112   output [7:0] F;  
113   wire [7:0]w0;  
114   Mux2To1_8bit mux0(LF,SHF,S[0],w0);  
115   Mux2To1_8bit mux1(AF,w0,S[1],F);  
116   endmodule  
117  
118
```

Stage 2: Control Unit

```
109 module Control_Unit(AF,LF,SHF,S,F);
110   input [7:0]AF,LF,SHF;
111   input [1:0] S;
112   output [7:0] F;
113   wire [7:0]w0;
114   Mux2To1_8bit mux0(LF,SHF,S[0],w0);
115   Mux2To1_8bit mux1(AF,w0,S[1],F);
116   endmodule
117
118
```

Stage 2: Control Unit

```
231 //Control Blocks
232 module Mux2To1_8bit(M1,M2,S,R);
233 input[7:0] M1,M2;
234 input S;
235 output[7:0] R;
236 Mux2To1_1bit mux1(M1[0],M2[0],S,R[0]);
237 Mux2To1_1bit mux2(M1[1],M2[1],S,R[1]);
238 Mux2To1_1bit mux3(M1[2],M2[2],S,R[2]);
239 Mux2To1_1bit mux4(M1[3],M2[3],S,R[3]);
240 Mux2To1_1bit mux5(M1[4],M2[4],S,R[4]);
241 Mux2To1_1bit mux6(M1[5],M2[5],S,R[5]);
242 Mux2To1_1bit mux7(M1[6],M2[6],S,R[6]);
243 Mux2To1_1bit mux8(M1[7],M2[7],S,R[7]);
244
245 endmodule
246 module Mux2To1_1bit(M1,M2,S,R);|
247 input M1,M2;
248 input S;
249 output R;
250 wire SBar;
251 not n0(SBar,S);
252 and a0(w1,M1,SBar);
253 and a1(w2,M2,S);
254 or o0(R,w1,w2);
255 endmodule
```

Stage 2: Test Bench

[illegible]

Stage 2: Design Metrics(Power)

The screenshot shows the 'Power Analyzer Summary' window in Quartus Prime. The window has a title bar with tabs for 'ALU.v', 'Compilation Report - ALU', 'alu_tb.v', 'ALU_inst.v', and 'Power Analyzer Tool'. The left sidebar contains a 'Table of Contents' with a tree view. The main area displays a summary of power analysis results.

Table of Contents:

- Flow Summary
- Flow Settings
- Flow Non-Default Global Setting
- Flow Elapsed Time
- Flow OS Summary
- Flow Log
- Analysis & Synthesis
- Fitter
- Power Analyzer
 - Parallel Compilation
 - Summary**
 - Settings
 - Indeterminate Toggle Rates
 - Operating Conditions Used
 - Thermal Power Dissipation t
 - Thermal Power Dissipation t
 - Thermal Power Dissipation t
 - Core Dynamic Thermal Powe
 - Current Drawn from Voltage
 - Confidence Metric Details
 - Signal Activities
 - Messages
 - Flow Messages
 - Flow Suppressed Messages

Power Analyzer Summary

Power Analyzer Status: Successful - Thu Dec 21 09:30:47 2023

Quartus Prime Version: 22.1std.2 Build 922 07/20/2023 SC Lite Edition

Revision Name: ALU

Top-level Entity Name: ALU

Family: Cyclone V

Device: 5CGXFC9E7F31C8

Power Models: Final

Total Thermal Power Dissipation: 526.35 mW

Core Dynamic Thermal Power Dissipation: 0.00 mW

Core Static Thermal Power Dissipation: 518.45 mW

I/O Thermal Power Dissipation: 7.90 mW

Power Estimation Confidence: Low: user provided insufficient toggle rate data

Stage 2: Design Metrics(Delay)

Table of Contents

Setup Summary

Hold Summary

Recovery Summary

Removal Summary

Minimum Pulse Width

Metastability Summary

Fast 1100mV OC Model

Setup Summary

Hold Summary

Recovery Summary

Removal Summary

Minimum Pulse Width

Metastability Summary

Multicorner Timing Analysis

Advanced I/O Timing

Board Trace Model As

Input Transition Times

Signal Integrity Metrics

(Slow 1100mv Oc Model)

(Slow 1100mv 85c Model)

(Fast 1100mv Oc Model)

(Fast 1100mv 85c Model)

Clock Transfers

Signal Integrity Metrics (Slow 1100mv Oc Model)

<<Filter>>

	Pin	I/O Standard	Board Delay on Rise	Board Delay on Fall	Steady State Voh at FPGA Pin	Steady State Vol at FPGA Pin	Voh Max
1	EQUAL	2.5 V	0 s	0 s	2.32 V	5.91e-08 V	2.35 V
2	GT	2.5 V	0 s	0 s	2.32 V	5.91e-08 V	2.35 V
3	LT	2.5 V	0 s	0 s	2.32 V	6.86e-08 V	2.4 V
4	Zero	2.5 V	0 s	0 s	2.32 V	6.86e-08 V	2.4 V
5	CarryOut	2.5 V	0 s	0 s	2.32 V	6.72e-08 V	2.4 V
6	Overflow	2.5 V	0 s	0 s	2.32 V	6.72e-08 V	2.4 V
7	F[0]	2.5 V	0 s	0 s	2.32 V	6.72e-08 V	2.4 V
8	F[1]	2.5 V	0 s	0 s	2.32 V	6.72e-08 V	2.4 V
9	F[2]	2.5 V	0 s	0 s	2.32 V	6.72e-08 V	2.4 V
10	F[3]	2.5 V	0 s	0 s	2.32 V	5.91e-08 V	2.35 V
11	F[4]	2.5 V	0 s	0 s	2.32 V	6.72e-08 V	2.4 V
12	F[5]	2.5 V	0 s	0 s	2.32 V	6.86e-08 V	2.4 V
13	F[6]	2.5 V	0 s	0 s	2.32 V	6.72e-08 V	2.4 V
14	F[7]	2.5 V	0 s	0 s	2.32 V	5.91e-08 V	2.35 V
15	FF[0]	2.5 V	0 s	0 s	2.32 V	6.72e-08 V	2.4 V
16	FF[1]	2.5 V	0 s	0 s	2.32 V	6.86e-08 V	2.4 V
17	FF[2]	2.5 V	0 s	0 s	2.32 V	6.72e-08 V	2.4 V
18	FF[3]	2.5 V	0 s	0 s	2.32 V	6.72e-08 V	2.4 V
19	FF[4]	2.5 V	0 s	0 s	2.32 V	6.86e-08 V	2.4 V
20	FF[5]	2.5 V	0 s	0 s	2.32 V	6.72e-08 V	2.4 V
21	FF[6]	2.5 V	0 s	0 s	2.32 V	6.86e-08 V	2.4 V
22	FF[7]	2.5 V	0 s	0 s	2.32 V	6.72e-08 V	2.4 V

Stage 2: Design Metrics(Resources Usage)

Fitter Status	Successful - Thu Dec 21 10:21:53 2023
Quartus Prime Version	22.1std.2 Build 922 07/20/2023 SC Lite Edition
Revision Name	ALU
Top-level Entity Name	ALU
Family	Cyclone V
Device	5CGXFC9E7F31C8
Timing Models	Final
Logic utilization (in ALMs)	44 / 113,560 (< 1 %)
Total registers	0
Total pins	43 / 536 (8 %)
Total virtual pins	0
Total block memory bits	0 / 12,492,800 (0 %)
Total RAM Blocks	0 / 1,220 (0 %)
Total DSP Blocks	0 / 342 (0 %)
Total HSSI RX PCSs	0 / 12 (0 %)
Total HSSI PMA RX Deserializers	0 / 12 (0 %)
Total HSSI TX PCSs	0 / 12 (0 %)
Total HSSI PMA TX Serializers	0 / 12 (0 %)
Total PLLs	0 / 20 (0 %)
Total DLLs	0 / 4 (0 %)

Stage 2: ALU

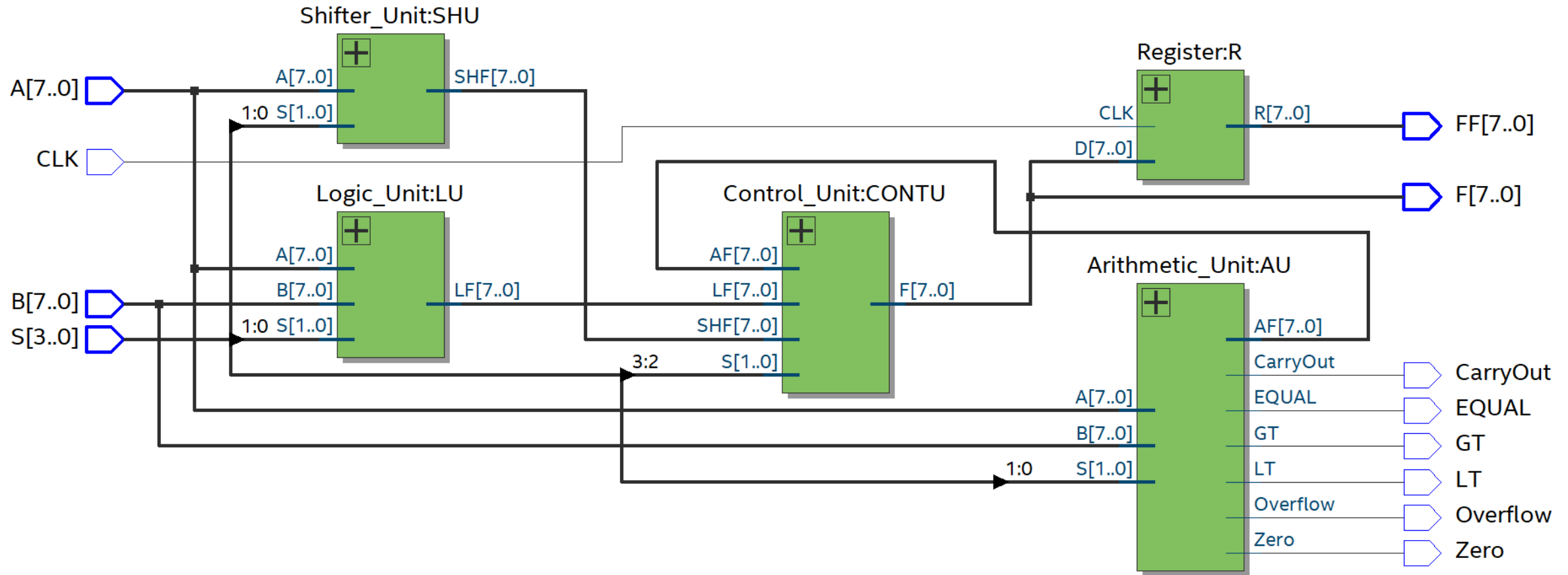


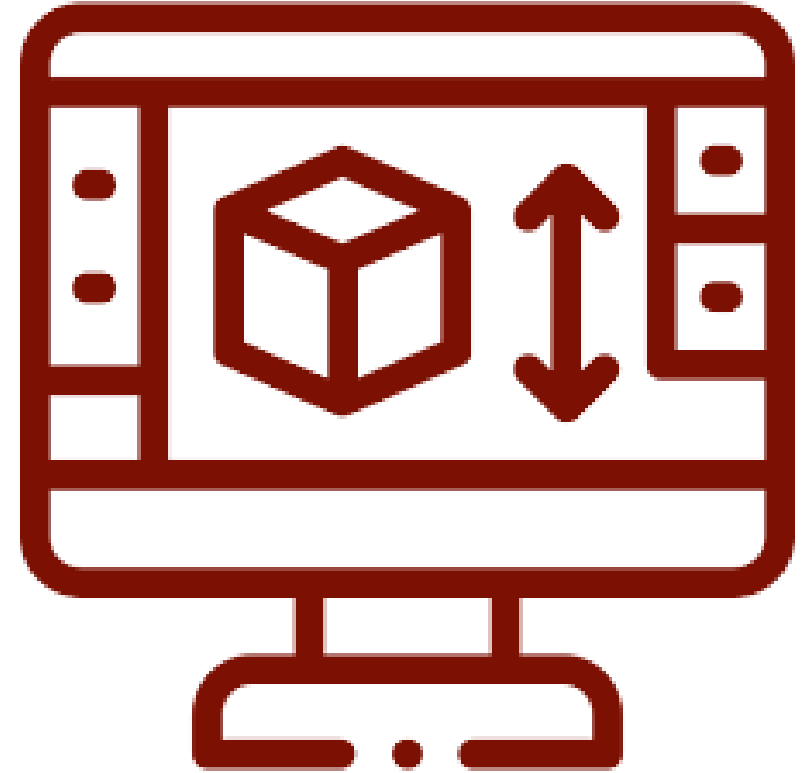
Table of contents

1. Introduction

2. Project Implementation

- I. Stage 1: Top-Level Design
- II. Stage 2: Structural ALU
- III. Stage 3: Behavioral ALU**

3. Conclusion



Stage 3: Behavioural

Stage 3: Behavioural

```
module ALU(CLK,A,B,S,EQUAL,GT,LT,Zero,CarryOut,Overflow,F,FF);
input [7:0]A,B;
input [3:0]S;
input CLK;
output reg EQUAL,GT,LT,Zero,CarryOut,Overflow;
output reg [7:0] F,FF;
reg [8:0] FEXT;
reg [7:0]BX;

always@(posedge CLK)
begin
FF <=F;
end
always @(*)
begin
case(S)
4'b0000:begin
BX=B;
FEXT= A+BX;
F = FEXT[7:0];
end
4'b0001:
begin
BX=-B;
FEXT=A+BX;
F = FEXT[7:0];
end
endcase
GT= (A>B)?1'b1:1'b0;
LT= (A<B)?1'b1:1'b0;
EQUAL= (A==B)?1'b1:1'b0;
Zero = (F==0)?1'b1:1'b0;
if((A[7]==BX[7]) && (BX[7] !=F[7]))
Overflow =1;
else
Overflow=0;
CarryOut=FEXT[8];
end
endmodule
```

Register

```
end
4'b0011:F=-B;
4'b1000:F=A&B;
4'b1001:F=A^B;
4'b1010:F=A|B;
4'b1011:F=~B;
4'b1100:F=(B>>1)|({B[0], {7{1'b0}}});
4'b1101:F=(B<<1)|({{7{1'b0}}, B[7]});
4'b1110:F=B>>1;
4'b1111:F=B<<1;
//default:0;

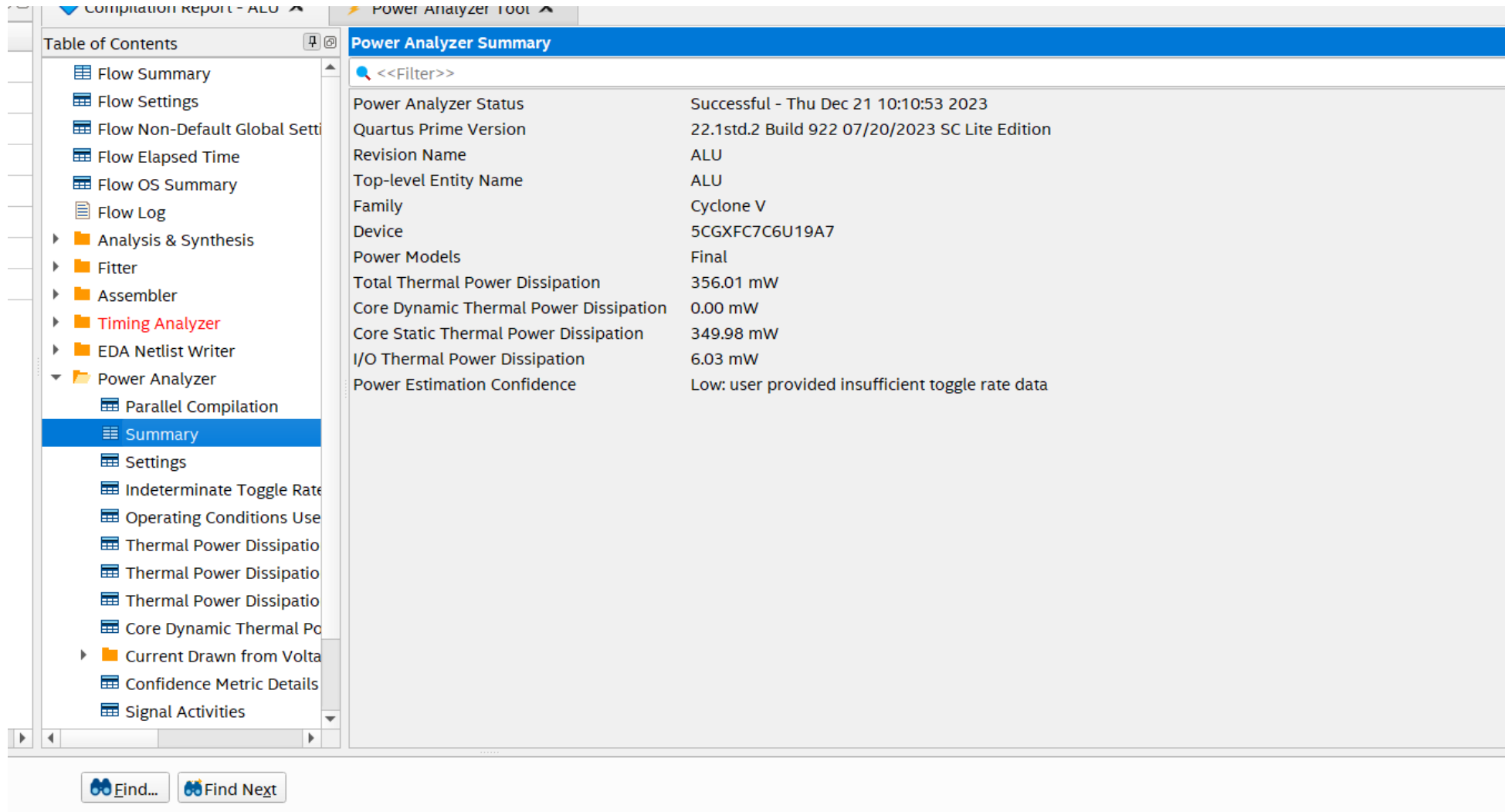
endcase
GT= (A>B)?1'b1:1'b0;
LT= (A<B)?1'b1:1'b0;
EQUAL= (A==B)?1'b1:1'b0;
Zero = (F==0)?1'b1:1'b0;
if((A[7]==BX[7]) && (BX[7] !=F[7]))
Overflow =1;
else
Overflow=0;
CarryOut=FEXT[8];
end
endmodule
```

Flags

Stage 3: Test Bench

[illegible]

Stage 3: Design Metrics(Power)



The screenshot displays the 'Power Analyzer Summary' window in the Quartus Prime software. The left sidebar contains a 'Table of Contents' with the following items:

- Flow Summary
- Flow Settings
- Flow Non-Default Global Settings
- Flow Elapsed Time
- Flow OS Summary
- Flow Log
- Analysis & Synthesis
- Fitter
- Assembler
- Timing Analyzer
- EDA Netlist Writer
- Power Analyzer
 - Parallel Compilation
 - Summary (selected)
 - Settings
 - Indeterminate Toggle Rate
 - Operating Conditions Used
 - Thermal Power Dissipation
 - Thermal Power Dissipation
 - Thermal Power Dissipation
 - Core Dynamic Thermal Power
 - Current Drawn from Voltage
 - Confidence Metric Details
 - Signal Activities

The main area of the window displays the 'Power Analyzer Summary' table:

Power Analyzer Summary	
<<Filter>>	
Power Analyzer Status	Successful - Thu Dec 21 10:10:53 2023
Quartus Prime Version	22.1std.2 Build 922 07/20/2023 SC Lite Edition
Revision Name	ALU
Top-level Entity Name	ALU
Family	Cyclone V
Device	5CGXFC7C6U19A7
Power Models	Final
Total Thermal Power Dissipation	356.01 mW
Core Dynamic Thermal Power Dissipation	0.00 mW
Core Static Thermal Power Dissipation	349.98 mW
I/O Thermal Power Dissipation	6.03 mW
Power Estimation Confidence	Low: user provided insufficient toggle rate data

At the bottom of the window, there are two buttons: 'Find...' and 'Find Next'.

Stage 3: Design Metrics(Delay)

Signal Integrity Metrics (Slow 1100mv n40c Model)							
<<Filter>>							
	Pin	I/O Standard	Board Delay on Rise	Board Delay on Fall	Steady State Voh at FPGA Pin	Steady State Vol at FPGA Pin	Voh Max
1	EQUAL	2.5 V	0 s	0 s	2.32 V	1.56e-08 V	2.43 V
2	GT	2.5 V	0 s	0 s	2.32 V	1.56e-08 V	2.43 V
3	LT	2.5 V	0 s	0 s	2.32 V	1.56e-08 V	2.43 V
4	Zero	2.5 V	0 s	0 s	2.32 V	1.56e-08 V	2.43 V
5	CarryOut	2.5 V	0 s	0 s	2.32 V	1.56e-08 V	2.43 V
6	Overflow	2.5 V	0 s	0 s	2.32 V	1.56e-08 V	2.43 V
7	F[0]	2.5 V	0 s	0 s	2.32 V	1.56e-08 V	2.43 V
8	F[1]	2.5 V	0 s	0 s	2.32 V	1.6e-08 V	2.42 V
9	F[2]	2.5 V	0 s	0 s	2.32 V	1.56e-08 V	2.43 V
10	F[3]	2.5 V	0 s	0 s	2.32 V	1.56e-08 V	2.43 V
11	F[4]	2.5 V	0 s	0 s	2.32 V	1.56e-08 V	2.43 V
12	F[5]	2.5 V	0 s	0 s	2.32 V	1.6e-08 V	2.42 V
13	F[6]	2.5 V	0 s	0 s	2.32 V	1.6e-08 V	2.42 V
14	F[7]	2.5 V	0 s	0 s	2.32 V	1.6e-08 V	2.42 V
15	FF[0]	2.5 V	0 s	0 s	2.32 V	1.33e-08 V	2.36 V
16	FF[1]	2.5 V	0 s	0 s	2.32 V	1.56e-08 V	2.43 V
17	FF[2]	2.5 V	0 s	0 s	2.32 V	1.33e-08 V	2.36 V
18	FF[3]	2.5 V	0 s	0 s	2.32 V	1.6e-08 V	2.42 V
19	FF[4]	2.5 V	0 s	0 s	2.32 V	1.33e-08 V	2.36 V
20	FF[5]	2.5 V	0 s	0 s	2.32 V	1.56e-08 V	2.43 V
21	FF[6]	2.5 V	0 s	0 s	2.32 V	1.33e-08 V	2.36 V
22	FF[7]	2.5 V	0 s	0 s	2.32 V	1.56e-08 V	2.43 V

IP Catalog

Stage 3: Design Metrics(Resources Usage)

Fitter Summary	
<<Filter>>	
Fitter Status	Successful - Thu Dec 21 10:02:10 2023
Quartus Prime Version	22.1std.2 Build 922 07/20/2023 SC Lite Edition
Revision Name	ALU
Top-level Entity Name	ALU
Family	Cyclone V
Device	5CGXFC7C6U19A7
Timing Models	Final
Logic utilization (in ALMs)	60 / 56,480 (< 1 %)
Total registers	8
Total pins	43 / 268 (16 %)
Total virtual pins	0
Total block memory bits	0 / 7,024,640 (0 %)
Total RAM Blocks	0 / 686 (0 %)
Total DSP Blocks	0 / 156 (0 %)
Total HSSI RX PCSs	0 / 6 (0 %)
Total HSSI PMA RX Deserializers	0 / 6 (0 %)
Total HSSI TX PCSs	0 / 6 (0 %)
Total HSSI PMA TX Serializers	0 / 6 (0 %)
Total PLLs	0 / 13 (0 %)
Total DLLs	0 / 4 (0 %)

Stage 3: ALU

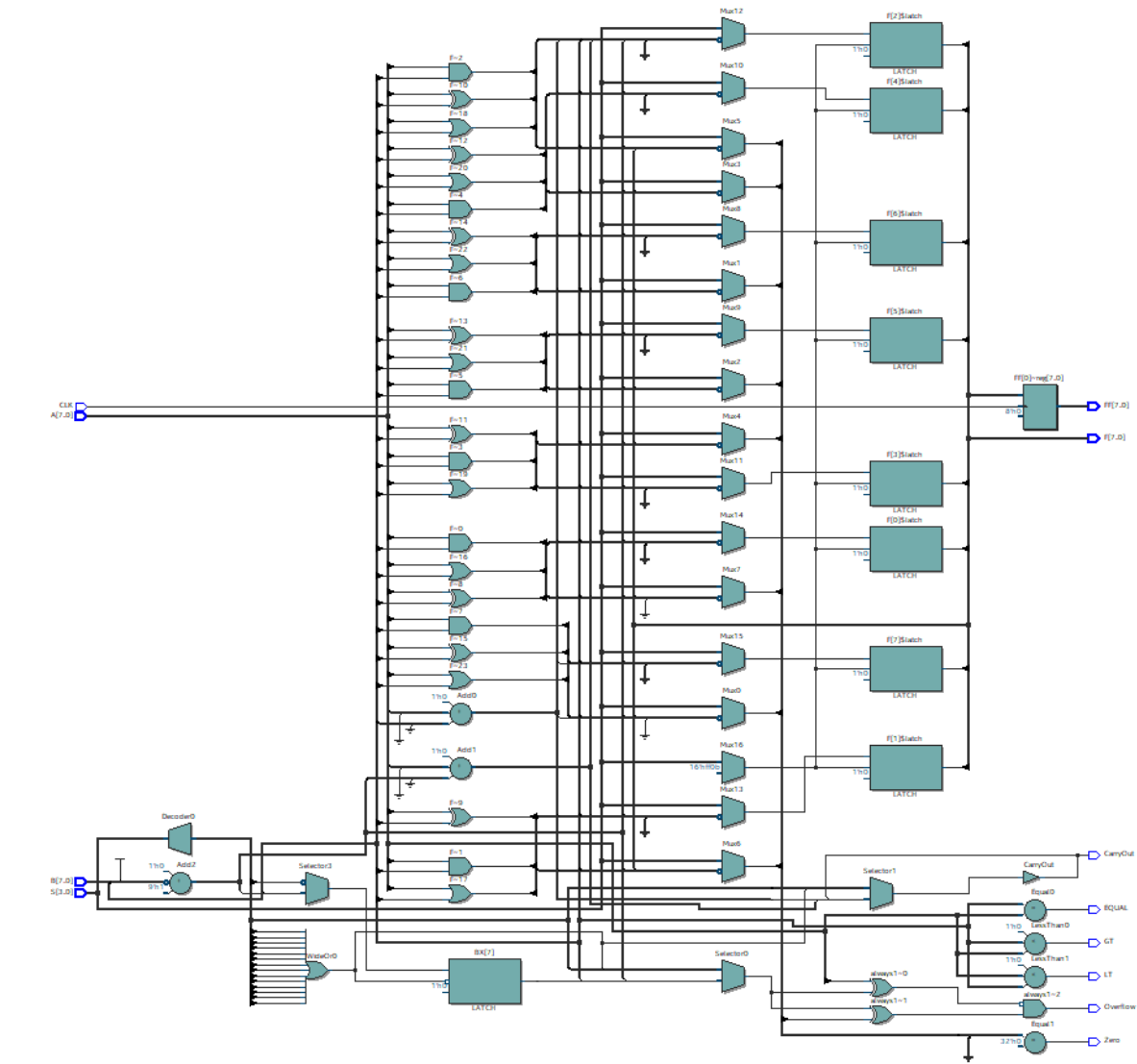


Table of contents

1. Introduction

2. Project Implementation

- I. Stage 1: Top-Level Design
- II. Stage 2: Structural ALU
- III. Stage 3: Behavioral ALU

3. Conclusion



Conclusion

In conclusion, the project of designing and implementing an 8-bit Arithmetic Logic Unit (ALU) has been successfully completed. The ALU was designed using digital logic gates to perform various arithmetic and logical operations, including addition, subtraction, AND, OR, etc. The design was then implemented and verified through simulation.

Through the project, the purpose and functionality of an ALU in a computer system has been demonstrated. The ALU is a critical component in a CPU, performing the arithmetic and logical operations that are essential to the processing of data. The 8-bit ALU designed in this project can perform different arithmetic and logical operations on two 8-bit inputs, with the specific operation being selected by a 4-bit control signal



THANK
YOU