Case Study for 32-Bit Pipelined CPU Design with New ALU Architecture

Adam Sumner

Illinois Institute of Technology

ECE 429-01

December 4th, 2015

Contents

1 Introduction

The objective of this project is to understand how a 32-bit pipelined Central Processing Unit (CPU) functions and how it is both physically and logically constructed. As the name suggests, the length of a word is 32 bits wide. Furthermore, due to the wonderful invention of pipelining, multiple instructions can be executed simultaneously. A CPU's normal processes (excluding interrupts) are synchronized, therefore the CPU will be in sync using an external clock signal. The instruction signals for addressing the memory file, Arithmetic Logic Unit (ALU) operands, and the ALU operation are also external.

1.1 Circuit Description

The complete overview of the primary building blocks and signals of the CPU are shown in Figure 1. The two main components are the memory file and the ALU. The external clock signal synchronizes the capture and release of data within the 32 x 32 memory file block. Because of pipelining, each instruction executed can be performed in 2 clock cycles. The first cycle involves the two decoders translating the external address selection signals used for specifying the contents of the memory file that should be read. In addition to this, the multiplexer blocks are used to select the operands for the ALU. In the second clock cycle, the ALU performs the specified operation. The output of the ALU can then be read from outside the CPU through a tri-state buffer, and they can also be written back into the memory file in the word specified by Address B.

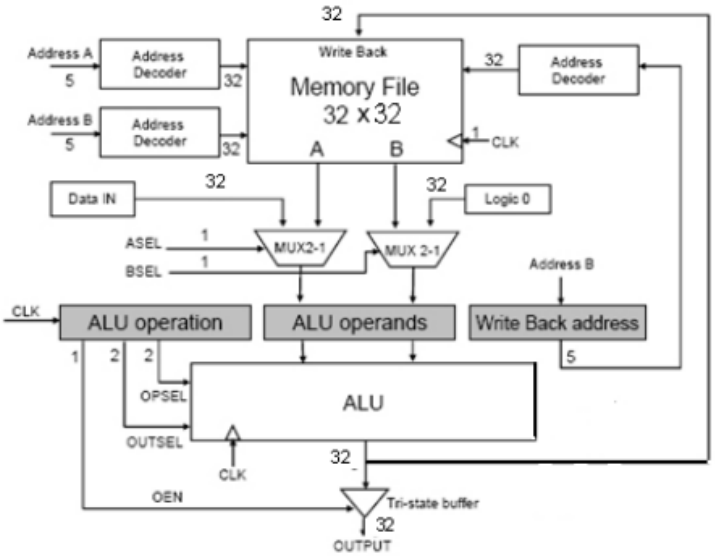

Figure 1: Circuit Overview

1.2 Memory File

The memory file stores 32 32-bit words. There are two read ports and one write port. The words that are read in each clock cycle are specified in Address A and Address B from Figure 1. The primary storage element used in this design is a D-register. The output of each register is connected to the two output read ports through tri-state buffers. The buffers are enabled through the decoded address A and address B. The value of address B specifies the word address in the memory file where the results of the ALU will be stored after the computation is complete in the second clock cycle.

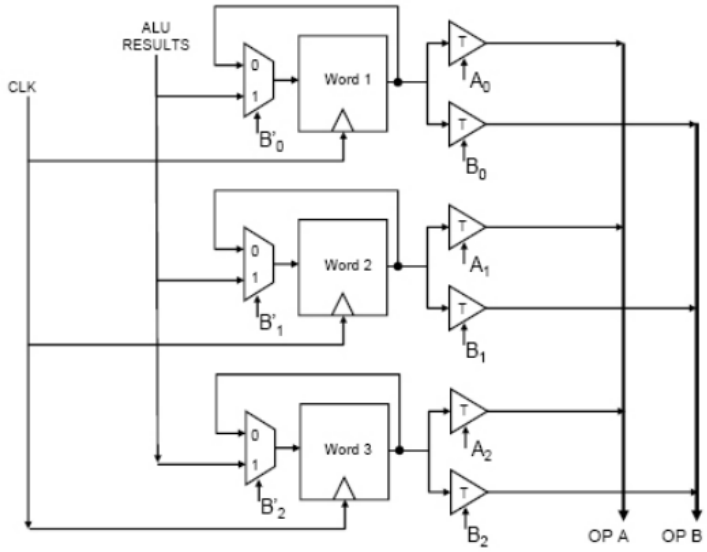

Figure 2: Memory File

1.3 ALU

The ALU of the circuit has two operands A and B, and it implements the following functions:

- A * B : multiplication
- A + B : addition
- A - B : subtraction
- B - A : subtraction
- A or B : logic OR function
- A and B : logic AND function
- A xor B: exclusive XOR function

- A xnor B : logic XNOR function

The ALU has three primary operation blocks: the multiplier, the adder, and the logic function block. This is shown in figure 3. The multiplier carries out the multiplication function. Because multiplication results in a 64-bit result, 2 clock cycles are used to store the result. This means that to execute the multiply instruction, 3 cycles instead of 2 are used in total.

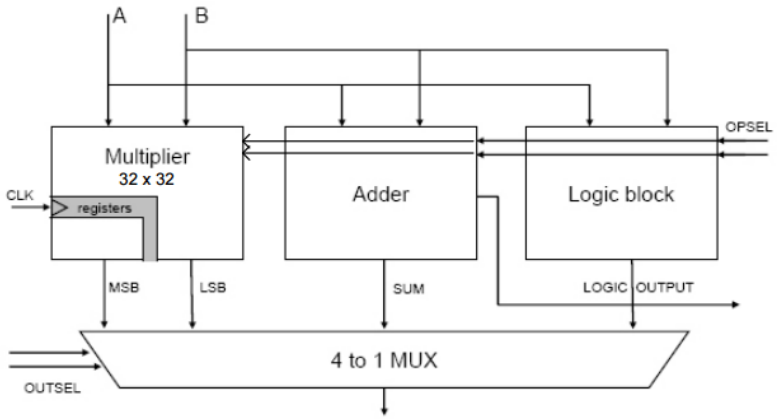

Figure 3: ALU Overview

The adder circuit within the ALU is a 32-bit adder/subtractor circuit. It executes the addition and subtraction operations. The selection of the operation is done by the two externally defined operation select signals OPSEL. The same signals are used to specify the operation executed within the logic function of the ALU. The final output select signal OUTSEL specifies the result of the ALU for output using the 4-to-1 multiplexer.

1.4 Synchronization

Each instruction executed by the CPU is determined by the external control signals. An instruction word is shown in Figure 4. Once the clock signals high, the instruction is applied to the signals that control the CPU operation. Since each instruction is executed in two steps, some control signals need to be stored at the internal registers of the CPU. In the first step of the instruction, the instruction specifies the contents of the memory file that will be read from the read ports A and B, and the operands of the ALU. In the second cycle, the control signals determine the operation to be executed internally in the ALU and the ALU output. The results of the ALU will be available if the OEN signal is set, and the result will also be written back into the memory file. This address is specified by address B. Data is written at the next positive edge of the clock. The period of the clock signal is determined by the longest data path delay in the circuit.

Figure 4: Instruction Word Contents

2 32-Bit CPU Design with Different Adders

2.1 Introduction

The objective of this project is to carry out the logical and physical synthesis of a 32-bit cpu using 4 separate types of adder circuits.

2.2 Carry Ripple Adder

An n-bit CRA is formed by concatenating n full adders in cascade with the carry output from one adder connected to the carry input of the next. This is shown in Figure 5.

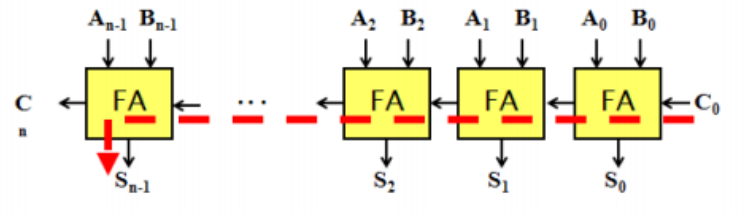

Figure 5: Carry Ripple Adder

2.2.1 RTL Simulation

Figures 6 and 7 show the display and simvision results of the RTL simulation. The detailed results are included with this report under the CRA directory in the file `stim_proj.out`.

```
Compiling source file "tb_cpu.v"
Compiling source file "cpu_CRA.v"

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CRA.v", 557: cra0(.sum(sum[7:0]), .c_out(c7)
          , .a(a[7:0]), .b(b[7:0]), .c_in(c_in))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CRA.v", 558: cra1(.sum(sum[15:8]), .c_out(
          c15), .a(a[15:8]), .b(b[15:8]), .c_in(c7))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CRA.v", 559: cra2(.sum(sum[23:16]), .c_out(
          c23), .a(a[23:16]), .b(b[23:16]), .c_in(c15))
Highest level modules:
stimulus

L30 "tb_cpu.v": $finish at simulation time 501
3 warnings
0 simulation events (use +profile or +listcounts option to count) + 21024 accelerated events
CPU time: 0.0 secs to compile + 0.1 secs to link + 0.8 secs in simulation
End of Tool:    VERILOG-XL      08.20.001-p   Dec  1, 2015  15:21:56
```

Figure 6: Display

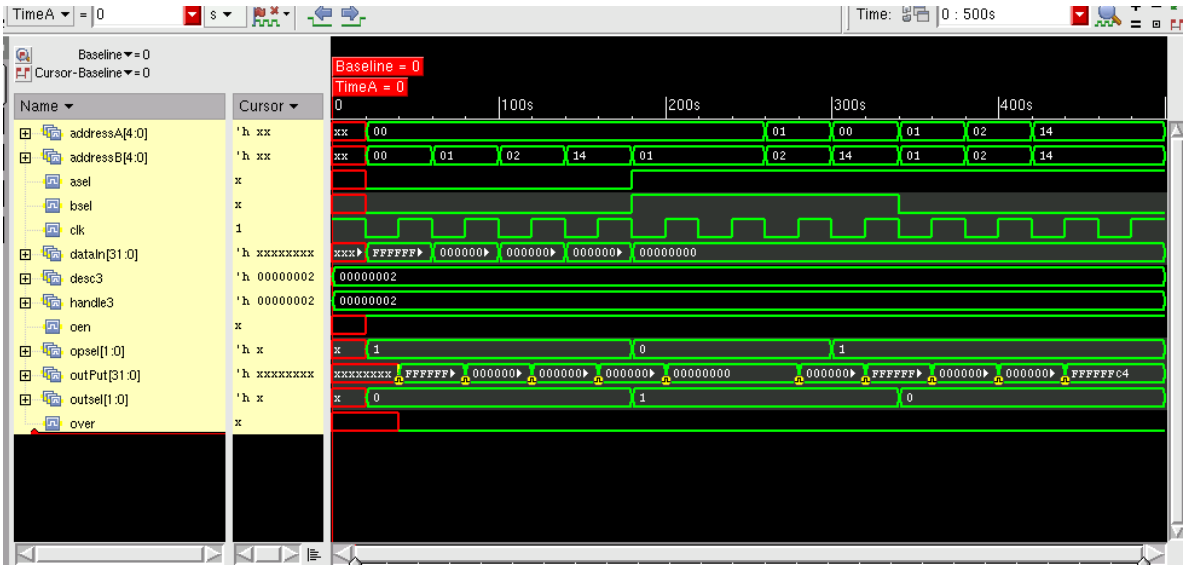

Figure 7: Simvision

2.2.2 Logic Synthesis and Post-Synthesis Simulation

Figures 8 and 9 show the display and simvision results of the simulation. The results are included under the CRA directory under the files `timing.rep` and `cell.rep`. They show that this design uses 48610.093084nm of area and has a slack time of 26.14 meaning that the clock rate can be increased.

```
Compiling source file "gscl45nm.v"
Compiling source file "tb_cpu.v"
Compiling source file "cpu.vh"
Highest level modules:
AOI21X1
BUFX4
CLKBUF1
CLKBUF2
CLKBUF3
DFFNEGX1
DFFSR
FAX1
HAX1
INVX2
INVX4
INVX8
LATCH
MUX2X1
NAND3X1
NOR3X1
OAI22X1
OR2X2
TBUFX1
stimulus


"gscl45nm.v", 299: Timing violation in stimulus.proj.\a/13/rc30/qout_reg
    $setup( negedge D:35999, posedge CLK:36000,  0.09 : 9 );

L30 "tb_cpu.v": $finish at simulation time 50100
0 simulation events (use +profile or +listcounts option to count) + 201694 accelerated events + 345566 timing check events
CPU time: 0.1 secs to compile + 0.3 secs to link + 0.2 secs in simulation
End of Tool:    VERILOG-XL      08.20.001-p   Dec  1, 2015  15:26:39
```

Figure 8: Display

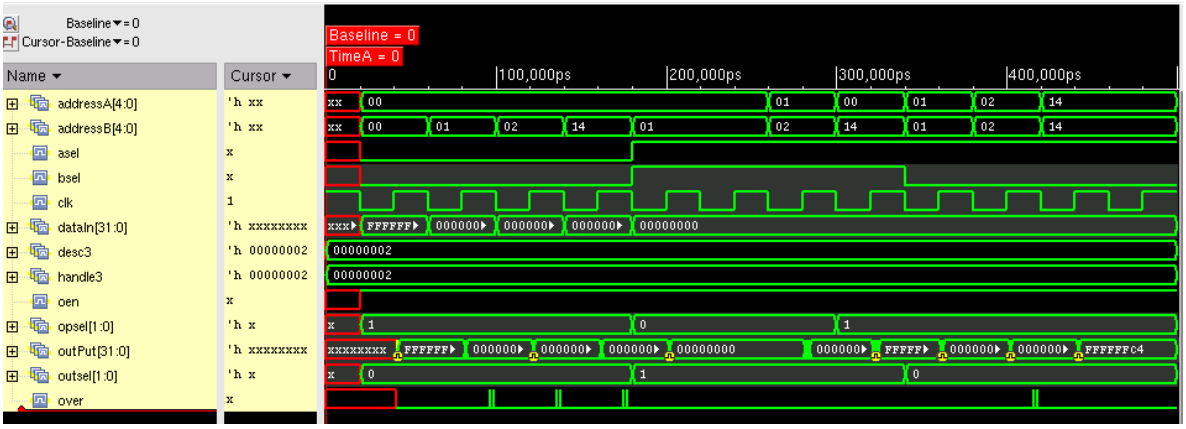

Figure 9: Simvision

2.2.3 Place & Route and Post-P&R Simulation

Figures 10 and 11 show the display and simvision results. The file `timing.rep.5.final` is located under the CRA directory included with this report. It shows that the slack time is about 17.458 which is an improvement from the post-synthesis design, but the clock can still be increased. The max clock rate was found to be $\approx$ 62 MHz.

```
Compiling source file "gscl45nm.v"
Compiling source file "tb_cpu.v"
Compiling source file "final.v"
Highest level modules:
AOI21X1
BUFX4
CLKBUF1
CLKBUF2
CLKBUF3
DFFNEGX1
DFFSR
FAX1
HAX1
INVX2
INVX4
LATCH
MUX2X1
NAND3X1
NOR3X1
OAI22X1
OR2X2
TBUFX1
stimulus


"gscl45nm.v", 299: Timing violation in stimulus.proj.\a/13/rc30/qout_reg
    $setup( negedge D:36017, posedge CLK:36018,  0.09 : 9 );

L30 "tb_cpu.v": $finish at simulation time 50100
0 simulation events (use +profile or +listcounts option to count) + 205645 accelerated events + 345566 timing check events
CPU time: 0.2 secs to compile + 0.2 secs to link + 0.3 secs in simulation
End of Tool:    VERILOG-XL      08.20.001-p   Dec  1, 2015  15:44:00
```

Figure 10: Display

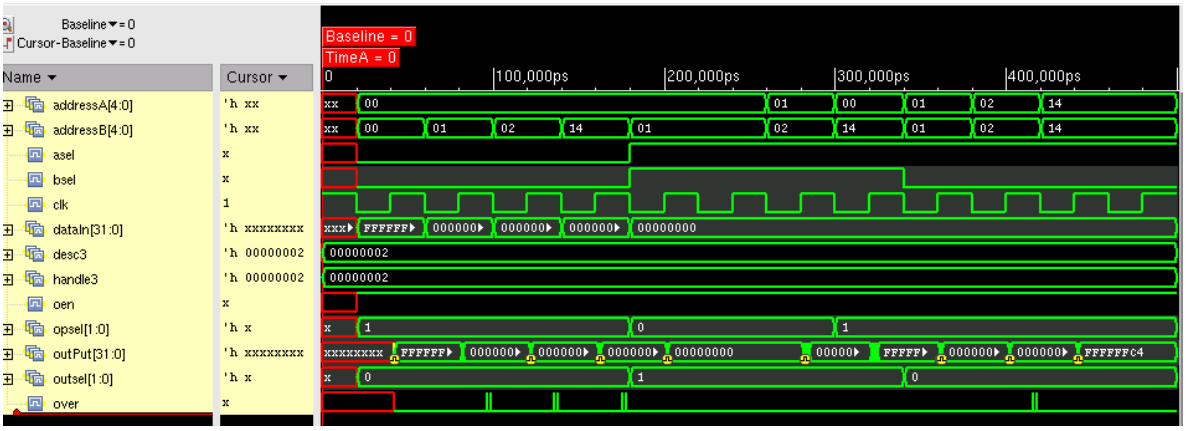

Figure 11: Simvision

2.2.4 Test Bench

Figures 12 and 13 show the display and simvision results. For a detailed analysis of the output please refer to the file `stim_proj_CRA.out` located under the CRA directory included with this report. Figures 14 and 15 show a detailed view of the ALU results.

```
Compiling source file "tb_test.v"
Compiling source file "cpu_CRA.v"

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CRA.v", 557: cra0(.sum(sum[7:0]), .c_out(c7)
          , .a(a[7:0]), .b(b[7:0]), .c_in(c_in))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CRA.v", 558: cra1(.sum(sum[15:8]), .c_out(
          c15), .a(a[15:8]), .b(b[15:8]), .c_in(c7))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CRA.v", 559: cra2(.sum(sum[23:16]), .c_out(
          c23), .a(a[23:16]), .b(b[23:16]), .c_in(c15))
Highest level modules:
stimulus

L30 "tb_test.v": $finish at simulation time 1101
3 warnings
0 simulation events (use +profile or +listcounts option to count) + 96988 accelerated events
CPU time: 0.0 secs to compile + 0.1 secs to link + 1.2 secs in simulation
End of Tool:    VERILOG-XL      08.20.001-p   Dec  1, 2015  15:46:34
```

Figure 12: RTL Display

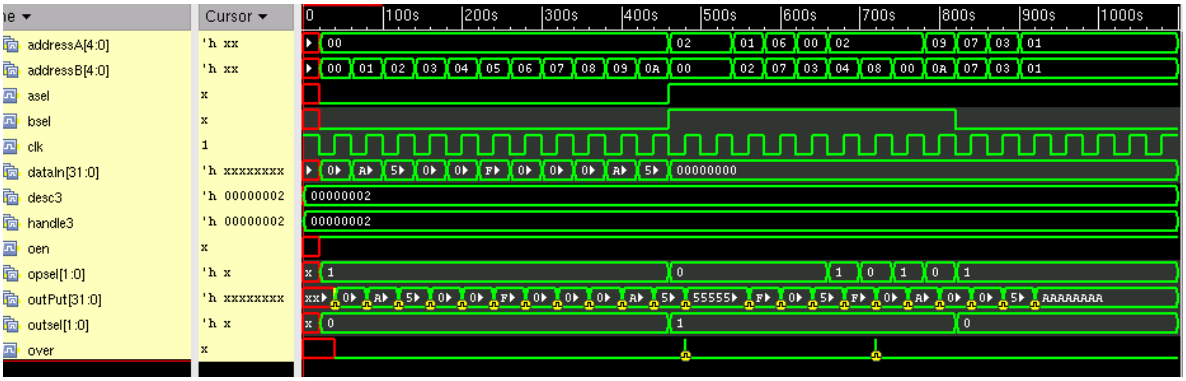

Figure 13: RTL Simvision

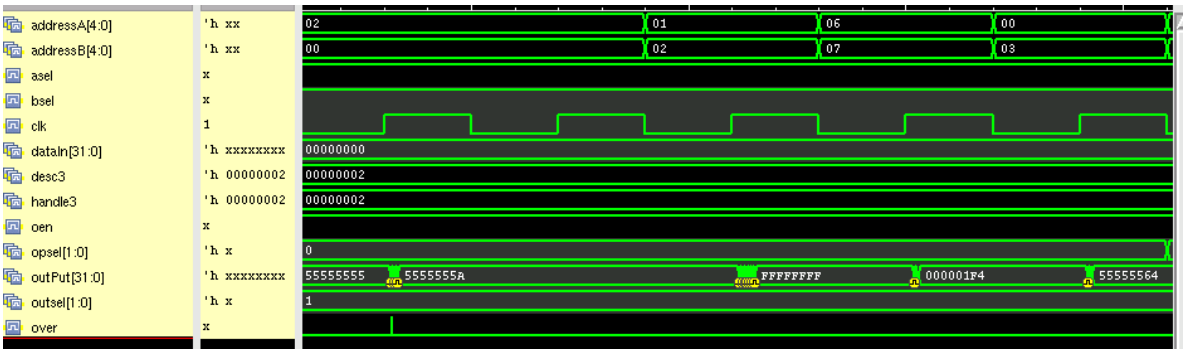

Figure 14: ALU Operations

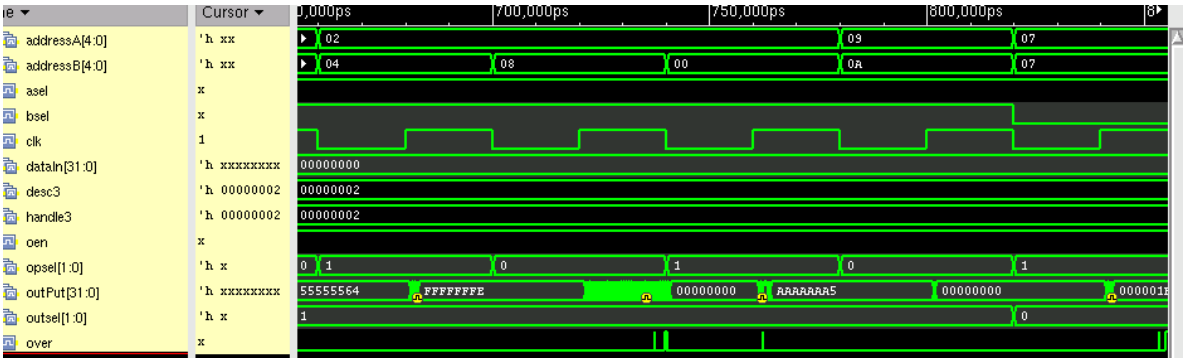

Figure 15: ALU Operations

2.3 Carry Lookahead Adder

Carry Lookahead circuits are special logic circuits that can dramatically reduce the time to perform addition at the price of more complex hardware. This is done by transforming the ripple carry design so that the carry logic over a fixed group of bits are reduced to two-level logic. This is shown in Figure 16.

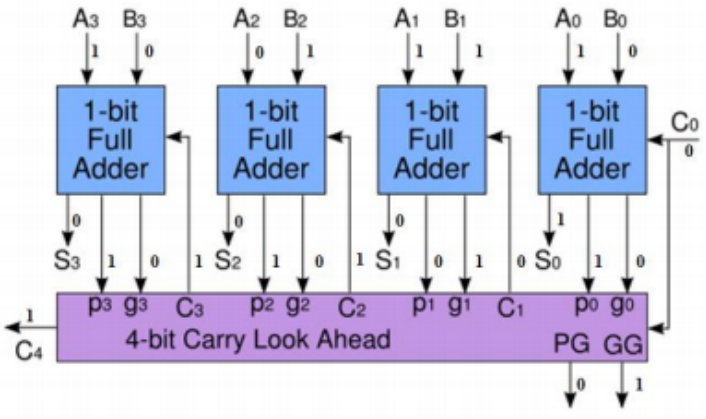

Figure 16: Carry Lookahead Adder

2.3.1 RTL Simulation

Figures 17 and 18 show the display and simvision results of the RTL simulation. The detailed results are included with this report under the CLA directory in the file `stim_proj.out`.

```
Compiling source file "tb_cpu.v"
Compiling source file "cpu_CLA.v"

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CLA.v", 636: CLA0(.s(s[15:0]), .a(a[15:0]),
          .b(b[15:0]), .c0(c0), .c16(c16), .p_16(p[0]), .
          g_16(g[0]))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CLA.v", 615: CLA0(.s(s[3:0]), .a(a[3:0]), .
          b(b[3:0]), .c0(c0), .c4(c4), .g_4(g[0]), .p_4(p[
          0]))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CLA.v", 616: CLA1(.s(s[7:4]), .a(a[7:4]), .
          b(b[7:4]), .c0(c4), .c4(c8), .g_4(g[1]), .p_4(p[
          1]))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CLA.v", 617: CLA2(.s(s[11:8]), .a(a[11:8]),
          .b(b[11:8]), .c0(c8), .c4(c12), .g_4(g[2]), .p_4(
          p[2]))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CLA.v", 615: CLA0(.s(s[3:0]), .a(a[3:0]), .
          b(b[3:0]), .c0(c0), .c4(c4), .g_4(g[0]), .p_4(p[
          0]))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CLA.v", 616: CLA1(.s(s[7:4]), .a(a[7:4]), .
          b(b[7:4]), .c0(c4), .c4(c8), .g_4(g[1]), .p_4(p[
          1]))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CLA.v", 617: CLA2(.s(s[11:8]), .a(a[11:8]),
          .b(b[11:8]), .c0(c8), .c4(c12), .g_4(g[2]), .p_4(
          p[2]))
Highest level modules:
stimulus

L30 "tb_cpu.v": $finish at simulation time 501
7 warnings
0 simulation events (use +profile or +listcounts option to count) + 19773 accelerated events
CPU time: 0.0 secs to compile + 0.1 secs to link + 0.8 secs in simulation
End of Tool:    VERILOG-XL      08.20.001-p   Dec  3, 2015  13:53:56
```

Figure 17: Display

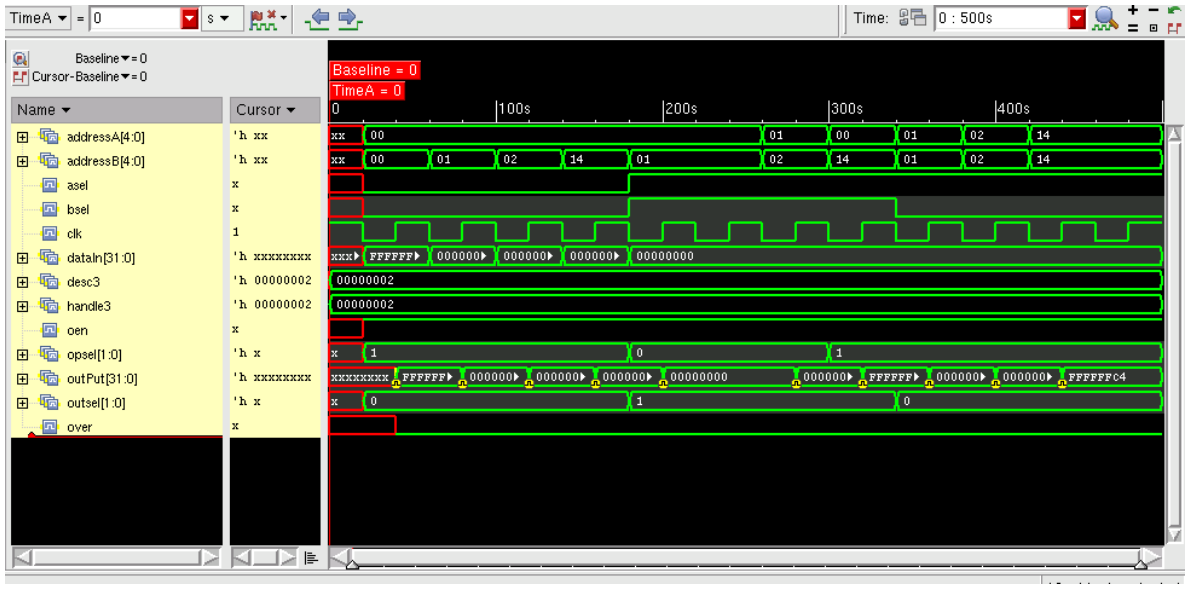

Figure 18: Simvision

2.3.2 Logic Synthesis and Post-Synthesis Simulation

Figures 8 and 9 show the display and simvision results of the simulation. The results are included under the CLA directory under the files `timing.rep` and `cell.rep`. They show that this design uses 48610.093084nm of area and has a slack time of 26.14 meaning that the clock rate can be increased.

```
Compiling source file "gscl45nm.v"
Compiling source file "tb_cpu.v"
Compiling source file "cpu.vh"
Highest level modules:
BUFX4
CLKBUF1
CLKBUF2
CLKBUF3
DFFNEGX1
DFFSR
FAX1
HAX1
INVX2
INVX4
INVX8
LATCH
MUX2X1
NAND3X1
NOR3X1
OAI22X1
OR2X2
TBUFX1
stimulus


"gscl45nm.v", 299: Timing violation in stimulus.proj.\a/13/rc30/qout_reg
    $setup( negedge D:35999, posedge CLK:36000,  0.09 : 9 );

L30 "tb_cpu.v": $finish at simulation time 50100
0 simulation events (use +profile or +listcounts option to count) + 199395 accelerated events + 345566 timing check events
CPU time: 0.1 secs to compile + 0.3 secs to link + 0.2 secs in simulation
End of Tool:    VERILOG-XL      08.20.001-p   Dec  1, 2015  15:04:30
```

Figure 19: Display

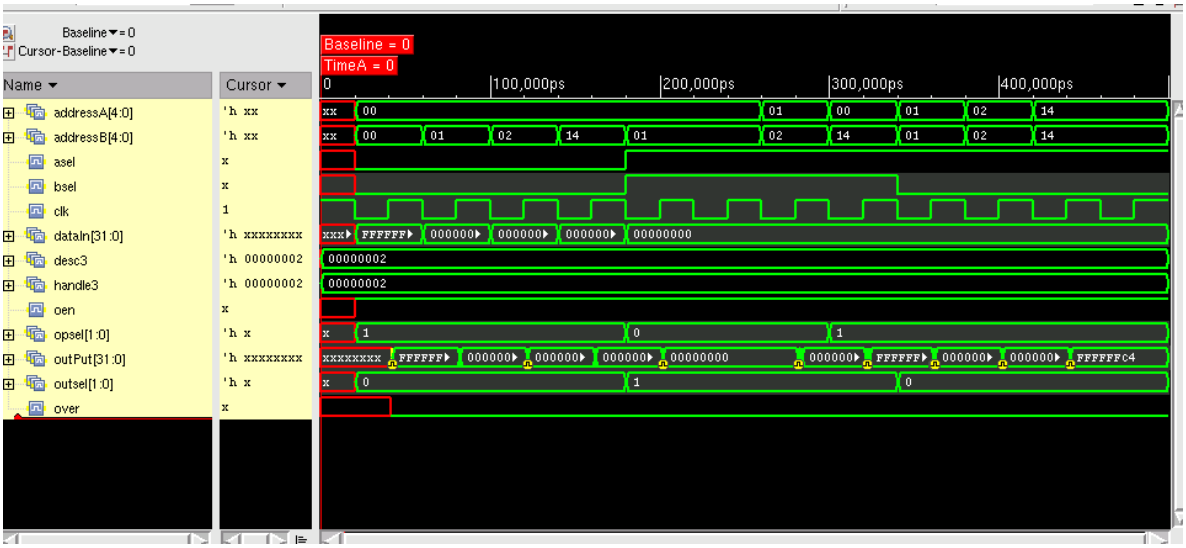

Figure 20: Simvision

2.3.3 Place & Route and Post-P&R Simulation

Figures 21 and 22 show the display and simvision results. The file `timing.rep.5.final` is located under the CLA directory included with this report. It shows that the slack

time is about 17.921 which is an improvement from the post-synthesis design, but the clock can still be increased. The max clock rate was found to be ≈ 66 MHz.

```
Compiling source file "gscl45nm.v"
Compiling source file "tb_cpu.v"
Compiling source file "final.v"
Highest level modules:
BUFX4
CLKBUF1
CLKBUF2
CLKBUF3
DFFNEGX1
DFFSR
FAX1
HAX1
INVX2
LATCH
MUX2X1
NAND3X1
NOR3X1
OAI22X1
OR2X2
TBUFX1
stimulus


"gscl45nm.v", 299: Timing violation in stimulus.proj.\a/13/rc30/qout_reg
    $setup( negedge D:36017, posedge CLK:36018,  0.09 : 9 );

L30 "tb_cpu.v": $finish at simulation time 50100
0 simulation events (use +profile or +listcounts option to count) + 203917 accelerated events + 345566 timing check events
CPU time: 0.2 secs to compile + 0.2 secs to link + 0.3 secs in simulation
End of Tool:    VERILOG-XL      08.20.001-p   Dec  1, 2015  15:11:54
```

Figure 21: Display

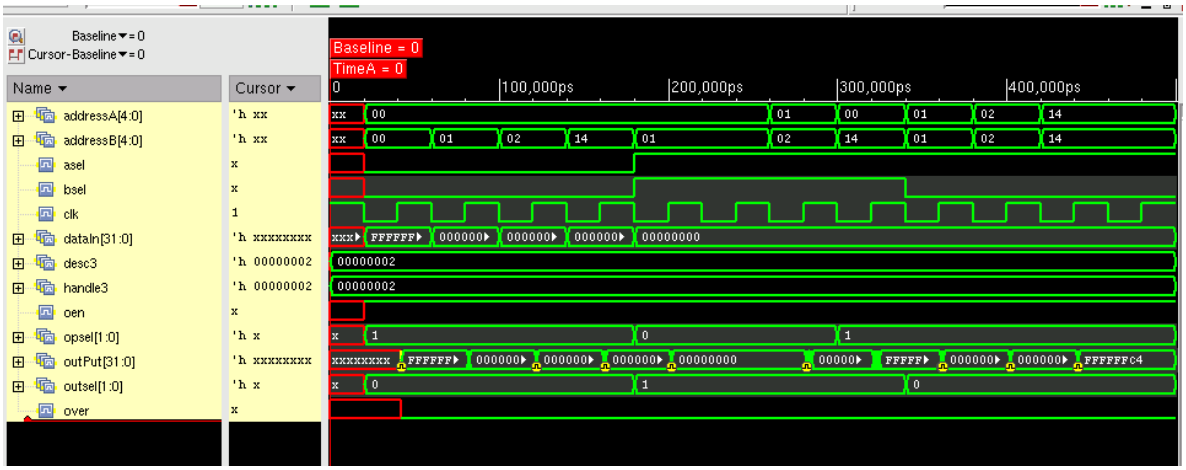

Figure 22: Simvision

2.3.4 Test Bench

Figures 23 and 24 show the display and simvision results. For a detailed analysis of the output please refer to the file `stim_proj_CLA.out` located under the CRA directory included with this report. Figures 25 and 26 show a detailed view of the ALU results.

```
Compiling source file "tb_test.v"
Compiling source file "cpu_CLA.v"

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CLA.v", 636: CLA0(.s(s[15:0]), .a(a[15:0]),
          .b(b[15:0]), .c0(c0), .c16(c16), .p_16(p[0]), .
          g_16(g[0]))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CLA.v", 615: CLA0(.s(s[3:0]), .a(a[3:0]), .
          b(b[3:0]), .c0(c0), .c4(c4), .g_4(g[0]), .p_4(p[
          0]))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CLA.v", 616: CLA1(.s(s[7:4]), .a(a[7:4]), .
          b(b[7:4]), .c0(c4), .c4(c8), .g_4(g[1]), .p_4(p[
          1]))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CLA.v", 617: CLA2(.s(s[11:8]), .a(a[11:8]),
          .b(b[11:8]), .c0(c8), .c4(c12), .g_4(g[2]), .p_4(
          p[2]))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CLA.v", 615: CLA0(.s(s[3:0]), .a(a[3:0]), .
          b(b[3:0]), .c0(c0), .c4(c4), .g_4(g[0]), .p_4(p[
          0]))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CLA.v", 616: CLA1(.s(s[7:4]), .a(a[7:4]), .
          b(b[7:4]), .c0(c4), .c4(c8), .g_4(g[1]), .p_4(p[
          1]))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CLA.v", 617: CLA2(.s(s[11:8]), .a(a[11:8]),
          .b(b[11:8]), .c0(c8), .c4(c12), .g_4(g[2]), .p_4(
          p[2]))
Highest level modules:
stimulus

L30 "tb_test.v": $finish at simulation time 1101
7 warnings
0 simulation events (use +profile or +listcounts option to count) + 94165 accelerated events
CPU time: 0.0 secs to compile + 0.1 secs to link + 1.3 secs in simulation
End of Tool:    VERILOG-XL      08.20.001-p   Dec  1, 2015  15:13:20
```

Figure 23: RTL Display

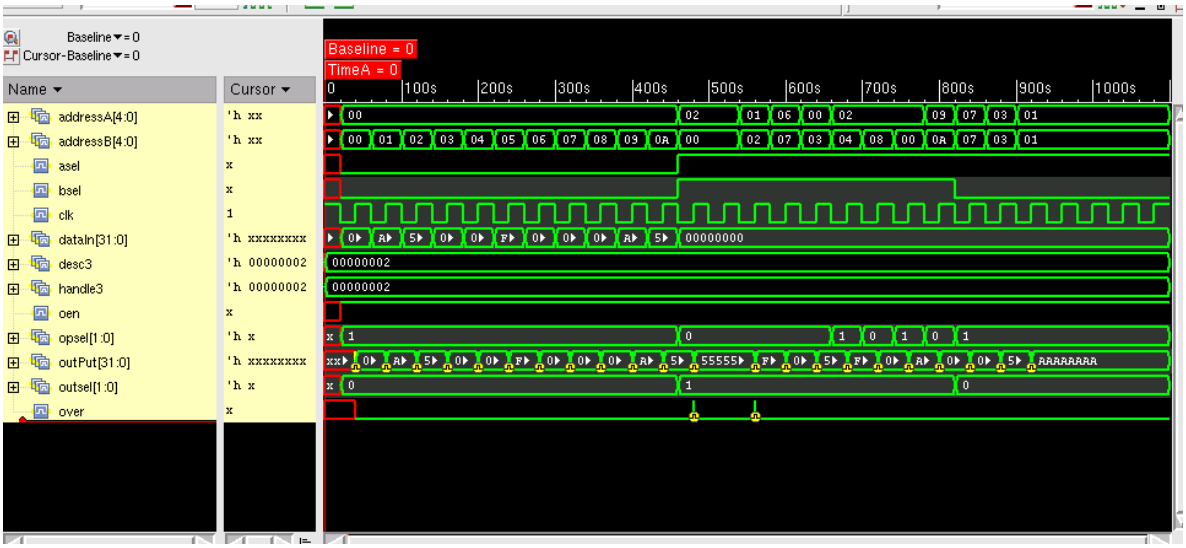

Figure 24: RTL Simvision

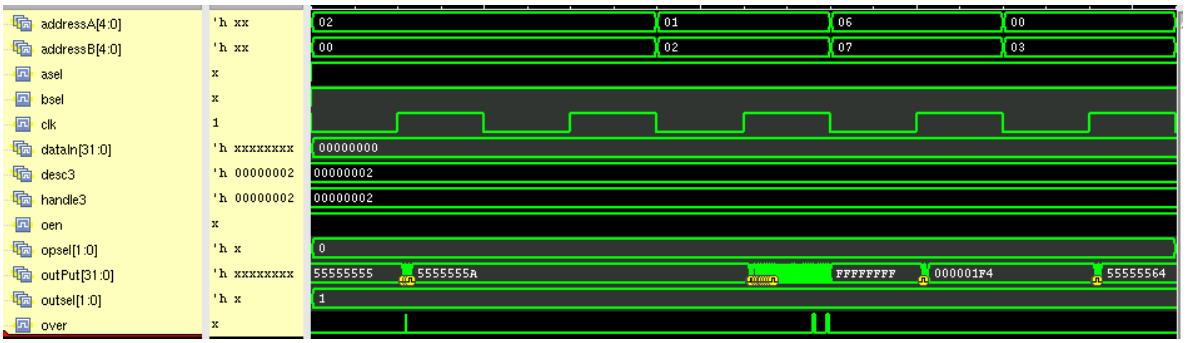

Figure 25: ALU 1

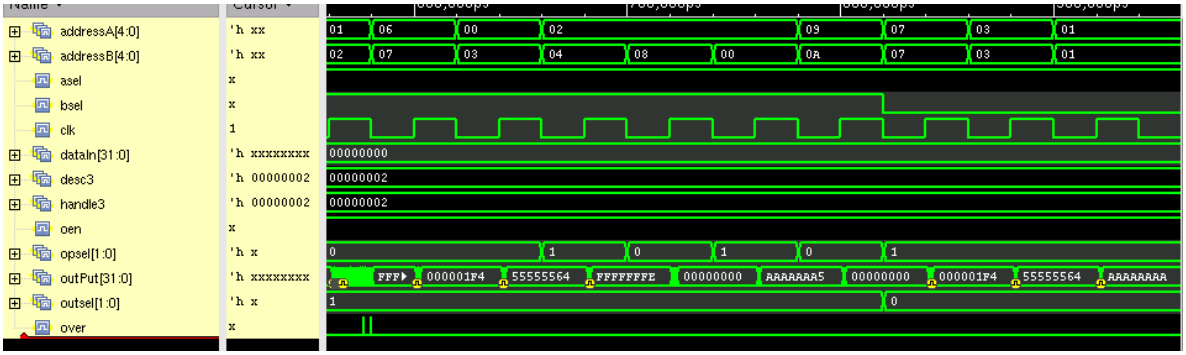

Figure 26: ALU 2

2.4 Carry Skip Adder

In the Carry Skip Adder, the operands are divided into blocked of r bits blocks. In each block, a ripple carry adder can be utilized to produce the sum bits and a carryout bit for the block. This is shown in Figure 27.

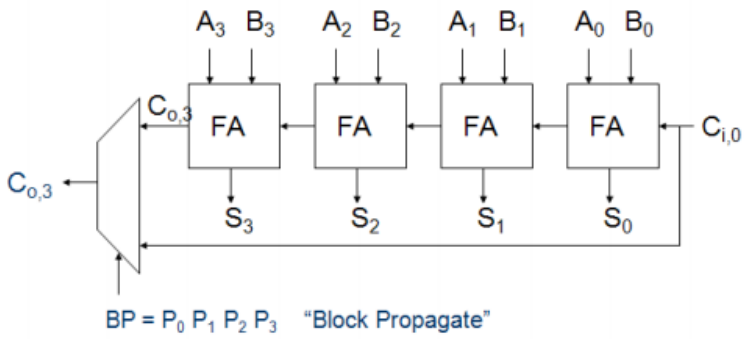

Figure 27: Carry Skip Adder

2.4.1 RTL Simulation

Figures 28 and 29 show the display and simvision results of the RTL simulation. The detailed results are included with this report under the CSA directory in the file `stim_proj.out`.

```
Compiling source file "tb_cpu.v"
Compiling source file "cpu_CSA.v"

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSA.v", 598: cs0_15(.s(s[15:0]), .cout(c), .
          a(a[15:0]), .b(b[15:0]), .cin(cin))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSA.v", 582: cs0_3(.s(s[3:0]), .cout(c[0]),
          .a(a[3:0]), .b(b[3:0]), .cin(cin))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSA.v", 583: cs4_7(.s(s[7:4]), .cout(c[1]),
          .a(a[7:4]), .b(b[7:4]), .cin(c[0]))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSA.v", 584: cs8_11(.s(s[11:8]), .cout(c[2])
          , .a(a[11:8]), .b(b[11:8]), .cin(c[1]))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSA.v", 582: cs0_3(.s(s[3:0]), .cout(c[0]),
          .a(a[3:0]), .b(b[3:0]), .cin(cin))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSA.v", 583: cs4_7(.s(s[7:4]), .cout(c[1]),
          .a(a[7:4]), .b(b[7:4]), .cin(c[0]))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSA.v", 584: cs8_11(.s(s[11:8]), .cout(c[2])
          , .a(a[11:8]), .b(b[11:8]), .cin(c[1]))
Highest level modules:
stimulus

L30 "tb_cpu.v": $finish at simulation time 501
7 warnings
0 simulation events (use +profile or +listcounts option to count) + 21065 accele
rated events
CPU time: 0.0 secs to compile + 0.1 secs to link + 0.7 secs in simulation
End of Tool:    VERILOG-XL      08.20.001-p   Dec  1, 2015  13:24:08
```

Figure 28: Display

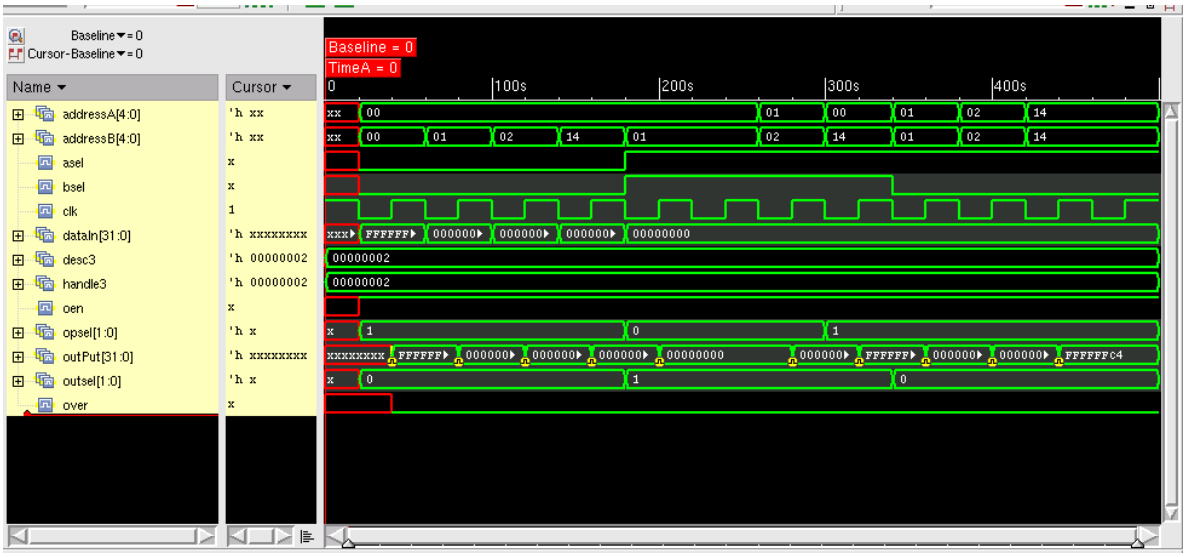

Figure 29: Simvision

2.4.2 Logic Synthesis and Post-Synthesis Simulation

Figures 30 and 31 show the display and simvision results of the simulation. The results are included under the CSA directory under the files `timing.rep` and `cell.rep`. They show that this design uses 48878.063377nm of area and has a slack time of 26.14 meaning that the clock rate can be increased.

```
Compiling source file "gscl45nm.v"
Compiling source file "tb_cpu.v"
Compiling source file "cpu.vh"
Highest level modules:
AOI21X1
BUFX4
CLKBUF1
CLKBUF2
CLKBUF3
DFFNEGX1
DFFSR
FAX1
HAX1
INVX2
INVX4
INVX8
LATCH
MUX2X1
NOR3X1
OAI22X1
OR2X2
TBUFX1
stimulus


"gscl45nm.v", 299: Timing violation in stimulus.proj.\a/13/rc30/qout_reg
    $setup( negedge D:35999, posedge CLK:36000,  0.09 : 9 );

L30 "tb_cpu.v": $finish at simulation time 50100
0 simulation events (use +profile or +listcounts option to count) + 203346 accelerated events + 345546 timing check events
CPU time: 0.1 secs to compile + 0.3 secs to link + 0.2 secs in simulation
End of Tool:    VERILOG-XL      08.20.001-p   Dec  3, 2015  15:25:26
```

Figure 30: Display

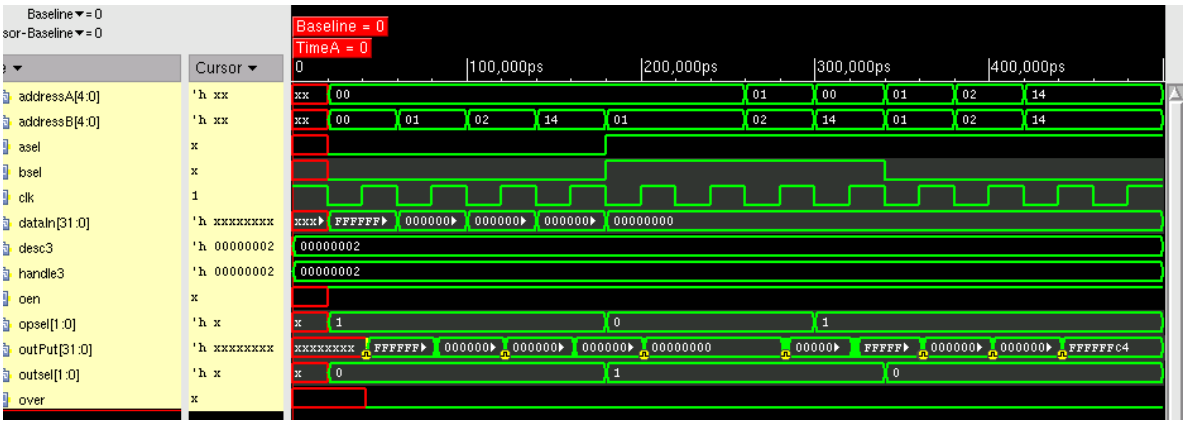

Figure 31: Simvision

2.4.3 Place & Route and Post-P&R Simulation

Figures 32 and 33 show the display and simvision results. The file `timing.rep.5.final` is located under the CSA directory included with this report. It shows that the slack time is about 18.318 which is an improvement from the post-synthesis design, but the clock can still be increased. The max clock rate was found to be $\approx$ 66 MHz.

```
Compiling source file "gscl45nm.v"
Compiling source file "tb_cpu.v"
Compiling source file "final.v"
Highest level modules:
AOI21X1
BUFX4
CLKBUF1
CLKBUF2
CLKBUF3
DFFNEGX1
DFFSR
FAX1
HAX1
INVX2
LATCH
MUX2X1
NOR3X1
OAI22X1
OR2X2
TBUFX1
stimulus


"gscl45nm.v", 299: Timing violation in stimulus.proj.\a/13/rc30/qout_reg
    $setup( negedge D:36017, posedge CLK:36018,  0.09 : 9 );

L30 "tb_cpu.v": $finish at simulation time 50100
0 simulation events (use +profile or +listcounts option to count) + 207877 acce
erated events + 345558 timing check events
CPU time: 0.2 secs to compile + 0.2 secs to link + 0.3 secs in simulation
End of Tool:    VERILOG-XL      08.20.001-p   Dec  1, 2015  13:04:24
```

Figure 32: Display

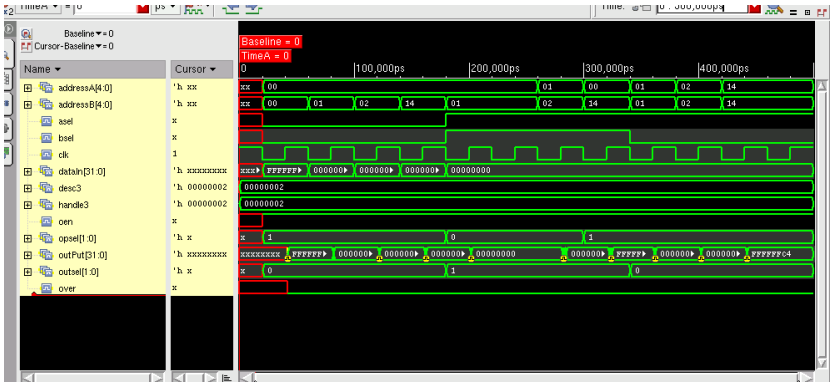

Figure 33: Simvision

2.4.4 Test Bench

Figures 34 - 36 show the display and simvision results. For a detailed analysis of the output please refer to the file `stim_proj_CSA.out` located under the CSA directory included with this report.

```
Compiling source file "tb_test.v"
Compiling source file "cpu_CSA.v"

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSA.v", 598: cs0_15(.s(s[15:0]), .cout(c), .
          a(a[15:0]), .b(b[15:0]), .cin(cin))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSA.v", 582: cs0_3(.s(s[3:0]), .cout(c[0]),
          .a(a[3:0]), .b(b[3:0]), .cin(cin))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSA.v", 583: cs4_7(.s(s[7:4]), .cout(c[1]),
          .a(a[7:4]), .b(b[7:4]), .cin(c[0]))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSA.v", 584: cs8_11(.s(s[11:8]), .cout(c[2])
          , .a(a[11:8]), .b(b[11:8]), .cin(c[1]))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSA.v", 582: cs0_3(.s(s[3:0]), .cout(c[0]),
          .a(a[3:0]), .b(b[3:0]), .cin(cin))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSA.v", 583: cs4_7(.s(s[7:4]), .cout(c[1]),
          .a(a[7:4]), .b(b[7:4]), .cin(c[0]))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSA.v", 584: cs8_11(.s(s[11:8]), .cout(c[2])
          , .a(a[11:8]), .b(b[11:8]), .cin(c[1]))
Highest level modules:
stimulus

L30 "tb_test.v": $finish at simulation time 501
7 warnings
0 simulation events (use +profile or +listcounts option to count) + 15982 accelerated events
CPU time: 0.0 secs to compile + 0.1 secs to link + 0.7 secs in simulation
End of Tool:    VERILOG-XL      08.20.001-p   Dec  1, 2015  13:45:58
```

Figure 34: Display

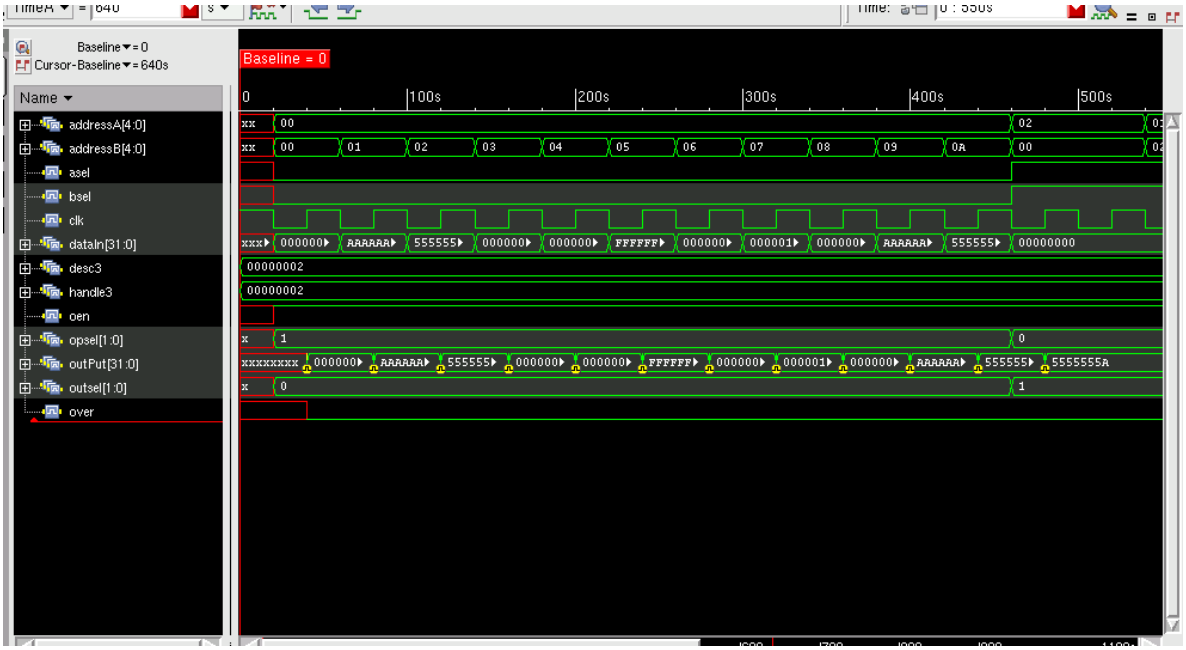

Figure 35: Simvision 1

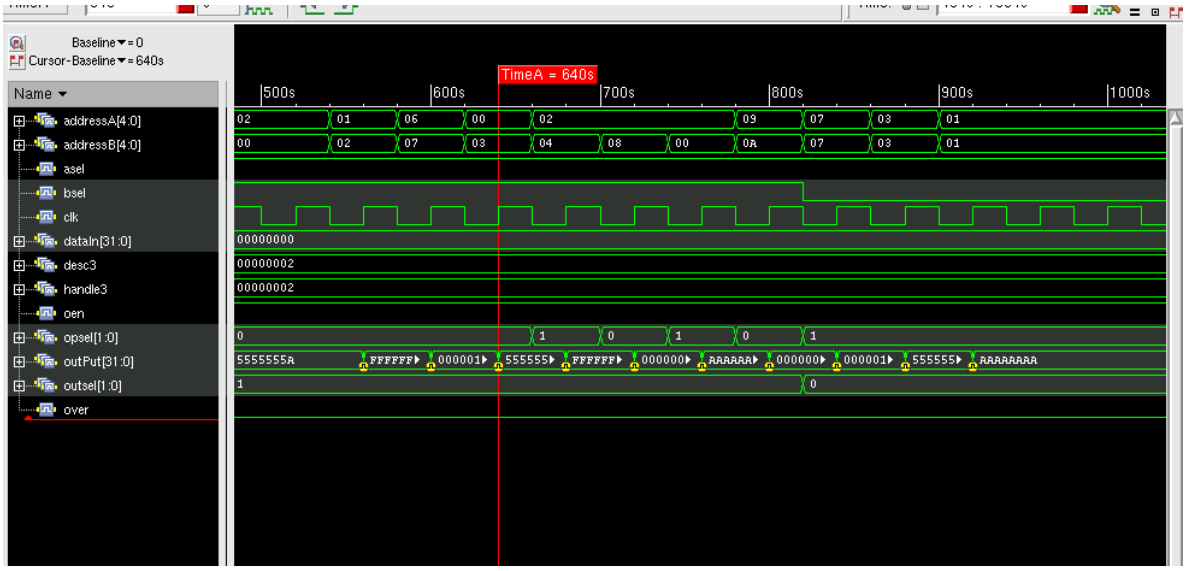

Figure 36: Simvision 2

2.5 Carry Select Adder

The Carry Select Adder divides the operands to be added into r bit blocks similar to the Carry Skip Adder. For each block, two r-bit Ripple Carry Adders operate in parallel to form two sets of sum bits and carry out signals. Each Ripple Carry Adder has two

sets of hard-coded carry-in signals. One Ripple Carry Adder has a carry-in of 0, while the other has a carry-in of 1. This is shown in Figure 37.

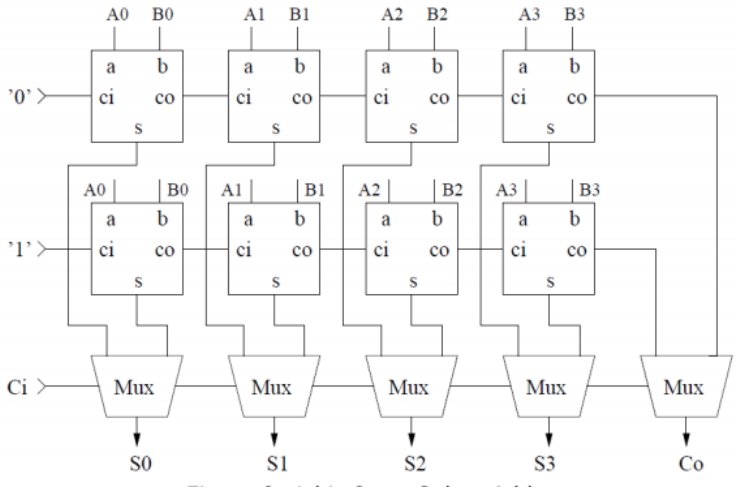

Figure 37: Carry Select Adder

2.5.1 RTL Simulation

Figures 38 and 39 show the display and simvision results of the RTL simulation. The detailed results are included with this report under the CSeA directory in the file `stim_proj.out`.

```
Compiling source file "tb_cpu.v"
Compiling source file "cpu_CSeA.v"

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSeA.v", 606: cse1(.s(s[15:0]), .cout(c), .
          a(a[15:0]), .b(b[15:0]), .cin(cin))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSeA.v", 591: c1(.s(s[3:0]), .cout(c[1]), .
          a(a[3:0]), .b(b[3:0]), .cin(cin))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSeA.v", 592: c2(.s(s[7:4]), .cout(c[2]), .
          a(a[7:4]), .b(b[7:4]), .cin(c[1]))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSeA.v", 593: c3(.s(s[11:8]), .cout(c[3]), .
          a(a[11:8]), .b(b[11:8]), .cin(c[2]))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSeA.v", 591: c1(.s(s[3:0]), .cout(c[1]), .
          a(a[3:0]), .b(b[3:0]), .cin(cin))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSeA.v", 592: c2(.s(s[7:4]), .cout(c[2]), .
          a(a[7:4]), .b(b[7:4]), .cin(c[1]))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSeA.v", 593: c3(.s(s[11:8]), .cout(c[3]), .
          a(a[11:8]), .b(b[11:8]), .cin(c[2]))
Highest level modules:
stimulus

L30 "tb_cpu.v": $finish at simulation time 501
7 warnings
0 simulation events (use +profile or +listcounts option to count) + 21936 accelerated events
CPU time: 0.0 secs to compile + 0.1 secs to link + 0.8 secs in simulation
End of Tool:    VERILOG-XL      08.20.001-p   Dec  1, 2015  16:01:59
```

Figure 38: Display

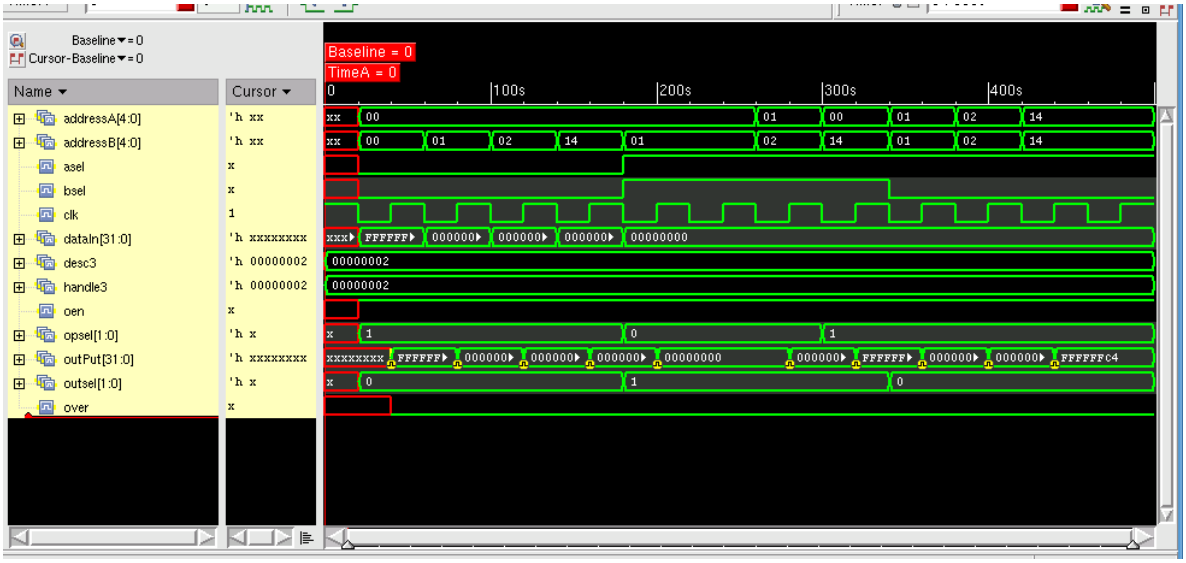

Figure 39: Simvision

2.5.2 Logic Synthesis and Post-Synthesis Simulation

Figures 40 and 41 show the display and simvision results of the simulation. The results are included under the CSeA directory under the files `timing.rep` and `cell.rep`. They

show that this design uses 49150.726672nm of area and has a slack time of 26.14 meaning that the clock rate can be increased.

```
Compiling source file "gscl45nm.v"
Compiling source file "tb_cpu.v"
Compiling source file "cpu.vh"
Highest level modules:
AOI21X1
BUFX4
CLKBUF1
CLKBUF2
CLKBUF3
DFFNEGX1
DFFSR
FAX1
HAX1
INVX2
INVX4
INVX8
LATCH
MUX2X1
NAND3X1
NOR3X1
OAI22X1
OR2X2
TBUFX1
stimulus


"gscl45nm.v", 299: Timing violation in stimulus.proj.\a/13/rc30/qout_reg
    $setup( negedge D:35999, posedge CLK:36000,  0.09 : 9 );

L30 "tb_cpu.v": $finish at simulation time 50100
0 simulation events (use +profile or +listcounts option to count) + 207669 accelerated events + 345638 timing check events
CPU time: 0.1 secs to compile + 0.3 secs to link + 0.2 secs in simulation
End of Tool:    VERILOG-XL      08.20.001-p   Dec  1, 2015  16:09:01
```

Figure 40: Display

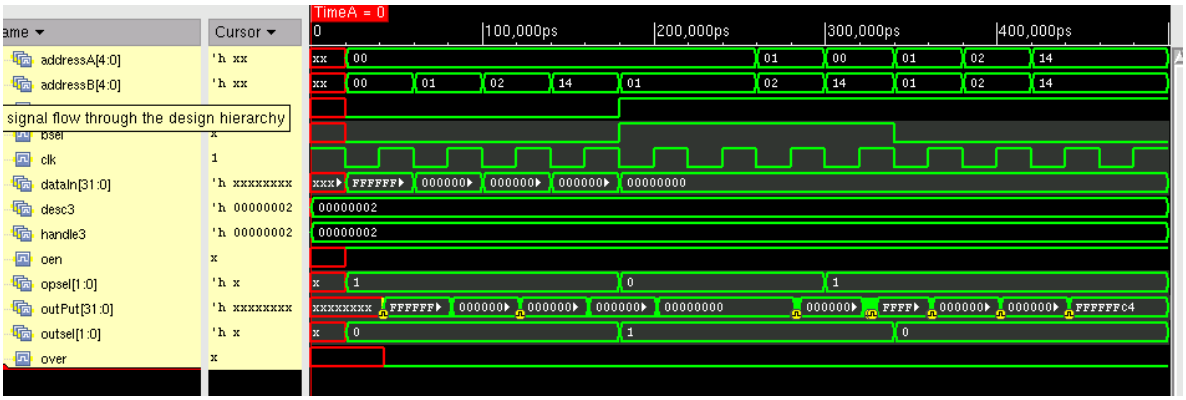

Figure 41: Simvision

2.5.3 Place & Route and Post-P&R Simulation

Figures 42 and 43 show the display and simvision results. The file `timing.rep.5.final` is located under the CSeA directory included with this report. It shows that the slack time is about 18.144 which is an improvement from the post-synthesis design, but the clock can still be increased. The max clock rate was found to be $\approx$ 66 MHz.

```
Compiling source file "gscl45nm.v"
Compiling source file "tb_cpu.v"
Compiling source file "final.v"
Highest level modules:
AOI21X1
BUFX4
CLKBUF1
CLKBUF2
CLKBUF3
DFFNEGX1
DFFSR
FAX1
HAX1
INVX2
INVX4
LATCH
MUX2X1
NAND3X1
NOR3X1
OAI22X1
OR2X2
TBUFX1
stimulus


"gscl45nm.v", 299: Timing violation in stimulus.proj.\a/13/rc30/qout_reg
    $setup( negedge D:36017, posedge CLK:36018,  0.09 : 9 );

L30 "tb_cpu.v": $finish at simulation time 50100
0 simulation events (use +profile or +listcounts option to count) + 211984 accelerated events + 345638 timing check events
CPU time: 0.2 secs to compile + 0.2 secs to link + 0.3 secs in simulation
End of Tool:    VERILOG-XL      08.20.001-p   Dec  1, 2015  16:18:14
```

Figure 42: Display

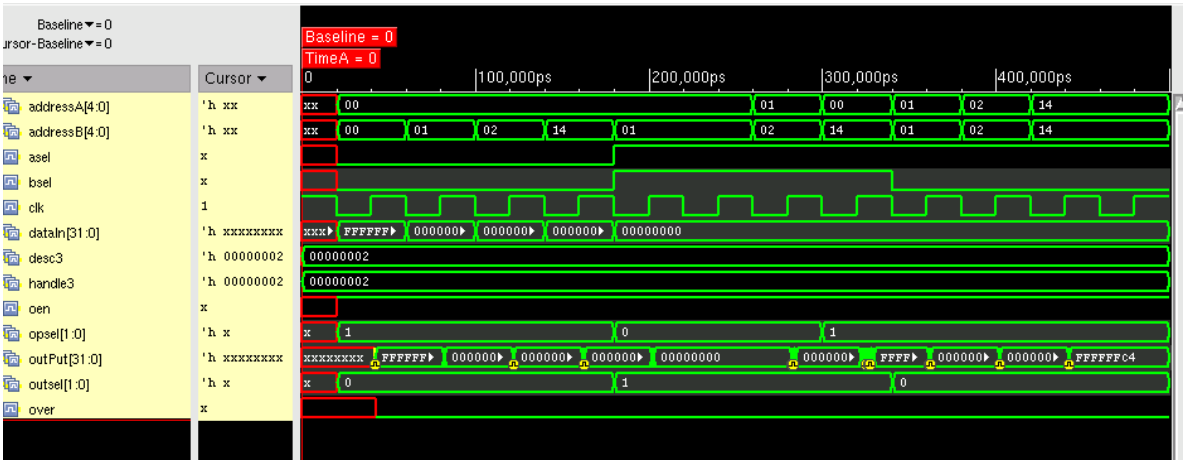

Figure 43: Simvision

2.5.4 Test Bench

Figures 44 and 45 show the display and simvision results. For a detailed analysis of the output please refer to the file `stim_proj_CSeA.out` located under the CSeA directory included with this report. Figures 46 and 47 show a detailed view of the ALU results.

```
Compiling source file "tb_test.v"
Compiling source file "cpu_CSeA.v"

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSeA.v", 606: cse1(.s(s[15:0]), .cout(c), .
          a(a[15:0]), .b(b[15:0]), .cin(cin))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSeA.v", 591: c1(.s(s[3:0]), .cout(c[1]), .
          a(a[3:0]), .b(b[3:0]), .cin(cin))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSeA.v", 592: c2(.s(s[7:4]), .cout(c[2]), .
          a(a[7:4]), .b(b[7:4]), .cin(c[1]))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSeA.v", 593: c3(.s(s[11:8]), .cout(c[3]), .
          a(a[11:8]), .b(b[11:8]), .cin(c[2]))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSeA.v", 591: c1(.s(s[3:0]), .cout(c[1]), .
          a(a[3:0]), .b(b[3:0]), .cin(cin))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSeA.v", 592: c2(.s(s[7:4]), .cout(c[2]), .
          a(a[7:4]), .b(b[7:4]), .cin(c[1]))

Warning!  Too few module port connections                    [Verilog-TFNPC]
          "cpu_CSeA.v", 593: c3(.s(s[11:8]), .cout(c[3]), .
          a(a[11:8]), .b(b[11:8]), .cin(c[2]))
Highest level modules:
stimulus

L30 "tb_test.v": $finish at simulation time 1101
7 warnings
0 simulation events (use +profile or +listcounts option to count) + 99444 accelerated events
CPU time: 0.0 secs to compile + 0.1 secs to link + 1.3 secs in simulation
End of Tool:    VERILOG-XL      08.20.001-p   Dec  1, 2015  16:20:02
```

Figure 44: Display

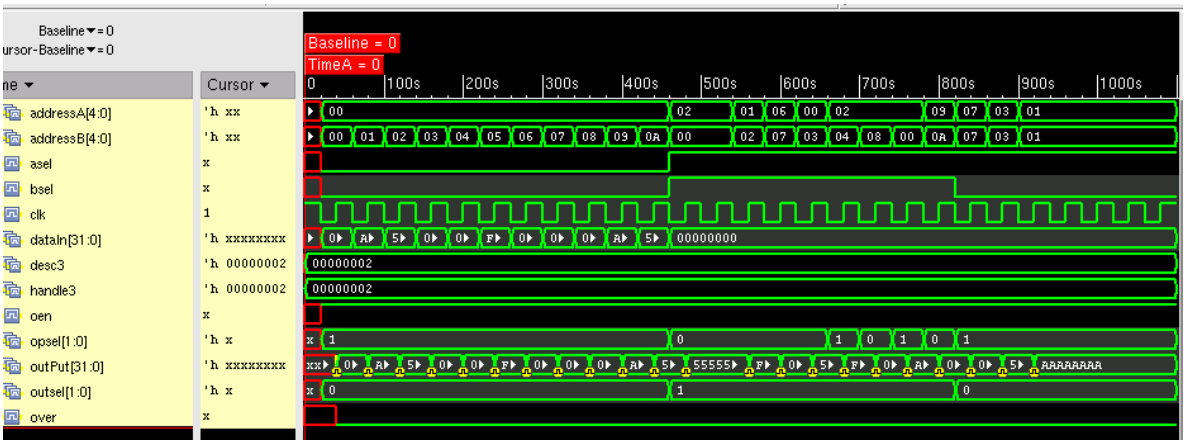

Figure 45: Simvision

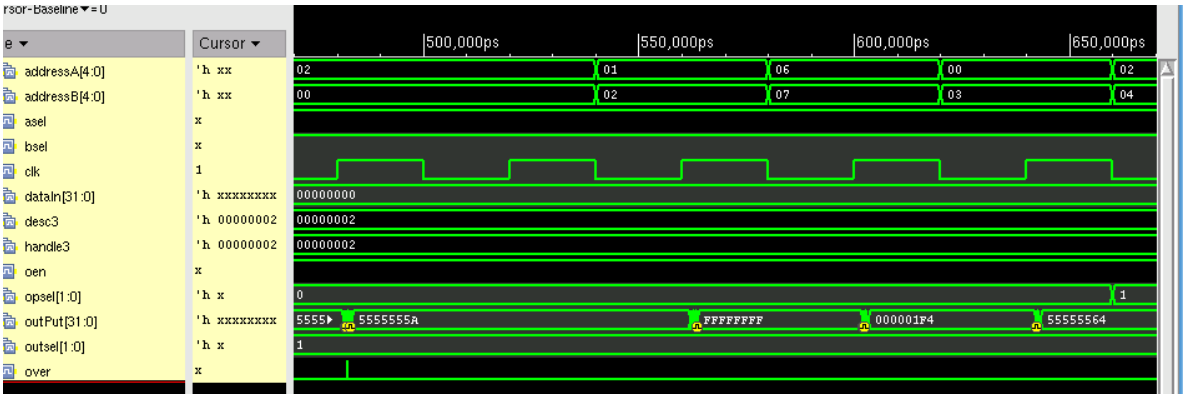

Figure 46: ALU Operations 1

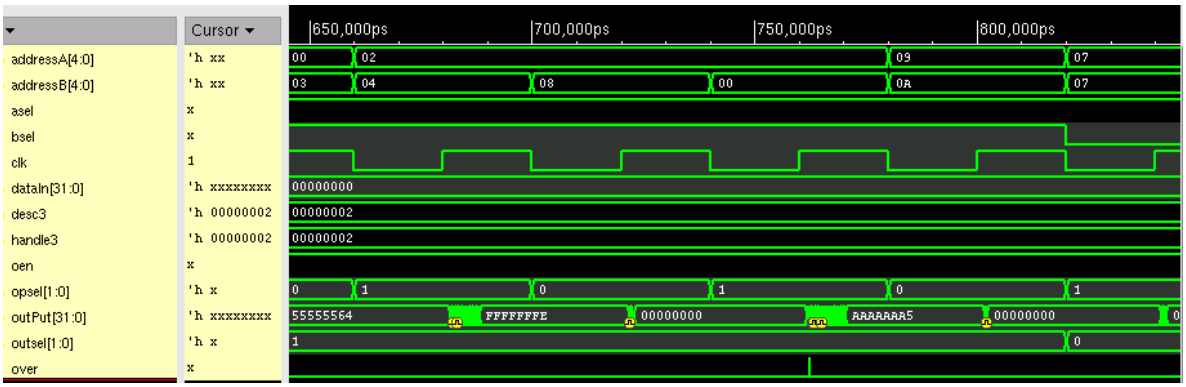

Figure 47: ALU Operations 2

2.6 Delay Comparison

Operation	CRA	CLA	CSA	CSeA
5555_5555 + 0000_0005	23.1ns	23.2ns	23.6ns	23.6ns
AAAA_AAAA + 5555_5555	25.2ns	40.2ns	28.1ns	23.9ns
0000_00C8 + 0000_012C	22.9ns	22.5ns	23.4ns	23.4ns
5555_555A + 0000_000A	23.1ns	22.8ns	22.9ns	23.7ns
FFFF_FFFF - 0000_0001	22.9ns	22.8ns	23.3ns	28.5ns
FFFF_FFFF + 0000_0001	40.8ns	22.9ns	30.4ns	23.1ns
FFFF_FFFF - 5555_555A	24.1ns	22.2ns	24.6ns	30.1ns
AAAA_AAAB + 5555_5555	22.1ns	22.1ns	25.2ns	23.2ns

Table 1: Width of 180nm

Overall each adder performed reasonably well. It should be noted that some adders had longer delays on some operations than others. This has to due with how they each

algorithmically perform operations. It should be noted that the Carry Skip Adder and Carry Select Adder performed the best on average.

2.7 New Test Bench Code

This code for `tb_test.v` is displayed below and is also included under the directory code included with this report.

```
1  module stimulus;
2
3  parameter T = 20;
4
5  reg [31:0] dataIn;
6  reg [1:0] opsel, outsel;
7  reg [4:0] addressA, addressB;
8  reg asel, bsel, oen, clk;
9
10 wire [31:0] outPut;
11 wire over;
12
13
14 integer handle3,desc3;
15
16 // Instantiate the design block counter
17 cpu proj(addressA, addressB, dataIn, asel,bsel, clk, opsel,
       outsel, oen, outPut, over);
18 initial
19 begin
20 clk = 1'b1;
21 forever #T  clk = ~clk;
22 end
23
24 initial
25  begin
26 handle3 = $fopen ("stim_proj.out");
27   $shm_open("shm.db",1);
28   $shm_probe(stimulus,"AS");
29   #1100  $shm_close();
30   #1 $finish;
31   end
32
33
34 always
35 begin
36 desc3=handle3;
```

```
37 #1 $fdisplay(desc3, $time,"clk=%b, addressA=%d, addressB=%d,
     dataIn=%d, opsel=%d, outsel=%d, asel=%d, bsel=%d, oen=%d, OUT
     =%d", clk, addressA, addressB, dataIn, opsel, outsel,asel,
     bsel, oen, outPut);
38
39 end
40
41  // Stimulate the Input Signals
42 initial
43  begin
44
45 // 1. store (0000_0005) in [0]
46
47 #T
48
49 addressA = 5'b00000;
50
51 addressB = 5'b00000;
52
53 dataIn = 32'h0000_0005;
54
55 opsel = 2'b01;
56
57 outsel =2'b00;
58
59 asel = 0;
60
61 bsel = 0;
62
63 oen = 1;
64
65
66 #(T*2)
67
68
69 // 2. store AAAA_AAAA  in [1]
70
71 addressA=5'b00000;
72
73 addressB= 5'b00001;
74
75 dataIn = 32'hAAAA_AAAA;
76
77 opsel = 2'b01;
78
79 outsel = 2'b00;
```

```
80
81 asel = 0;
82
83 bsel = 0;
84
85 oen = 1;
86
87 #(T*2)
88
89 // 3. store 5555_5555 in [2]
90
91 addressA = 5'b00000;
92
93 addressB = 5'b00010;
94
95 dataIn = 32'h5555_5555;
96
97
98 opsel = 2'b01;
99
100 outsel =2'b00;
101
102 asel = 0;
103
104 bsel = 0;
105
106 oen = 1;
107
108
109
110 #(T*2)
111
112 // 4. store 0000_000A in [3]
113
114 addressA = 5'b00000;
115
116 addressB = 5'b00011;
117
118 dataIn = 32'h0000_000A;
119
120 opsel = 2'b01;
121
122 outsel =2'b00;
123
124 asel = 0;
125
```

```
126 bsel = 0;
127
128 oen = 1;
129
130 #(T*2)
131
132 // 5. store 0000_0001 in [4]
133
134 addressA = 5'b00000;
135
136 addressB = 5'b00100;
137
138 dataIn = 32'h0000_0001;
139
140 opsel = 2'b01;
141
142 outsel =2'b00;
143
144 asel = 0;
145
146 bsel = 0;
147
148 oen = 1;
149
150 #(T*2)
151
152 // 6. store FFFF_FFFF in [5]
153
154 addressA = 5'b00000;
155
156 addressB = 5'b00101;
157
158 dataIn = 32'hFFFF_FFFF;
159
160 opsel = 2'b01;
161
162 outsel =2'b00;
163
164 asel = 0;
165
166 bsel = 0;
167
168 oen = 1;
169
170 #(T*2)
171
```

```
172 // 7. store 0000_00C8 in [6]
173
174 addressA = 5'b00000;
175
176 addressB = 5'b00110;
177
178 dataIn = 32'h0000_00C8;
179
180 opsel = 2'b01;
181
182 outsel =2'b00;
183
184 asel = 0;
185
186 bsel = 0;
187
188 oen = 1;
189
190 #(T*2)
191
192 // 8. store 0000_012C in [7]
193
194 addressA = 5'b00000;
195
196 addressB = 5'b00111;
197
198 dataIn = 32'h0000_012C;
199
200 opsel = 2'b01;
201
202 outsel =2'b00;
203
204 asel = 0;
205
206 bsel = 0;
207
208 oen = 1;
209
210 #(T*2)
211
212 // 9. store 0000_0001 in [8]
213
214 addressA = 5'b00000;
215
216 addressB = 5'b01000;
217
```

```
218 dataIn = 32'h0000_0001;
219
220 opsel = 2'b01;
221
222 outsel =2'b00;
223
224 asel = 0;
225
226 bsel = 0;
227
228 oen = 1;
229
230 #(T*2)
231
232 // 10. store AAAA_AAAB in [9]
233
234 addressA = 5'b00000;
235
236 addressB = 5'b01001;
237
238 dataIn = 32'hAAAA_AAAB;
239
240 opsel = 2'b01;
241
242 outsel =2'b00;
243
244 asel = 0;
245
246 bsel = 0;
247
248 oen = 1;
249
250 #(T*2)
251
252 // 11. store 5555_5555 in [10]
253
254 addressA = 5'b00000;
255
256 addressB = 5'b01010;
257
258 dataIn = 32'h5555_5555;
259
260 opsel = 2'b01;
261
262 outsel =2'b00;
263
```

```
264 asel = 0;
265
266 bsel = 0;
267
268 oen = 1;
269
270 #(T*2)
271 //12. ADD      [2][0]
272
273 addressA = 5'd2;
274
275 addressB = 5'd0;
276
277 dataIn = 32'd0;
278
279 opsel = 2'b00;
280
281 outsel =2'b01;
282
283 asel = 1;
284
285 bsel = 1;
286
287 oen = 1;
288
289
290 #(T*2)
291
292 #(T*2)
293
294 //13. ADD      [1][2]
295 addressA = 5'd1;
296
297 addressB = 5'd2;
298
299 dataIn = 32'd0;
300
301 opsel = 2'b00;
302
303 outsel =2'b01;
304
305 asel = 1;
306
307 bsel = 1;
308
309 oen = 1;
```

```
310
311 #(T*2)
312
313 //14. ADD       [6][7]
314 addressA = 5'd6;
315
316 addressB = 5'd7;
317
318 dataIn = 32'd0;
319
320 opsel = 2'b00;
321
322 outsel =2'b01;
323
324 asel = 1;
325
326 bsel = 1;
327
328 oen = 1;
329
330 #(T*2)
331
332 //14. ADD       [0][3]
333 addressA = 5'd0;
334
335 addressB = 5'd3;
336
337 dataIn = 32'd0;
338
339 opsel = 2'b00;
340
341 outsel =2'b01;
342
343 asel = 1;
344
345 bsel = 1;
346
347 oen = 1;
348
349 #(T*2)
350
351 //15. SUB       [2][4]
352 addressA = 5'd2;
353
354 addressB = 5'd4;
355
```

```
356 dataIn = 32'd0;
357
358 opsel = 2'b01;
359
360 outsel =2'b01;
361
362 asel = 1;
363
364 bsel = 1;
365
366 oen = 1;
367
368 #(T*2)
369
370 //16. ADD       [2][8]
371 addressA = 5'd2;
372
373 addressB = 5'd8;
374
375 dataIn = 32'd0;
376
377 opsel = 2'b00;
378
379 outsel =2'b01;
380
381 asel = 1;
382
383 bsel = 1;
384
385 oen = 1;
386
387 #(T*2)
388
389 //17. SUB       [2][0]
390 addressA = 5'd2;
391
392 addressB = 5'd0;
393
394 dataIn = 32'd0;
395
396 opsel = 2'b01;
397
398 outsel =2'b01;
399
400 asel = 1;
401
```

```
402 bsel = 1;
403
404 oen = 1;
405
406 #(T*2)
407
408 //18. ADD       [9][10]
409 addressA = 5'd9;
410
411 addressB = 5'd10;
412
413 dataIn = 32'd0;
414
415 opsel = 2'b00;
416
417 outsel =2'b01;
418
419 asel = 1;
420
421 bsel = 1;
422
423 oen = 1;
424
425 #(T*2)
426
427 //19. READ      [7]
428 addressA = 5'd7;
429
430 addressB = 5'd7;
431
432 dataIn = 32'd0;
433
434 opsel = 2'b01;
435
436 outsel =2'b00;
437
438 asel = 1;
439
440 bsel = 0;
441
442 oen = 1;
443
444 #(T*2)
445
446 //20. READ      [3]
447 addressA = 5'd3;
```

```
448
449 addressB = 5'd3;
450
451 dataIn = 32'd0;
452
453 opsel = 2'b01;
454
455 outsel =2'b00;
456
457 asel = 1;
458
459 bsel = 0;
460
461 oen = 1;
462
463 #(T*2)
464
465 //21. READ      [1]
466 addressA = 5'd1;
467
468 addressB = 5'd1;
469
470 dataIn = 32'd0;
471
472 opsel = 2'b01;
473
474 outsel =2'b00;
475
476 asel = 1;
477
478 bsel = 0;
479
480 oen = 1;
481
482 end
483 endmodule
```

3 32-Bit CPU Design with New ALU Architecture

3.1 Code

Below is the code developed to include the 32-bit comparator in the CPU.

```
1935 //one bit comparator
1936 module one_bit_comp(a, b, f1, f0);
```

```
1937 input a, b;
1938 output f1, f0;
1939
1940 //if (a > b) then {f1, f0} = 2'b01
1941 //if (a < b) then {f1, f0} = 2'b00
1942 //if (a == b) then {f1, f0} = 2'b10
1943 reg x,y;
1944 always @(a or b) begin
1945     if (a == b)
1946         begin
1947             x = 1;
1948             y = 0;
1949         end
1950     else if (a > b)
1951         begin
1952             x = 0;
1953             y = 1;
1954         end
1955         else begin
1956                 x = 0;
1957                 y = 0;
1958              end
1959 end
1960 assign f1 = x;
1961 assign f0 = y;
1962
1963 endmodule
1964
1965 //mux to select the f1 f0 outputs
1966 module mux_4to2(hi_f1, hi_f0, lo_f1, lo_f0, f1, f0);
1967
1968 input hi_f1, hi_f0, lo_f1, lo_f0;
1969 output f1, f0;
1970
1971 //use hi_f1 to select the correct outputs
1972 reg x,y;
1973 always @(hi_f1 or hi_f0 or lo_f1 or lo_f0) begin
1974     if (hi_f1 == 0)
1975         begin
1976             x = hi_f1;
1977             y = hi_f0;
1978         end
1979     else
1980         begin
1981             x = lo_f1;
1982             y = lo_f0;
```

```
1983         end
1984 end
1985 assign f1 = x;
1986 assign f0 = y;
1987 endmodule
1988
1989 //32-bit tree comparator
1990 module tree_comp(A, B, f1, f0);
1991 input [31 : 0] A, B;
1992 output f1, f0;
1993
1994 wire [31 : 0] f1_L5, f0_L5;
1995 wire [15 : 0] f1_L4, f0_L4;
1996 wire [7 : 0] f1_L3, f0_L3;
1997 wire [3 : 0] f1_L2, f0_L2;
1998 wire [1 : 0] f1_L1, f0_L1;
1999
2000 //write your code here
2001 //Level 5: 32 one_bit_comp go here
2002     one_bit_comp comp1(A[0], B[0], f1_L5[0],f0_L5[0]);
2003     one_bit_comp comp2(A[1], B[1], f1_L5[1],f0_L5[1]);
2004     one_bit_comp comp3(A[2], B[2], f1_L5[2],f0_L5[2]);
2005     one_bit_comp comp4(A[3], B[3], f1_L5[3],f0_L5[3]);
2006     one_bit_comp comp5(A[4], B[4], f1_L5[4],f0_L5[4]);
2007     one_bit_comp comp6(A[5], B[5], f1_L5[5],f0_L5[5]);
2008     one_bit_comp comp7(A[6], B[6], f1_L5[6],f0_L5[6]);
2009     one_bit_comp comp8(A[7], B[7], f1_L5[7],f0_L5[7]);
2010     one_bit_comp comp9(A[8], B[8], f1_L5[8],f0_L5[8]);
2011     one_bit_comp comp10(A[9], B[9], f1_L5[9],f0_L5[9]);
2012     one_bit_comp comp11(A[10], B[10], f1_L5[10],f0_L5[10]);
2013     one_bit_comp comp12(A[11], B[11], f1_L5[11],f0_L5[11]);
2014     one_bit_comp comp13(A[12], B[12], f1_L5[12],f0_L5[12]);
2015     one_bit_comp comp14(A[13], B[13], f1_L5[13],f0_L5[13]);
2016     one_bit_comp comp15(A[14], B[14], f1_L5[14],f0_L5[14]);
2017     one_bit_comp comp16(A[15], B[15], f1_L5[15],f0_L5[15]);
2018     one_bit_comp comp17(A[16], B[16], f1_L5[16],f0_L5[16]);
2019     one_bit_comp comp18(A[17], B[17], f1_L5[17],f0_L5[17]);
2020     one_bit_comp comp19(A[18], B[18], f1_L5[18],f0_L5[18]);
2021     one_bit_comp comp20(A[19], B[19], f1_L5[19],f0_L5[19]);
2022     one_bit_comp comp21(A[20], B[20], f1_L5[20],f0_L5[20]);
2023     one_bit_comp comp22(A[21], B[21], f1_L5[21],f0_L5[21]);
2024     one_bit_comp comp23(A[22], B[22], f1_L5[22],f0_L5[22]);
2025     one_bit_comp comp24(A[23], B[23], f1_L5[23],f0_L5[23]);
2026     one_bit_comp comp25(A[24], B[24], f1_L5[24],f0_L5[24]);
2027     one_bit_comp comp26(A[25], B[25], f1_L5[25],f0_L5[25]);
2028     one_bit_comp comp27(A[26], B[26], f1_L5[26],f0_L5[26]);
```

```
2029     one_bit_comp comp28(A[27], B[27], f1_L5[27],f0_L5[27]);
2030     one_bit_comp comp29(A[28], B[28], f1_L5[28],f0_L5[28]);
2031     one_bit_comp comp30(A[29], B[29], f1_L5[29],f0_L5[29]);
2032     one_bit_comp comp31(A[30], B[30], f1_L5[30],f0_L5[30]);
2033     one_bit_comp comp32(A[31], B[31], f1_L5[31],f0_L5[31]);
2034 //Level 4: 16 mux_4to2 go here
2035     mux_4to2 mx41(f1_L5[1],f0_L5[1],f1_L5[0], f0_L5[0], f1_L4
        [0], f0_L4[0]);
2036     mux_4to2 mx42(f1_L5[3],f0_L5[3],f1_L5[2], f0_L5[2], f1_L4
        [1], f0_L4[1]);
2037     mux_4to2 mx43(f1_L5[5],f0_L5[5],f1_L5[4], f0_L5[4], f1_L4
        [2], f0_L4[2]);
2038     mux_4to2 mx44(f1_L5[7],f0_L5[7],f1_L5[6], f0_L5[6], f1_L4
        [3], f0_L4[3]);
2039     mux_4to2 mx45(f1_L5[9],f0_L5[9],f1_L5[8], f0_L5[8], f1_L4
        [4], f0_L4[4]);
2040     mux_4to2 mx46(f1_L5[11],f0_L5[11],f1_L5[10], f0_L5[10],
        f1_L4[5], f0_L4[5]);
2041     mux_4to2 mx47(f1_L5[13],f0_L5[13],f1_L5[12], f0_L5[12],
        f1_L4[6], f0_L4[6]);
2042     mux_4to2 mx48(f1_L5[15],f0_L5[15],f1_L5[14], f0_L5[14],
        f1_L4[7], f0_L4[7]);
2043     mux_4to2 mx49(f1_L5[17],f0_L5[17],f1_L5[16], f0_L5[16],
        f1_L4[8], f0_L4[8]);
2044     mux_4to2 mx410(f1_L5[19],f0_L5[19],f1_L5[18], f0_L5[18],
        f1_L4[9], f0_L4[9]);
2045     mux_4to2 mx411(f1_L5[21],f0_L5[21],f1_L5[20], f0_L5[20],
        f1_L4[10], f0_L4[10]);
2046     mux_4to2 mx412(f1_L5[23],f0_L5[23],f1_L5[22], f0_L5[22],
        f1_L4[11], f0_L4[11]);
2047     mux_4to2 mx413(f1_L5[25],f0_L5[25],f1_L5[24], f0_L5[24],
        f1_L4[12], f0_L4[12]);
2048     mux_4to2 mx414(f1_L5[27],f0_L5[27],f1_L5[26], f0_L5[26],
        f1_L4[13], f0_L4[13]);
2049     mux_4to2 mx415(f1_L5[29],f0_L5[29],f1_L5[28], f0_L5[28],
        f1_L4[14], f0_L4[14]);
2050     mux_4to2 mx416(f1_L5[31],f0_L5[31],f1_L5[30], f0_L5[30],
        f1_L4[15], f0_L4[15]);
2051
2052 //Level 3: 8 mux_4to2 go here
2053     mux_4to2 mx31(f1_L4[1],f0_L4[1],f1_L4[0], f0_L4[0], f1_L3
        [0], f0_L3[0]);
2054     mux_4to2 mx32(f1_L4[3],f0_L4[3],f1_L4[2], f0_L4[2], f1_L3
        [1], f0_L3[1]);
2055     mux_4to2 mx33(f1_L4[5],f0_L4[5],f1_L4[4], f0_L4[4], f1_L3
        [2], f0_L3[2]);
```

```
2056      mux_4to2 mx34(f1_L4[7],f0_L4[7],f1_L4[6], f0_L4[6], f1_L3
          [3], f0_L3[3]);
2057      mux_4to2 mx35( f1_L4[9],f0_L4[9],f1_L4[8], f0_L4[8],f1_L3
          [4], f0_L3[4]);
2058      mux_4to2 mx36(f1_L4[11],f0_L4[11],f1_L4[10], f0_L4[10],
          f1_L3[5], f0_L3[5]);
2059      mux_4to2 mx37(f1_L4[13],f0_L4[13],f1_L4[12], f0_L4[12],
          f1_L3[6], f0_L3[6]);
2060      mux_4to2 mx38(f1_L4[15],f0_L4[15],f1_L4[14], f0_L4[14],
          f1_L3[7], f0_L3[7]);
2061
2062 //Level 2: 4 mux_4to2 go here
2063      mux_4to2 mx21(f1_L3[1],f0_L3[1],f1_L3[0], f0_L3[0],f1_L2[0],
           f0_L2[0]);
2064      mux_4to2 mx22(f1_L3[3],f0_L3[3],f1_L3[2], f0_L3[2], f1_L2
          [1], f0_L2[1]);
2065      mux_4to2 mx23(f1_L3[5],f0_L3[5],f1_L3[4], f0_L3[4], f1_L2
          [2], f0_L2[2]);
2066      mux_4to2 mx24(f1_L3[7],f0_L3[7],f1_L3[6], f0_L3[6], f1_L2
          [3], f0_L2[3]);
2067
2068 //Level 1: 2 mux_4to2 go here
2069      mux_4to2 mx11(f1_L2[1],f0_L2[1],f1_L2[0], f0_L2[0], f1_L1
          [0], f0_L1[0]);
2070      mux_4to2 mx12(f1_L2[3],f0_L2[3],f1_L2[2], f0_L2[2], f1_L1
          [1], f0_L1[1]);
2071
2072 //Level 0: 1 mux_4to2 goes here
2073      mux_4to2 mx01(f1_L1[1],f0_L1[1],f1_L1[0], f0_L1[0], f1, f0);
2074
2075 endmodule
```

The file `tb_test_comp.v` is not included in this report due to the length. It is included separately with the report under the directory code.

3.2 RTL Test Bench

Figures 48 - 51 show the RTL simulation simvision results of `tb_test_comp.v`, which is included in under the code directory included with this report. The results are also included in the file `stim_proj.out` under the directory case2.

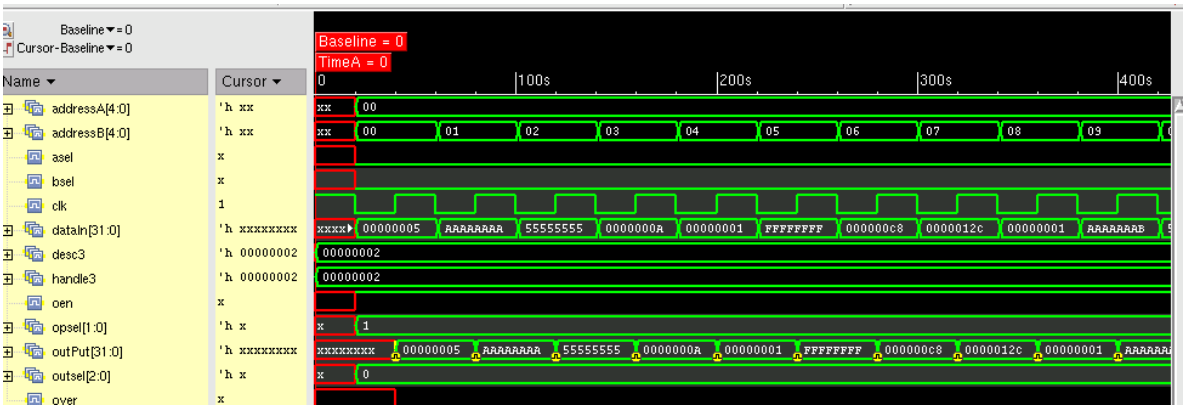

Figure 48: Test Bench 1

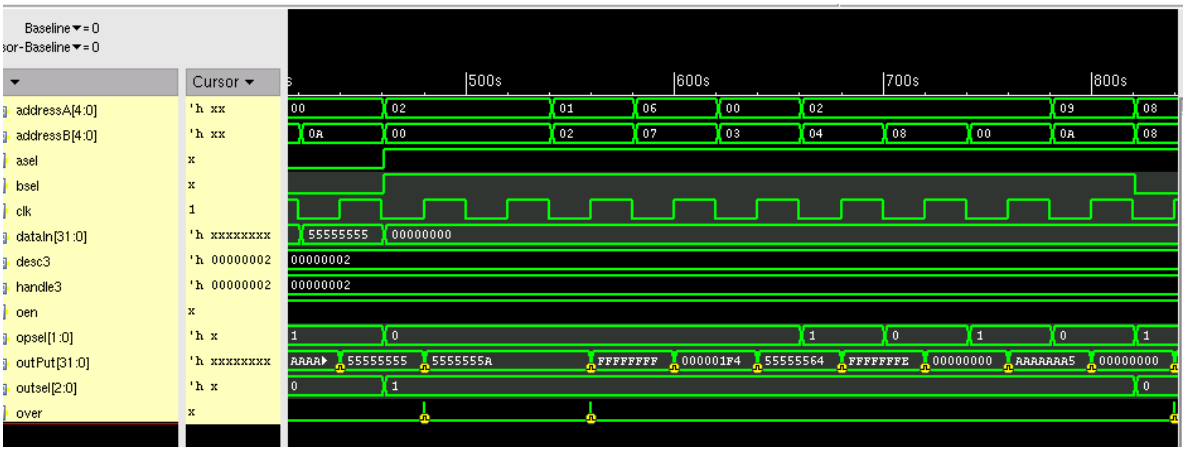

Figure 49: Test Bench 2

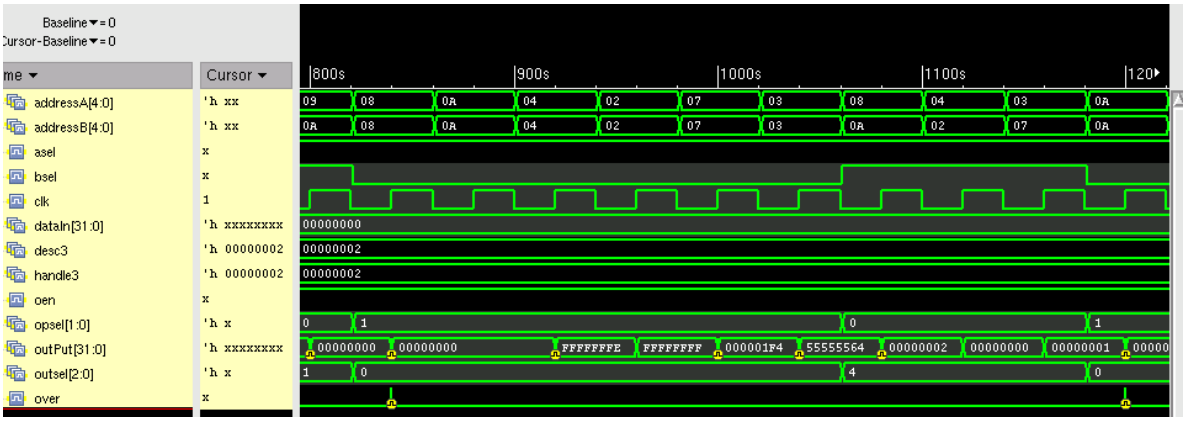

Figure 50: Test Bench 3

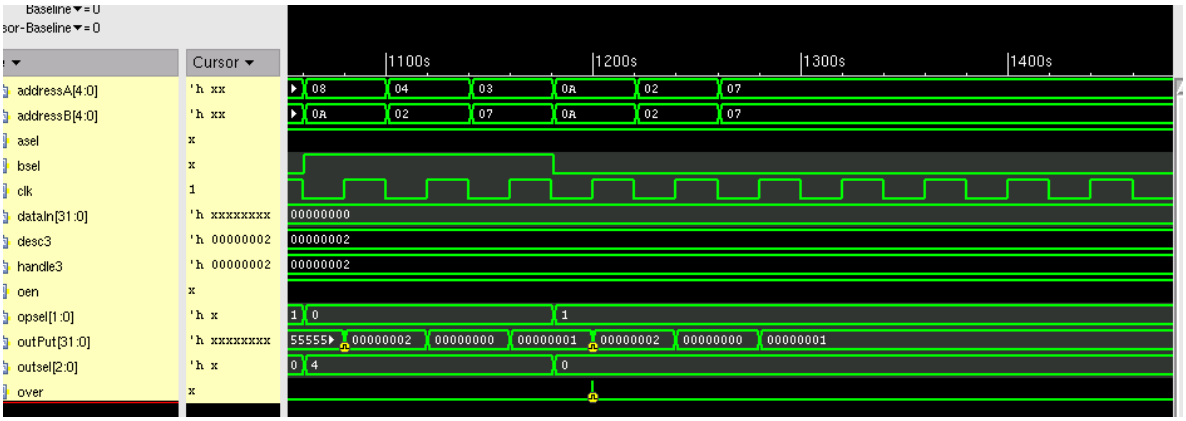

Figure 51: Test Bench 4

3.2.1 Logic Synthesis and Post-Synthesis Simulation

The results of the post synthesis simulation simvision results are shown in Figures 52 - 55. The area of this design is 49305.595675nm and the slack is 26.48. The results of `timing.rep` and `timing.cell` are included with this report under the directory case2.

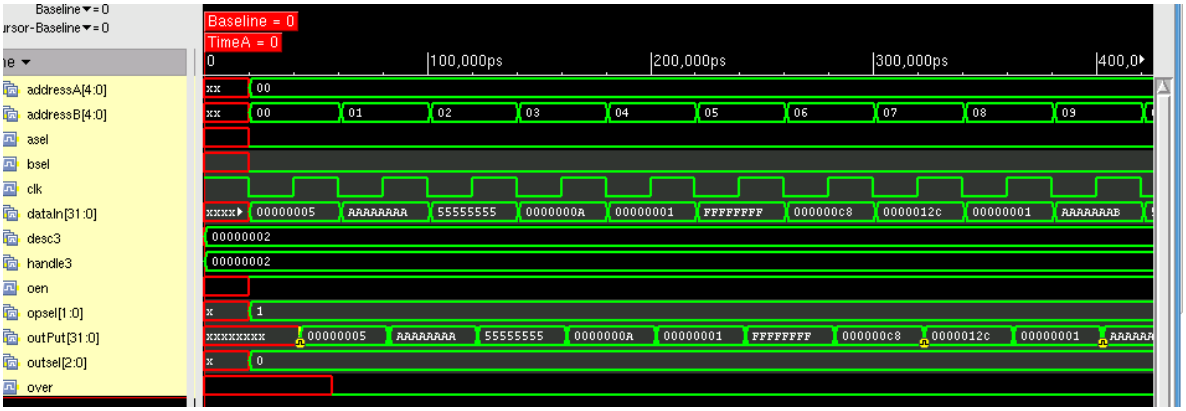

Figure 52: Synthesis Simulation 1

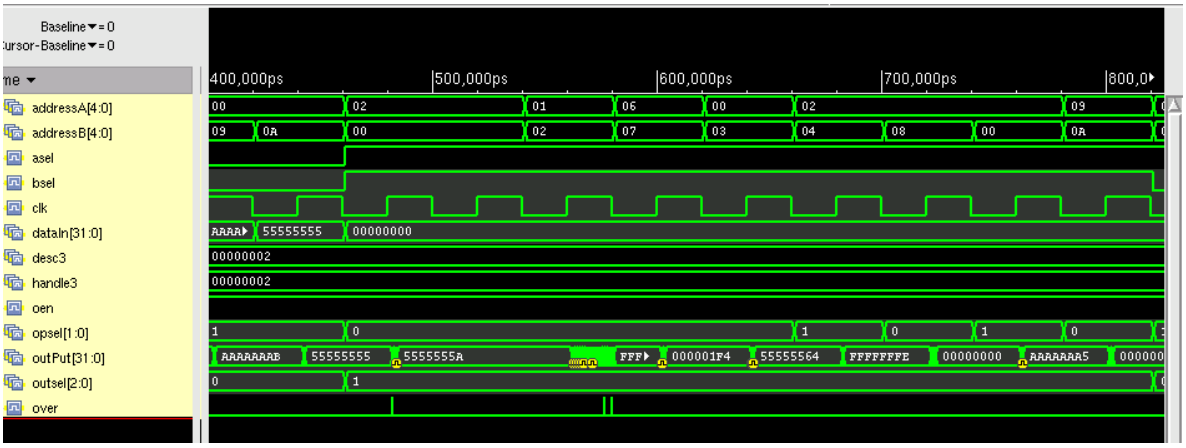

Figure 53: Synthesis Simulation 2

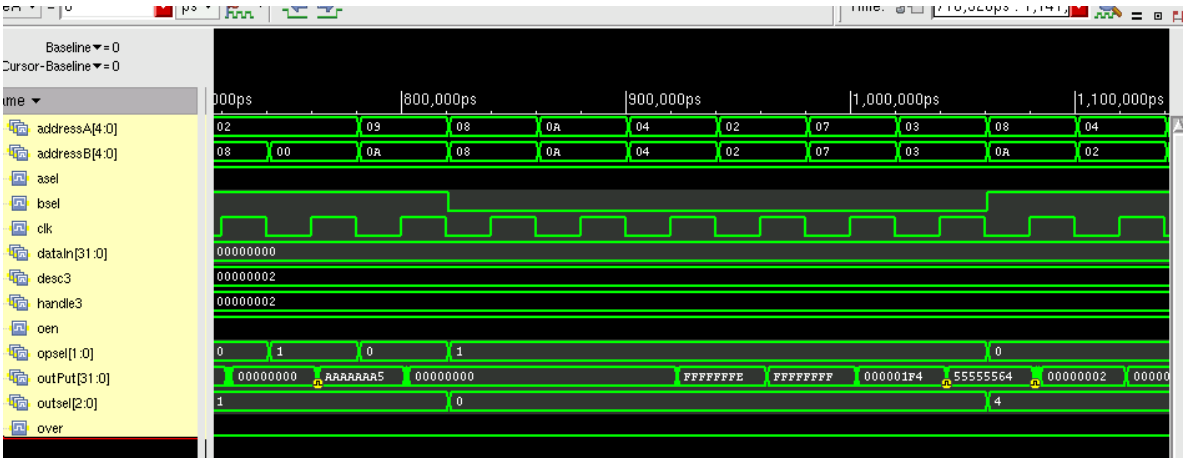

Figure 54: Synthesis Simulation 3

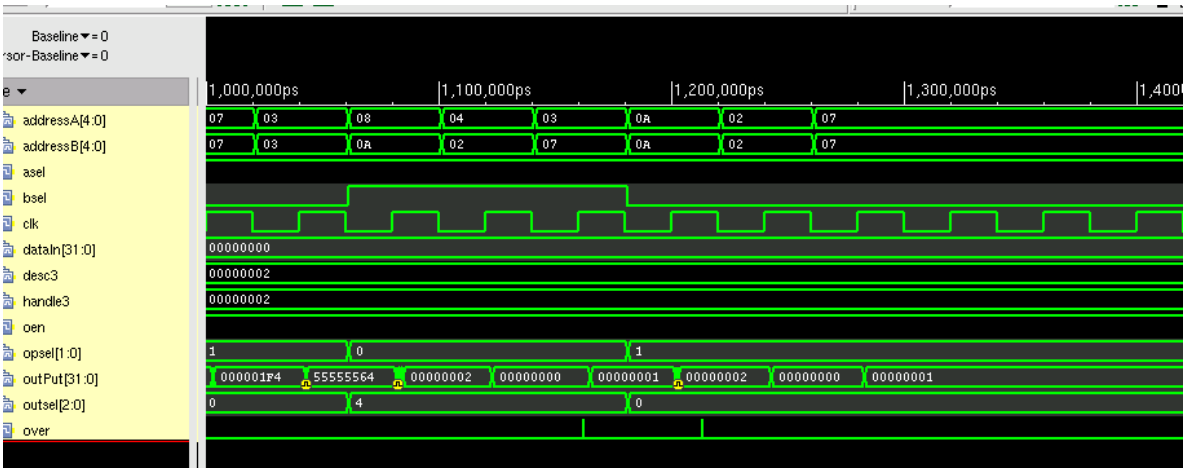

Figure 55: Synthesis Simulation 4

3.2.2 Place & Route and Post-P&R Simulation

The results of the post-P&R simulation simvision results are shown in Figures 56 - 60. The results of `timing.rep.5.final` show a slack of 19.848. This file is included under the directory case2. The maximum clock frequency was found to be ≈ 71 MHz.

```
Finished Calculating power
2015-Dec-01 21:24:09 (2015-Dec-02 03:24:09 GMT)
*



Total Power
--------------------------------------------------------------------------------
Total Internal Power:       2.528      54.74%
Total Switching Power:      1.824      39.48%
Total Leakage Power:      0.2671      5.783%
Total Power:       4.619
--------------------------------------------------------------------------------
report_power consumed time (real time) 00:00:03 : peak memory (593M)
Output file is power.final
****************************************
* Encounter script finished           *
*                                     *
* Results:                            *
* --------                            *
* Layout:  final.gds2                 *
* Netlist: final.v                    *
* Timing:  timing.rep.5.final         *
* Area:    area.final                 *
* Power:   power.final                *
*                                     *
* Type 'win' to get the Main Window   *
* or type 'exit' to quit              *
*                                     *
****************************************
```

Figure 56: Power Consumption

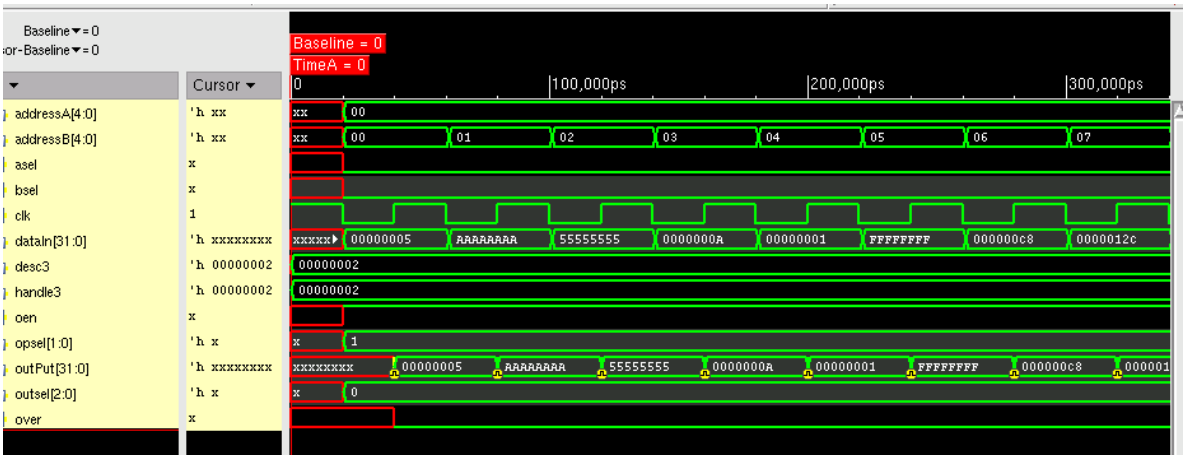

Figure 57: Layout Simulation 1

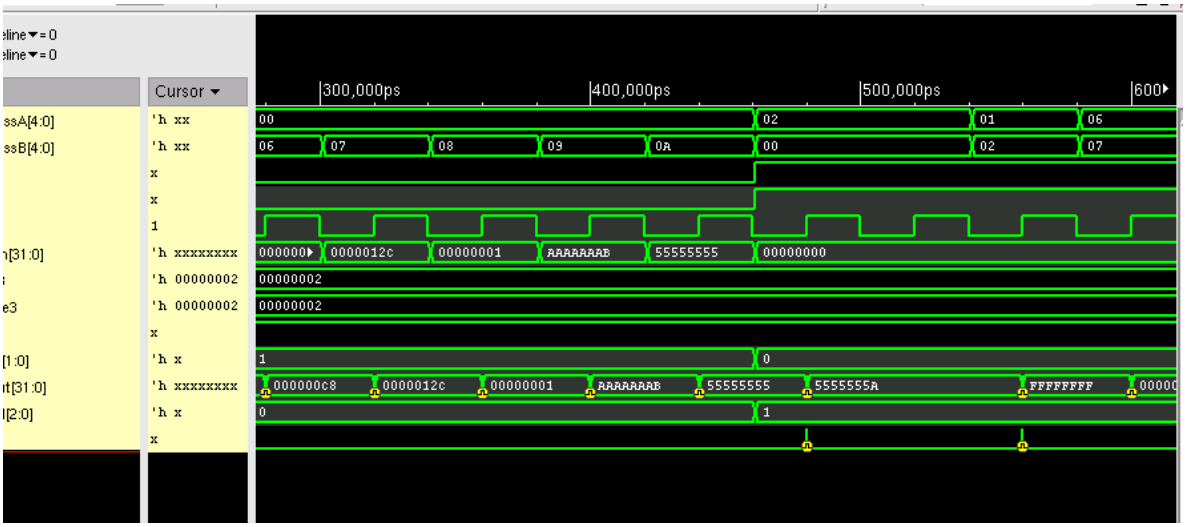

Figure 58: Layout Simulation 2

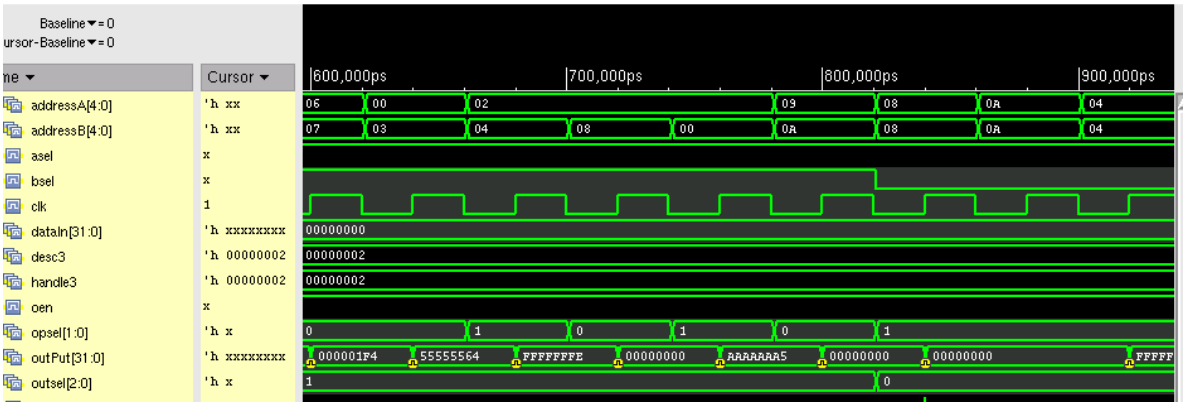

Figure 59: Layout Simulation 3

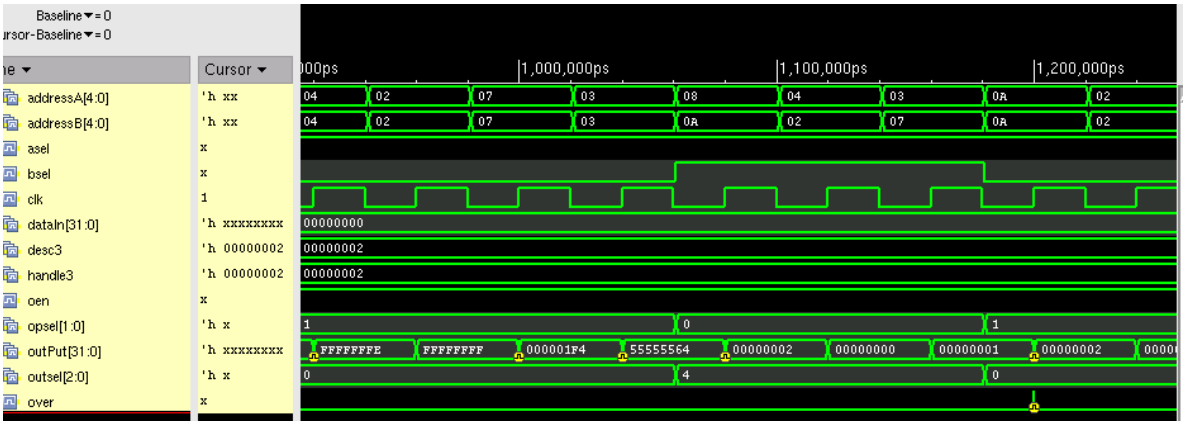

Figure 60: Layout Simulation 4

4 Conclusions

Overall a 32-bit pipelined CPU was successfully studied. Through the simulation and analysis of the code, it is clear how a CPU is both physically and logically constructed. Different adder types were adequately analyzed and it is apparent that each configuration affects the power and delay of the CPU as a whole. Furthermore, a comparator was successfully added to the ALU of the CPU. The goals of this project were all achieved successfully.