# CSCE 611
# Testbenches, Simulation, and JTAG and the MIPS ALU

Instructor: Jason D. Bakos

UNIVERSITY OF SOUTH CAROLINA

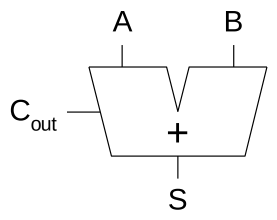
## Tutorial Outline

- Tutorial 1: Simple adder
  - Design register and full adder using Verilog
  - Connect full adders to create 9-bit adder using Quartus schematic tool
  - Create test bench to verify functionality
  - Synthesize and implement on DE2 board
  - Convert to carry lookahead adder to improve clock speed
  - Convert to native adder to compare clock speed

- Tutorial 2: MIPS ALU
  - Design ALU that we'll use in the CPU designer later
  - Implement all MIPS arithmetic, logic, shift, and comparison instructions

# Review: 1-Bit Adders

## Half Adder

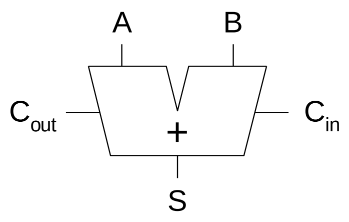


| A | B | $C_{out}$ | S |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

$S = A \oplus B$

$C_{out} = AB$

## Full Adder



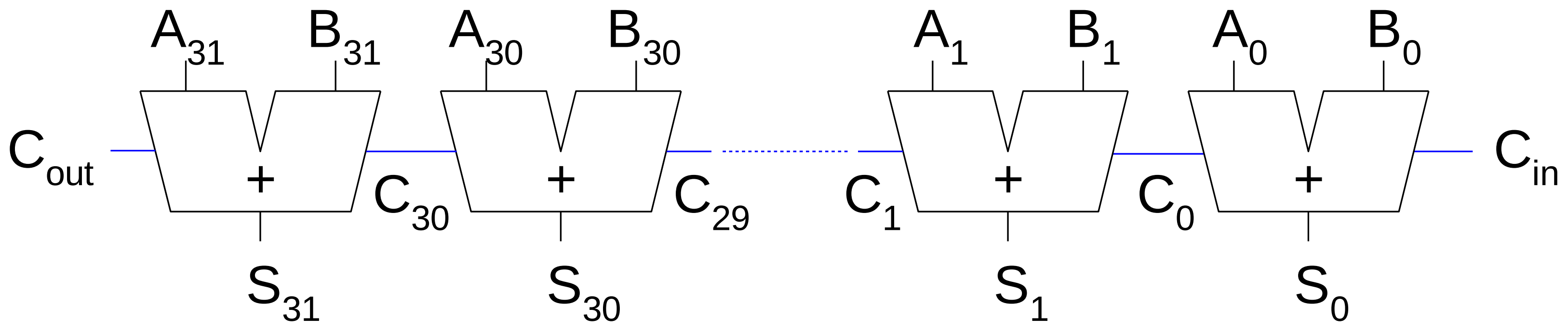
| $C_{in}$ | A | B | $C_{out}$ | S |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

$S = A \oplus B \oplus C_{in}$

$C_{out} = AB + AC_{in} + BC_{in}$

## Review: Ripple-Carry Adder

- Chain 1-bit adders together
- Carry ripples through entire chain
- Disadvantage: slow



## Full Adder

- File | New | Verilog HDL File

```
module full_adder (input a,b,cin,output s,cout);
  assign s = a ^ b ^ cin;
  assign cout = (a & b) | (a & cin) | (b & cin);
endmodule
```

Save as “fulladder.v”

## Bottom-up Design

- Combine 9 full adders to create 9 bit adder
- Create new Verilog file:

```
module adder (input [8:0] a,b,output [8:0] s,output cout);
  wire [9:0] carry_chain;

  full_adder add0(a[0],b[0],1'b0,s[0],carry_chain[0]);
  full_adder add1(a[1],b[1],carry_chain[0],s[1],carry_chain[1]);
  full_adder add2(a[2],b[2],carry_chain[1],s[2],carry_chain[2]);
  full_adder add3(a[3],b[3],carry_chain[2],s[3],carry_chain[3]);
  full_adder add4(a[4],b[4],carry_chain[3],s[4],carry_chain[4]);
  full_adder add5(a[5],b[5],carry_chain[4],s[5],carry_chain[5]);
  full_adder add6(a[6],b[6],carry_chain[5],s[6],carry_chain[6]);
  full_adder add7(a[7],b[7],carry_chain[6],s[7],carry_chain[7]);
  full_adder add8(a[8],b[8],carry_chain[7],s[8],carry_chain[8]);

  assign cout = carry_chain[8];

endmodule
```

- Save as “adder.v”

## Bottom-up Design

- Copying and pasting each instantiation is bulky and isn’t amenable to variable adder sizes
- Use generate statement:

```
module adder (input [8:0] a,b,output [8:0] s,output cout);
  wire [9:0] carry_chain;
  genvar i;

  generate for (i=0;i<9;i=i+1) begin : adder_array
  if (i==0)
    full_adder add(a[i],b[i],1'b0,s[i],carry_chain[i]);
  else
    full_adder add(a[i],b[i],carry_chain[i-1],s[i],carry_chain[i]);
  end
  endgenerate

  assign cout = carry_chain[8];

endmodule
```

## Parameters

- Now let's add a parameter:

```
module adder #(parameter width=32) (input [width-1:0] a,b,output [width-
1:0] s,output cout);
  wire [width-1:0] carry_chain;
  genvar i;
  generate for (i=0;i<width;i=i+1) begin : adder_array
  if (i==0)
    full_adder add(a[i],b[i],1'b0,s[i],carry_chain[i]);
  else
    full_adder add(a[i],b[i],carry_chain[i-1],s[i],carry_chain[i]);
  end
  endgenerate
    assign cout = carry_chain[width-1];
  endmodule
```

## Register

- Let’s create a register
- New Verilog file:

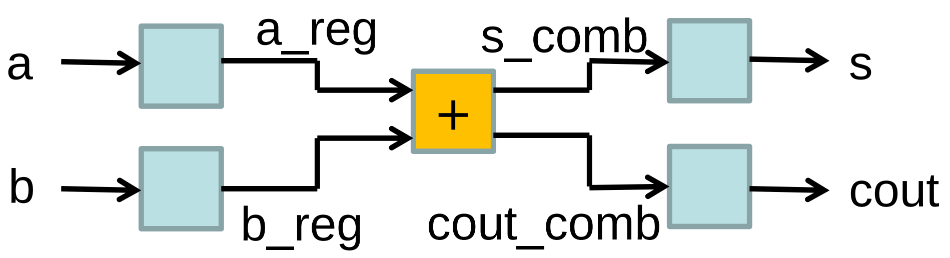
```
module regn #(parameter width=32) (input clk,input [width-1:0]
d,output reg [width-1:0] q);

always @(posedge clk)
begin
  q <= d;
end

endmodule
```

- Save as regn.v

# Adder Datapath

- Create an adder datapath, where registers supply input data and registers capture output data:

```
module adder_reg #(parameter width=32) (input clk,input [width-1:0] a,b,output[width-
1:0] s,output cout);

wire [width-1:0] a_reg,b_reg,s_comb;
wire cout_comb;

regn #(width) input_reg_a(clk,a,a_reg);
regn #(width) input_reg_b(clk,b,b_reg);
adder#(width) my_adder(a_reg,b_reg,s_comb,cout_comb);
regn #(width) output_reg_s(clk,s_comb,s);
regn #(1) output_reg_cout(clk,cout_comb,cout);

endmodule
```



## Testbenches

- HDL code written to test another HDL module, the *device under test* (dut), also called the *unit under test* (uut)

- Generally not synthesize(d|able)

- Types of testbenches:
  - Simple testbench
  - Self-checking testbench
  - Self-checking testbench with testvectors

## Example

```
module sillyfunction(input  a, b, c,
                     output y);
  assign y = ~b & ~c | a & ~b;
endmodule
```

## Simple Testbench

```
module testbench1();
  reg  a, b, c;
  wire y;
  // instantiate device under test
  sillyfunction dut(a, b, c, y);
  // apply inputs one at a time
  initial begin
    a = 1'b0; b = 1'b0; c = 1'b0; #10;
    c = 1'b1; #10;
    b = 1'b1; c = 1'b0; #10;
    c = 1'b1; #10;
    a = 1'b1; b = 1'b0; c = 1'b0; #10;
    c = 1'b1; #10;
    b = 1'b1; c = 1'b0; #10;
    c = 1'b1; #10;
  end
endmodule
```

# Self-checking Testbench

```
module testbench2();
  reg  a, b, c;
  wire y;
  // instantiate device under test
  sillyfunction dut(a, b, c, y);
  // apply inputs one at a time
  // checking results
  initial begin
    a = 1'b0; b = 1'b0; c = 1'b0; #10;
    if (y !== 1'b1) $display("000 failed.");
    c = 1'b1; #10;
    if (y !== 1'b0) $display("001 failed.");
    b = 1'b1; c = 1'b0; #10;
    if (y !== 1'b0) $display("010 failed.");
    c = 1'b1; #10;
    if (y !== 0) $display("011 failed.");
    a = 1'b1; b = 1'b0; c = 1'b0; #10;
    if (y !== 1'b1) $display("100 failed.");
    c = 1'b1; #10;
    if (y !== 1) $display("101 failed.");
    b = 1'b1; c = 1'b0; #10;
    if (y !== 1'b0) $display("110 failed.");
    c = 1'b1; #10;
    if (y !== 1'b0) $display("111 failed.");
  end
endmodule
```

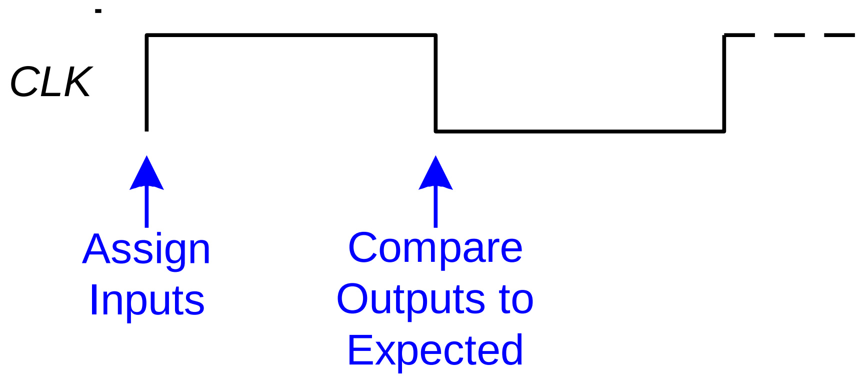
# Testbench with Testvectors

- Write testvector file: inputs and expected outputs

- Testbench:
  1. Generate clock for assigning inputs, reading outputs
  2. Read testvectors file into array
  3. Assign inputs, expected outputs
  4. Compare outputs to expected outputs and report errors

## Testbench with Testvectors

- Testbench clock is used to assign inputs (on the rising edge) and compare outputs with expected outputs (on the falling edge).



- The testbench clock may also be used as the clock source for synchronous sequential circuits

## Testvectors File

File: `example.tv - contains vectors of abc_yexpected`

```
000_1
001_0
010_0
011_0
100_1
101_1
110_0
111_0
```

## Testbench: 1. Generate Clock

```
module testbench3();
  reg         clk, reset;
  reg         a, b, c, yexpected;
  wire        y;
  reg  [31:0] vectornum, errors;    // bookkeeping variables
  reg  [3:0]  testvectors[10000:0]; // array of testvectors

  // instantiate device under test
  sillyfunction dut(a, b, c, y);

  // generate clock
  always     // no sensitivity list, so it always executes
    begin
      clk = 1'b1; #5; clk = 1'b0; #5;
    end
```

## 2. Read Testvectors into Array

```
  // at start of test, load vectors
  // and pulse reset

  initial
     begin
       $readmemb("example.tv", testvectors);
       vectornum = 0; errors = 0;
       reset = 1'b1; #27; reset = 1'b0;
     end

 // Note: $readmemh reads testvector files written in
 // hexadecimal (as opposed to $readmemb in binary)
 // Can insert Verilog comments into file
 // Can also add addresses (use @ with hex address)
```

## 3. Assign Inputs and Expected Outputs

```
// apply test vectors on rising edge of clk
 always @(posedge clk)
   begin
     #1; {a, b, c, yexpected} = testvectors[vectornum];
   end
```

## 4. Compare Outputs with Expected Outputs

```
// check results on falling edge of clk
   always @(negedge clk)
    if (~reset) begin // skip during reset
      if (y !== yexpected) begin
        $display("Error: inputs = %b", {a, b, c});
        $display("  outputs = %b (%b expected)",y,yexpected);
        errors = errors + 1;
      end

// Note: to print in hexadecimal, use %h. For example,
//       $display("Error: inputs = %h", {a, b, c});
```

## 4. Compare Outputs with Expected Outputs

```
// increment array index and read next testvector
      vectornum = vectornum + 1;
      if (testvectors[vectornum] === 4'bx) begin
          $display("%d tests completed with %d errors",
                vectornum, errors);
        $finish;
      end
    end
endmodule


// Note: === and !== can compare values that are
// x or z.
```

## Another Testbench

```
module testbench();

reg clk, reset;
reg [31:0] vectornum, errors; // bookkeeping variables
reg [23:0] testvectors[10000:0]; // array of testvectors

reg [7:0]
a,b,sexpected,sexpected_dly1,sexpected_dly2,sexpected_dly3,a_dly1,a
_dly2,b_dly1,b_dly2;
wire [7:0] s;
wire cout;

adder_reg #(8) myadder (clk,a,b,s,cout);

always begin
                clk=1;#5;clk=0;#5;
end

initial begin
                $readmemh("addtests.tv", testvectors);
                vectornum = 0; errors = 0;
                reset = 1; #4; reset = 0;
end

always @(posedge clk) begin
                {a, b, sexpected} <= testvectors[vectornum];
                sexpected_dly1 <= sexpected;
                sexpected_dly2 <= sexpected_dly1;
                a_dly1 <= a;
                a_dly2 <= a_dly1;
                b_dly1 <= b;
                b_dly2 <= b_dly1;
end
```

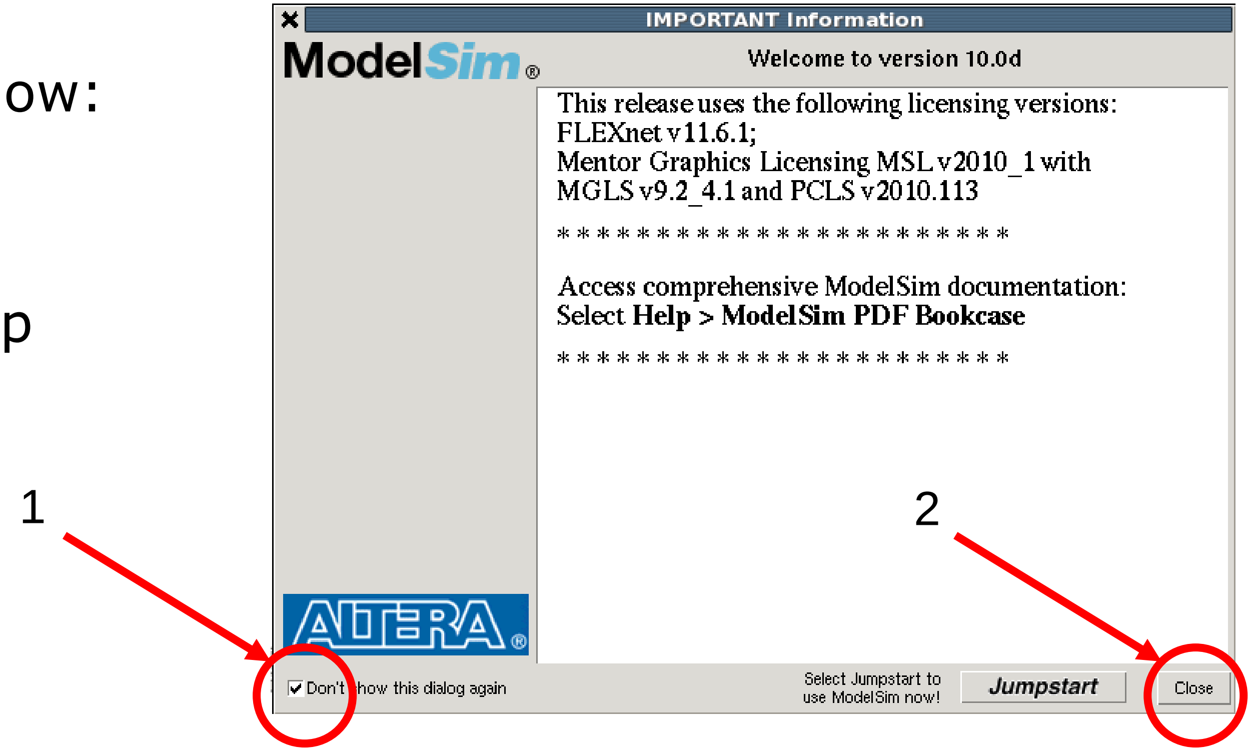
```
// check results on falling edge of clk
always @(negedge clk) begin
                if (s !== sexpected_dly2) begin
                                $display("Error: a = %d, b =
%d, s = %d, t = %d", a_dly2,b_dly2,s,$time);
                                errors = errors + 1;
                end
                vectornum = vectornum + 1;
                if (testvectors[vectornum] === 4'bx) begin
                                $display("%d tests completed
with %d errors",vectornum, errors);
                                $finish;
                end
end

endmodule
```

# Simulation

- In order to simulate we need to start Modelsim separately
- In terminal window:

```
cd quartus_work
vsim&
```

- Close any startup windows


1
2
IMPORTANT Information
ModelSim
ALTERA
Don't show this dialog again
Select Jumpstart to use ModelSim now!
Jumpstart
Close

Welcome to version 10.0d

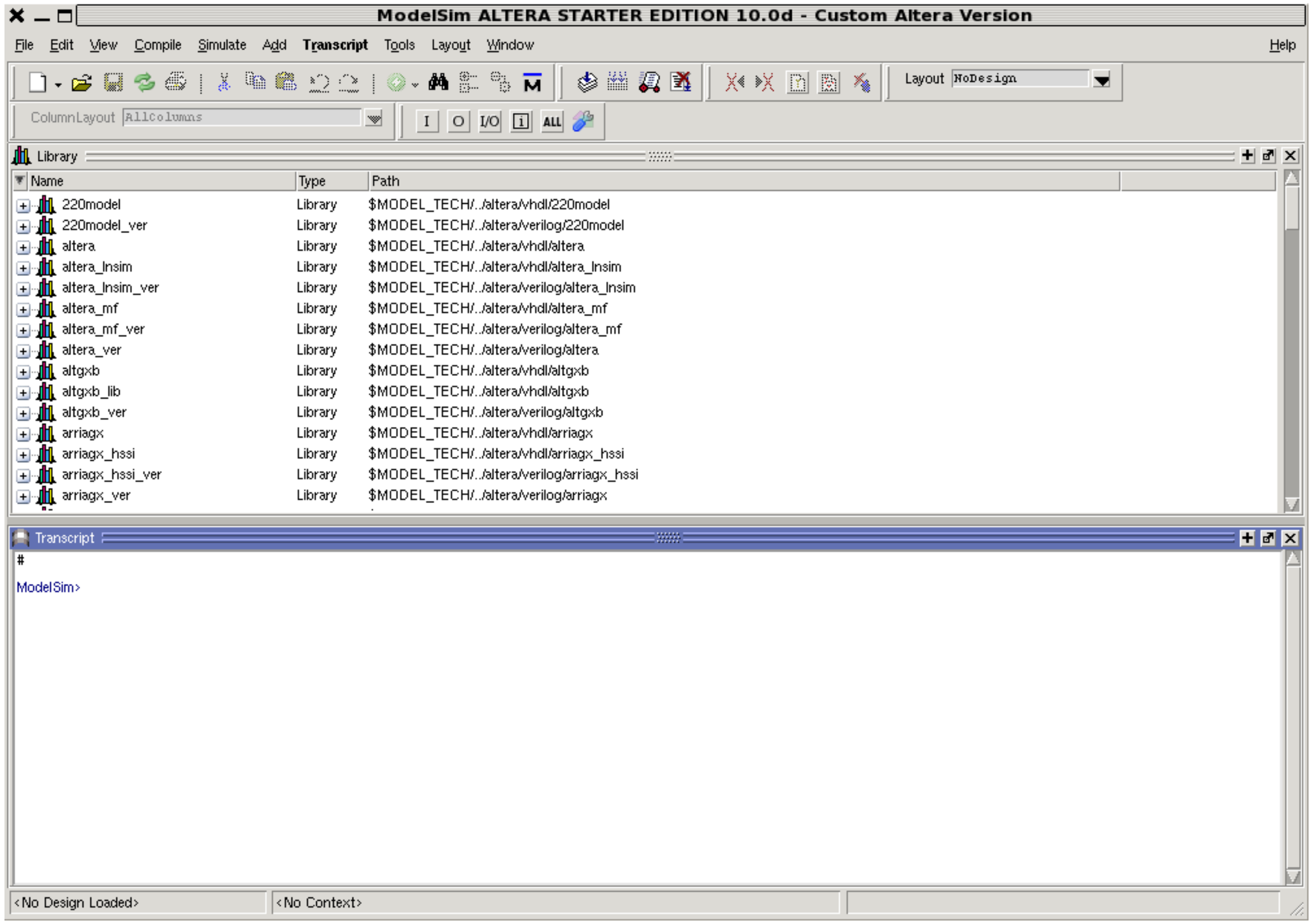
This release uses the following licensing versions:
FLEXnet v11.6.1;
Mentor Graphics Licensing MSL v2010_1 with
MGLS v9.2_4.1 and PCLS v2010.113

* * * * * * * * * * * * * * * * * * * * * * *

Access comprehensive ModelSim documentation:
Select **Help > ModelSim PDF Bookcase**

* * * * * * * * * * * * * * * * * * * * * * *

# Modelsim


ModelSim ALTERA STARTER EDITION 10.0d - Custom Altera Version
File Edit View Compile Simulate Add Transcript Tools Layout Window Help
Layout NoDesign
ColumnLayout AllColumns
Library
Transcript

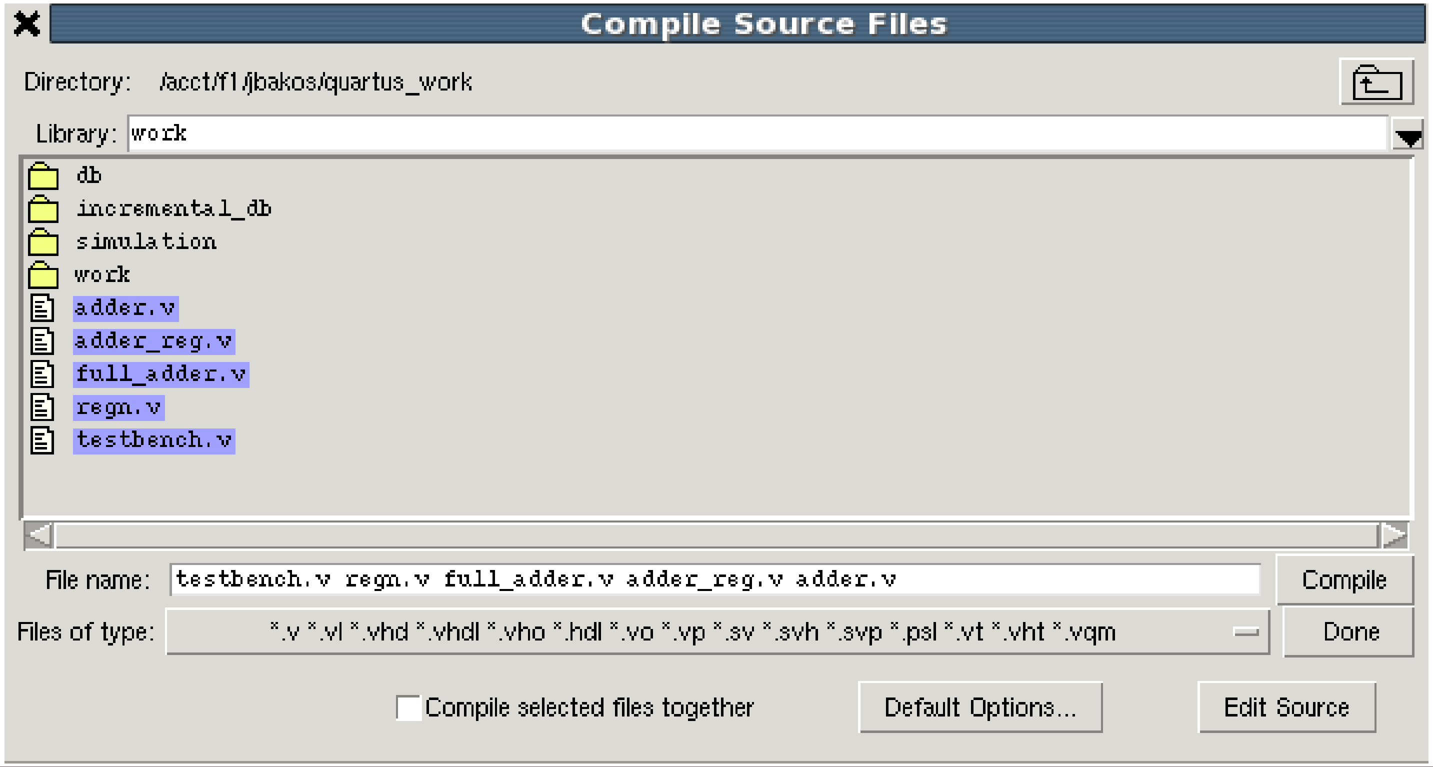
| Name | Type | Path |
|---|---|---|
| 220model | Library | $MODEL_TECH/../altera/vhdl/220model |
| 220model_ver | Library | $MODEL_TECH/../altera/verilog/220model |
| altera | Library | $MODEL_TECH/../altera/vhdl/altera |
| altera_lnsim | Library | $MODEL_TECH/../altera/vhdl/altera_lnsim |
| altera_lnsim_ver | Library | $MODEL_TECH/../altera/verilog/altera_lnsim |
| altera_mf | Library | $MODEL_TECH/../altera/vhdl/altera_mf |
| altera_mf_ver | Library | $MODEL_TECH/../altera/verilog/altera_mf |
| altera_ver | Library | $MODEL_TECH/../altera/verilog/altera |
| altgxb | Library | $MODEL_TECH/../altera/vhdl/altgxb |
| altgxb_lib | Library | $MODEL_TECH/../altera/vhdl/altgxb |
| altgxb_ver | Library | $MODEL_TECH/../altera/verilog/altgxb |
| arriagx | Library | $MODEL_TECH/../altera/vhdl/arriagx |
| arriagx_hssi | Library | $MODEL_TECH/../altera/vhdl/arriagx_hssi |
| arriagx_hssi_ver | Library | $MODEL_TECH/../altera/verilog/arriagx_hssi |
| arriagx_ver | Library | $MODEL_TECH/../altera/verilog/arriagx |

```
#
ModelSim>
```

<No Design Loaded> <No Context>

## Modelsim

- Select Compile | Compile…
- Select all five Verilog files for our design and click Compile then Done


Compile Source Files

Directory: /acct/f1/jbakos/quartus_work

Library: work

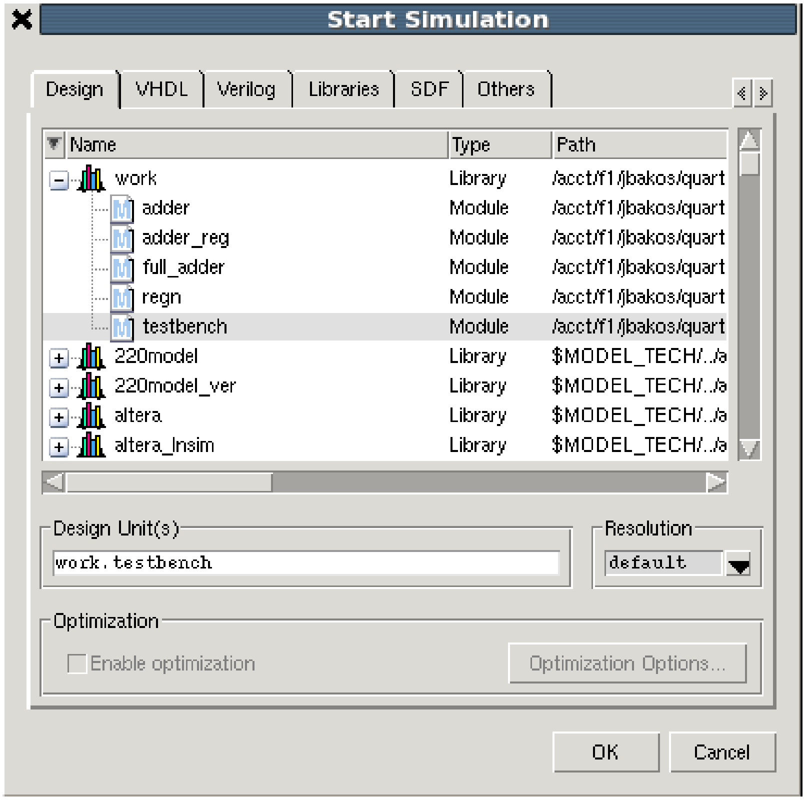
```
db
incremental_db
simulation
work
adder.v
adder_reg.v
full_adder.v
regn.v
testbench.v
```

File name: testbench.v regn.v full_adder.v adder_reg.v adder.v

Files of type: *.v *.vl *.vhd *.vhdl *.vho *.hdl *.vo *.vp *.sv *.svh *.svp *.psl *.vt *.vht *.vqm

# Modelsim

- Now select Simulate | Start Simulation
- Under the work library select testbench and click Ok



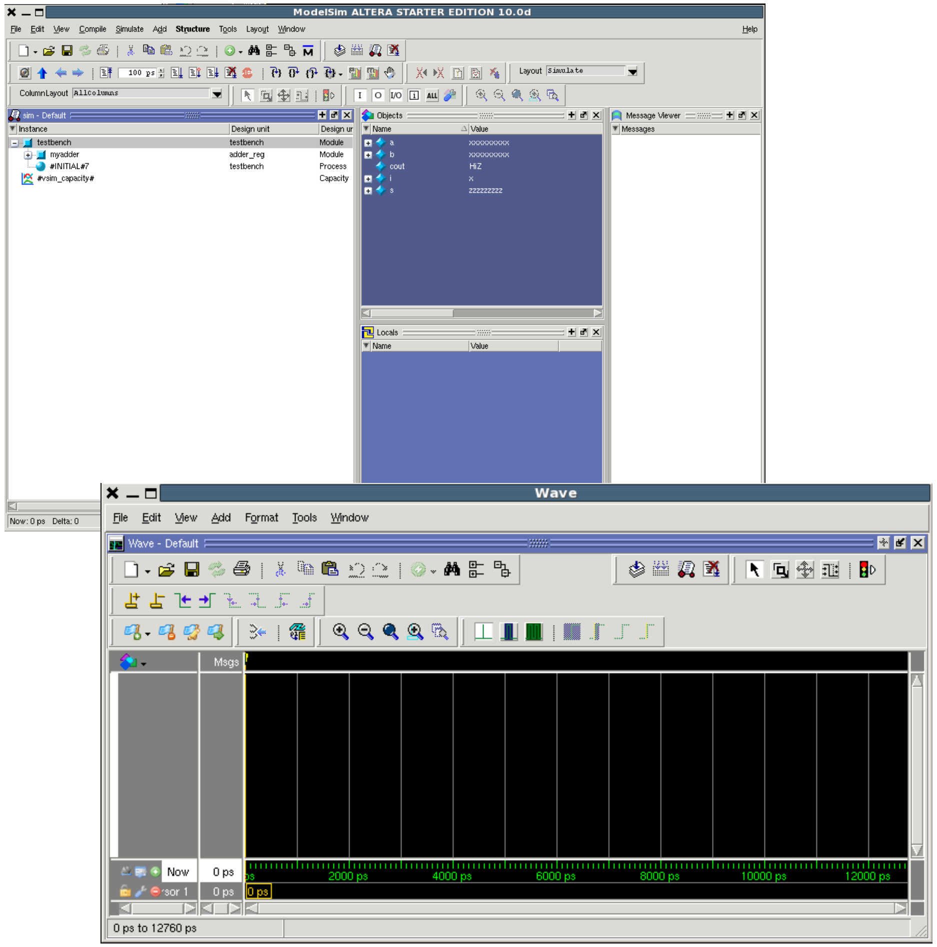
| Name | Type | Path |
|---|---|---|
| work | Library | /acct/f1/jbakos/quart |
| adder | Module | /acct/f1/jbakos/quart |
| adder_reg | Module | /acct/f1/jbakos/quart |
| full_adder | Module | /acct/f1/jbakos/quart |
| regn | Module | /acct/f1/jbakos/quart |
| testbench | Module | /acct/f1/jbakos/quart |
| 220model | Library | $MODEL_TECH/../a |
| 220model_ver | Library | $MODEL_TECH/../a |
| altera | Library | $MODEL_TECH/../a |
| altera_lnsim | Library | $MODEL_TECH/../a |

Design Unit(s): work.testbench

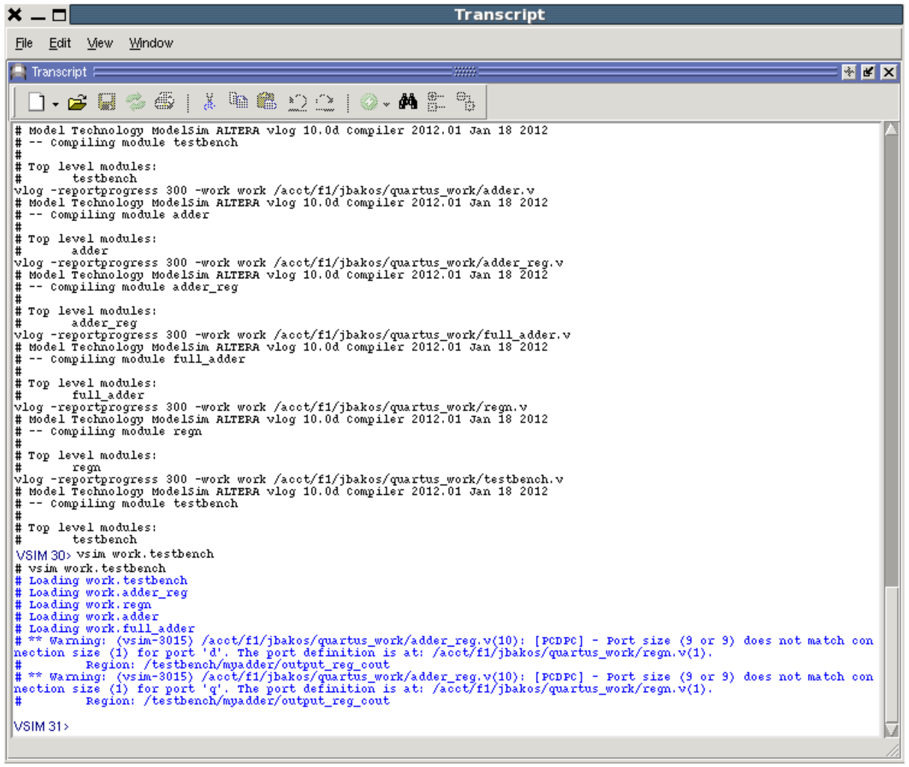
Resolution: default

Optimization: Enable optimization

# Modelsim Windows

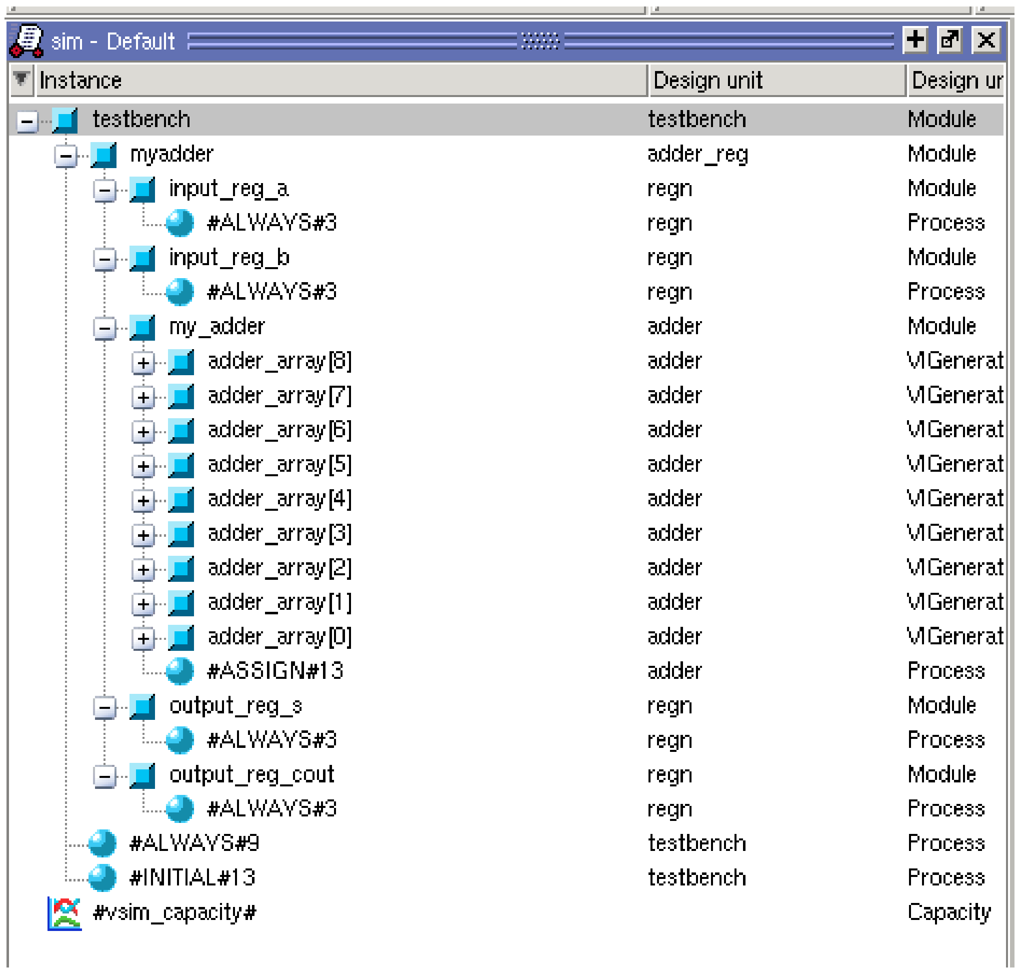


ModelSim ALTERA STARTER EDITION 10.0d
File Edit View Compile Simulate Add Structure Tools Layout Window Help
sim - Default
Instance Design unit Design unit type
testbench testbench Module
myadder adder_reg Module
#INITIAL#7 testbench Process
#vsim_capacity# Capacity
Objects
Name Value
a xxxxxxxxx
b xxxxxxxxx
cout HiZ
i x
s zzzzzzzzz
Locals
Message Viewer
Messages
Now: 0 ps Delta: 0
Wave
File Edit View Add Format Tools Window
Wave - Default
Msgs
Now 0 ps
Cursor 1 0 ps
2000 ps 4000 ps 6000 ps 8000 ps 10000 ps 12000 ps
0 ps to 12760 ps



```
# Model Technology ModelSim ALTERA vlog 10.0d Compiler 2012.01 Jan 18 2012
# -- Compiling module testbench
#
# Top level modules:
#       testbench
vlog -reportprogress 300 -work work /acct/f1/jbakos/quartus_work/adder.v
# Model Technology ModelSim ALTERA vlog 10.0d Compiler 2012.01 Jan 18 2012
# -- Compiling module adder
#
# Top level modules:
#       adder
vlog -reportprogress 300 -work work /acct/f1/jbakos/quartus_work/adder_reg.v
# Model Technology ModelSim ALTERA vlog 10.0d Compiler 2012.01 Jan 18 2012
# -- Compiling module adder_reg
#
# Top level modules:
#       adder_reg
vlog -reportprogress 300 -work work /acct/f1/jbakos/quartus_work/full_adder.v
# Model Technology ModelSim ALTERA vlog 10.0d Compiler 2012.01 Jan 18 2012
# -- Compiling module full_adder
#
# Top level modules:
#       full_adder
vlog -reportprogress 300 -work work /acct/f1/jbakos/quartus_work/regn.v
# Model Technology ModelSim ALTERA vlog 10.0d Compiler 2012.01 Jan 18 2012
# -- Compiling module regn
#
# Top level modules:
#       regn
vlog -reportprogress 300 -work work /acct/f1/jbakos/quartus_work/testbench.v
# Model Technology ModelSim ALTERA vlog 10.0d Compiler 2012.01 Jan 18 2012
# -- Compiling module testbench
#
# Top level modules:
#       testbench
VSIM 30> vsim work.testbench
# vsim work.testbench
# Loading work.testbench
# Loading work.adder_reg
# Loading work.regn
# Loading work.adder
# Loading work.full_adder
# ** Warning: (vsim-3015) /acct/f1/jbakos/quartus_work/adder_reg.v(10): [PCDPC] - Port size (9 or 9) does not match con
nection size (1) for port 'd'. The port definition is at: /acct/f1/jbakos/quartus_work/regn.v(1).
#         Region: /testbench/myadder/output_reg_cout
# ** Warning: (vsim-3015) /acct/f1/jbakos/quartus_work/adder_reg.v(10): [PCDPC] - Port size (9 or 9) does not match con
nection size (1) for port 'q'. The port definition is at: /acct/f1/jbakos/quartus_work/regn.v(1).
#         Region: /testbench/myadder/output_reg_cout

VSIM 31>
```

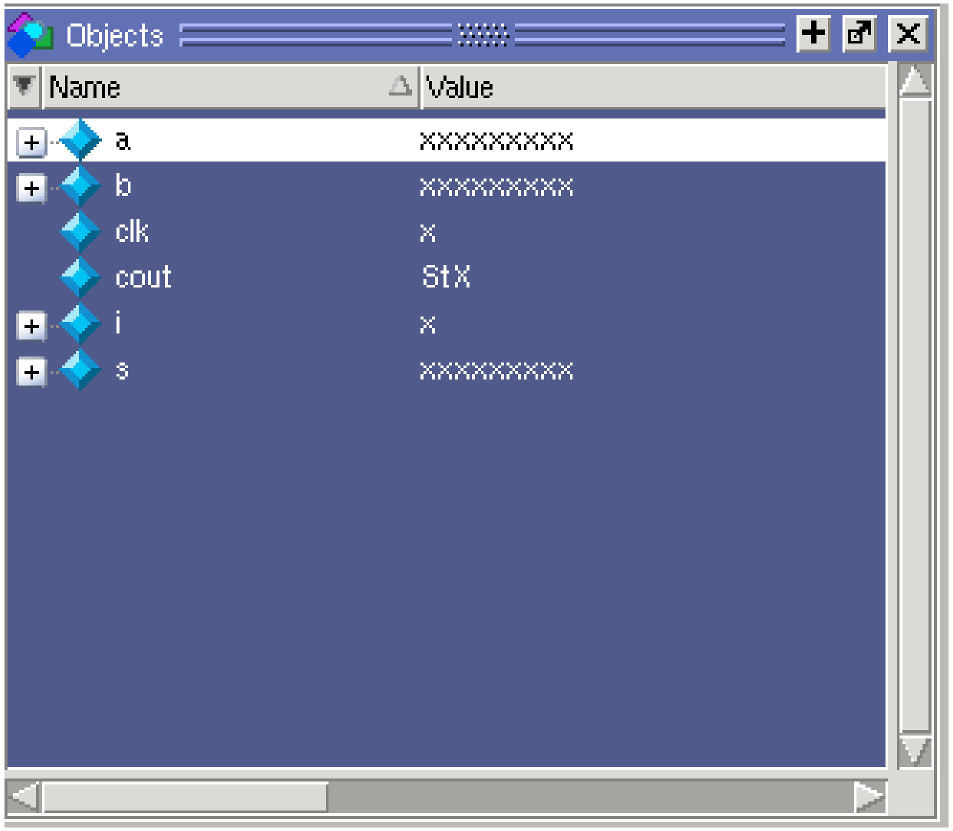
## Design Structure


sim - Default

| Instance | Design unit | Design ur |
|---|---|---|
| testbench | testbench | Module |
| myadder | adder_reg | Module |
| input_reg_a | regn | Module |
| #ALWAYS#3 | regn | Process |
| input_reg_b | regn | Module |
| #ALWAYS#3 | regn | Process |
| my_adder | adder | Module |
| adder_array[8] | adder | VlGenerat |
| adder_array[7] | adder | VlGenerat |
| adder_array[6] | adder | VlGenerat |
| adder_array[5] | adder | VlGenerat |
| adder_array[4] | adder | VlGenerat |
| adder_array[3] | adder | VlGenerat |
| adder_array[2] | adder | VlGenerat |
| adder_array[1] | adder | VlGenerat |
| adder_array[0] | adder | VlGenerat |
| #ASSIGN#13 | adder | Process |
| output_reg_s | regn | Module |
| #ALWAYS#3 | regn | Process |
| output_reg_cout | regn | Module |
| #ALWAYS#3 | regn | Process |
| #ALWAYS#9 | testbench | Process |
| #INITIAL#13 | testbench | Process |
| #vsim_capacity# | | Capacity |

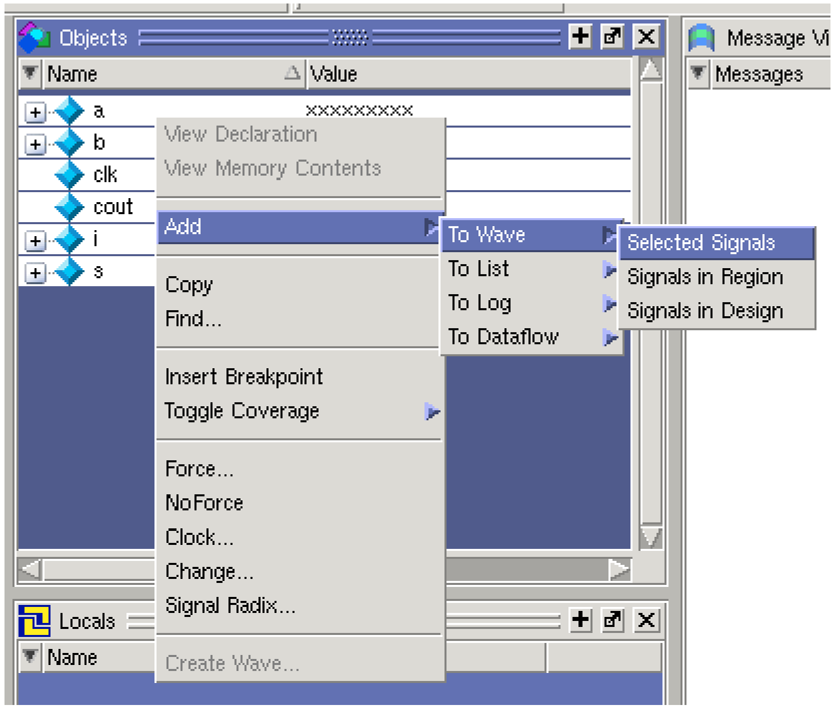
## Design Objects

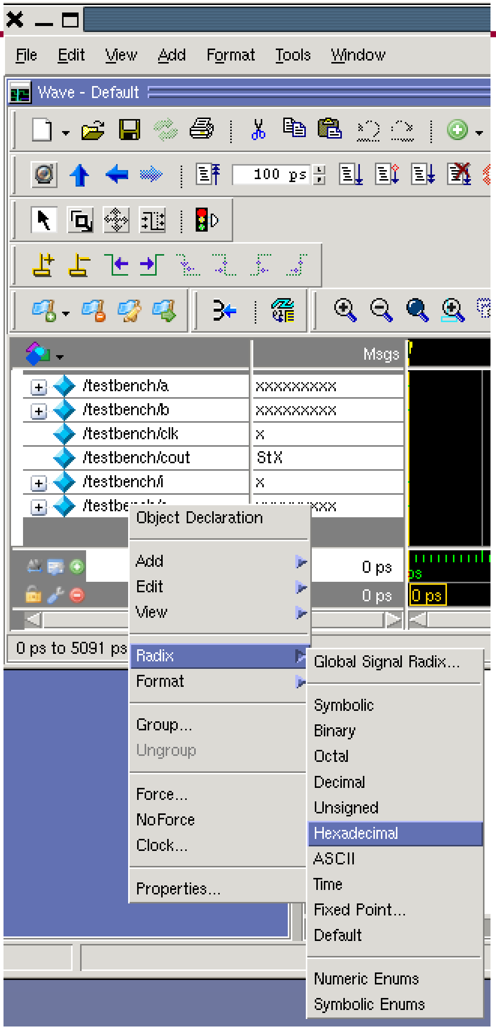
- Note: the contents of this pane depend on what is selected in structure pane



| Name | Value |
|---|---|
| a | xxxxxxxxx |
| b | xxxxxxxxx |
| clk | x |
| cout | StX |
| i | x |
| s | xxxxxxxxx |

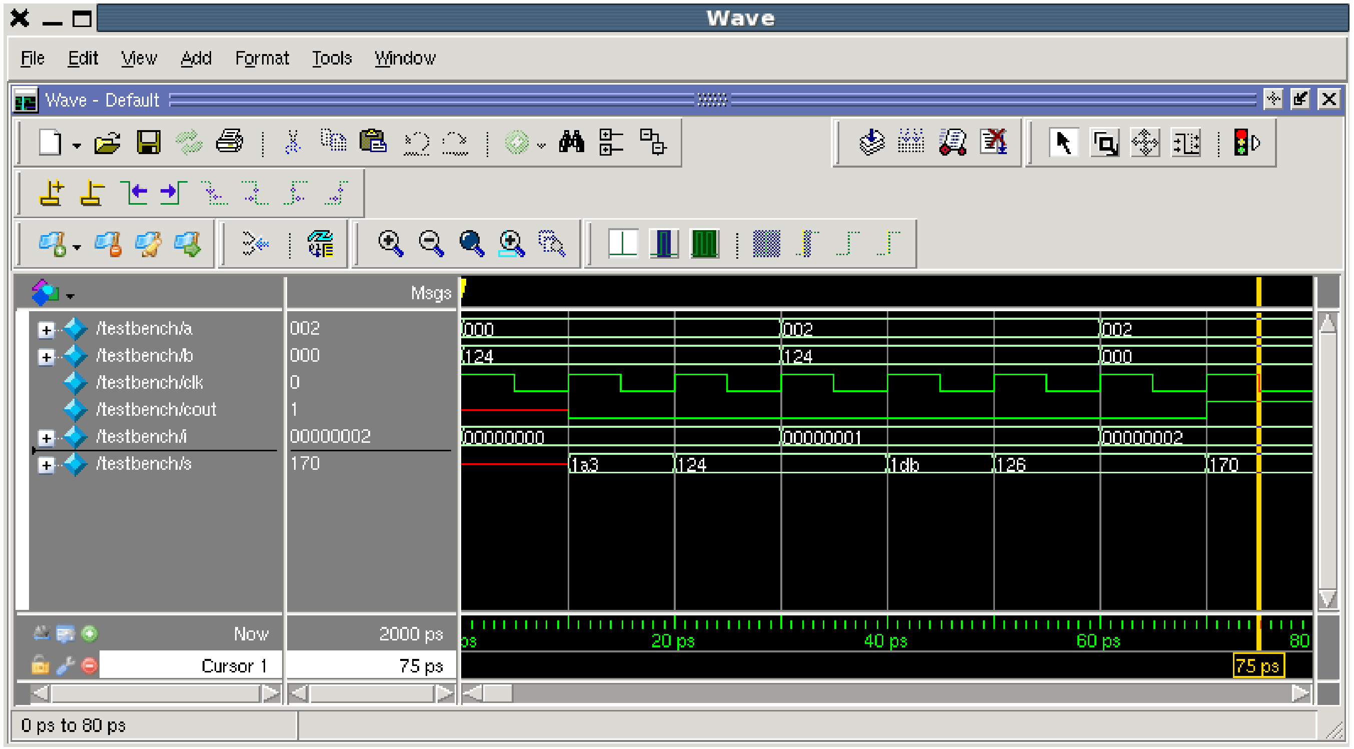
# Wave Window

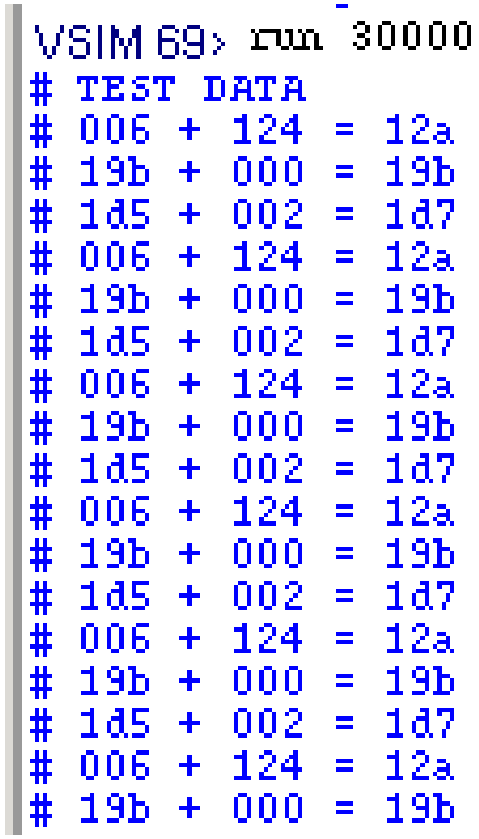

Objects
Name
Value
a
b
clk
cout
i
s
xxxxxxxxx
View Declaration
View Memory Contents
Add
Copy
Find...
Insert Breakpoint
Toggle Coverage
Force...
NoForce
Clock...
Change...
Signal Radix...
Create Wave...
To Wave
To List
To Log
To Dataflow
Selected Signals
Signals in Region
Signals in Design
Message Vi
Messages
Locals
Name


File
Edit
View
Add
Format
Tools
Window
Wave - Default
100 ps
Msgs
/testbench/a
/testbench/b
/testbench/clk
/testbench/cout
/testbench/i
/testbench/s
xxxxxxxxx
xxxxxxxxx
x
StX
x
0 ps
0 ps
0 ps to 5091 ps
Object Declaration
Add
Edit
View
Radix
Format
Group...
Ungroup
Force...
NoForce
Clock...
Properties...
Global Signal Radix...
Symbolic
Binary
Octal
Decimal
Unsigned
Hexadecimal
ASCII
Time
Fixed Point...
Default
Numeric Enums
Symbolic Enums

# Modelsim

- In transcript window, type "run 30000"





```
VSIM 69> run 30000
# TEST DATA
# 006 + 124 = 12a
# 19b + 000 = 19b
# 1d5 + 002 = 1d7
# 006 + 124 = 12a
# 19b + 000 = 19b
# 1d5 + 002 = 1d7
# 006 + 124 = 12a
# 19b + 000 = 19b
# 1d5 + 002 = 1d7
# 006 + 124 = 12a
# 19b + 000 = 19b
# 1d5 + 002 = 1d7
# 006 + 124 = 12a
# 19b + 000 = 19b
# 1d5 + 002 = 1d7
# 006 + 124 = 12a
# 19b + 000 = 19b
```

# Modelsim Notes

- Need to recompile after any changes
- “restart –f” will reload and restart simulation
- Signals must be added to wave or “log” in order to track
- You can add signals in lower levels of hierarchy by “drilling down” in structure and choosing signals in objects window

- Other useful Modelsim commands
  - force A X”DEADBEEF”
  - force op “0001”
  - force push 1
  - run 100
  - force -freeze clk 1 0, 0 {50 ps} -r 100

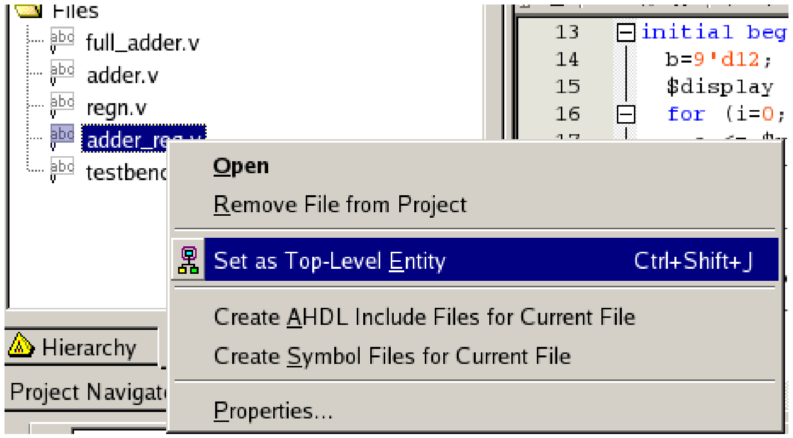
# Modelsim Notes

- Can also create a script for Modelsim:
- Example: test.do:

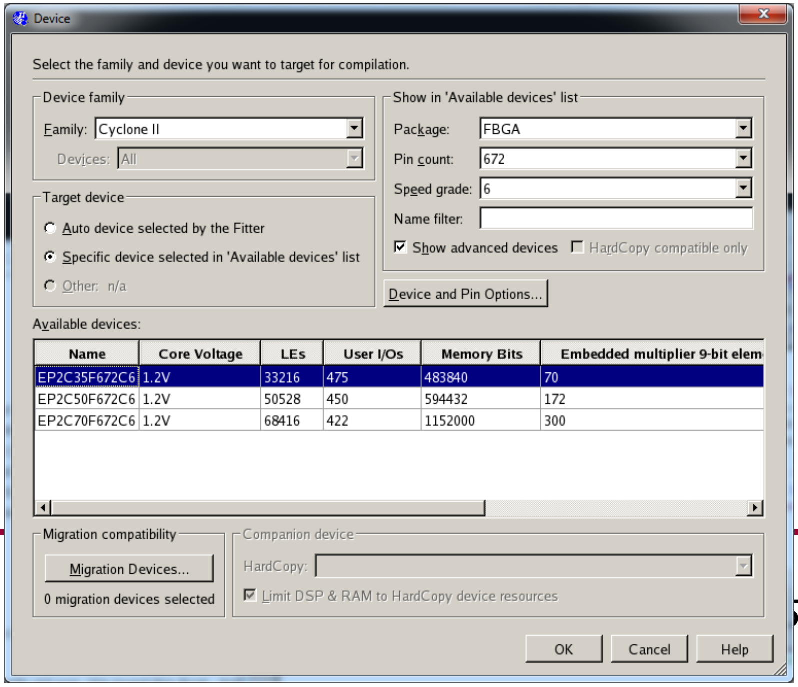
```
restart -f
force -freeze clk 1 0, 0 {50 ps} -r 100
force rst 1
force push 0
force pop 0
force data_in X"00000000"
run
force rst 0
force data_in X"00DECADE"
force push 1
run
force data_in X"00DEFACE"
run
force data_in X"0000FADE"
run
force data_in X"00DECAFE"
run
force pop 1
force push 0
run
run
run
run
force pop 0
run
```

# Synthesis

- Back to Quartus…
- Set adder_reg as top-level entity
- Select device
  - Assignments | Settings
  - Click "Device"
  - Select EP2C35F672C6 and click OK


Files
full_adder.v
adder.v
regn.v
adder_re
testbenc
Open
Remove File from Project
Set as Top-Level Entity Ctrl+Shift+J
Create AHDL Include Files for Current File
Create Symbol Files for Current File
Properties...
Hierarchy
Project Navigat

```
initial beg
  b=9'd12;
  $display
  for (i=0;
```



Select the family and device you want to target for compilation.

Device family
Family: Cyclone II
Devices: All

Show in 'Available devices' list
Package: FBGA
Pin count: 672
Speed grade: 6
Name filter:
☑ Show advanced devices ☐ HardCopy compatible only

Target device
○ Auto device selected by the Fitter
● Specific device selected in 'Available devices' list
○ Other: n/a

Device and Pin Options...

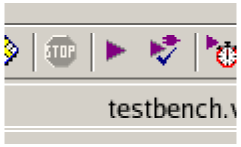
Available devices:

| Name | Core Voltage | LEs | User I/Os | Memory Bits | Embedded multiplier 9-bit elem |
|---|---|---|---|---|---|
| EP2C35F672C6 | 1.2V | 33216 | 475 | 483840 | 70 |
| EP2C50F672C6 | 1.2V | 50528 | 450 | 594432 | 172 |
| EP2C70F672C6 | 1.2V | 68416 | 422 | 1152000 | 300 |

Migration compatibility
Migration Devices...
0 migration devices selected

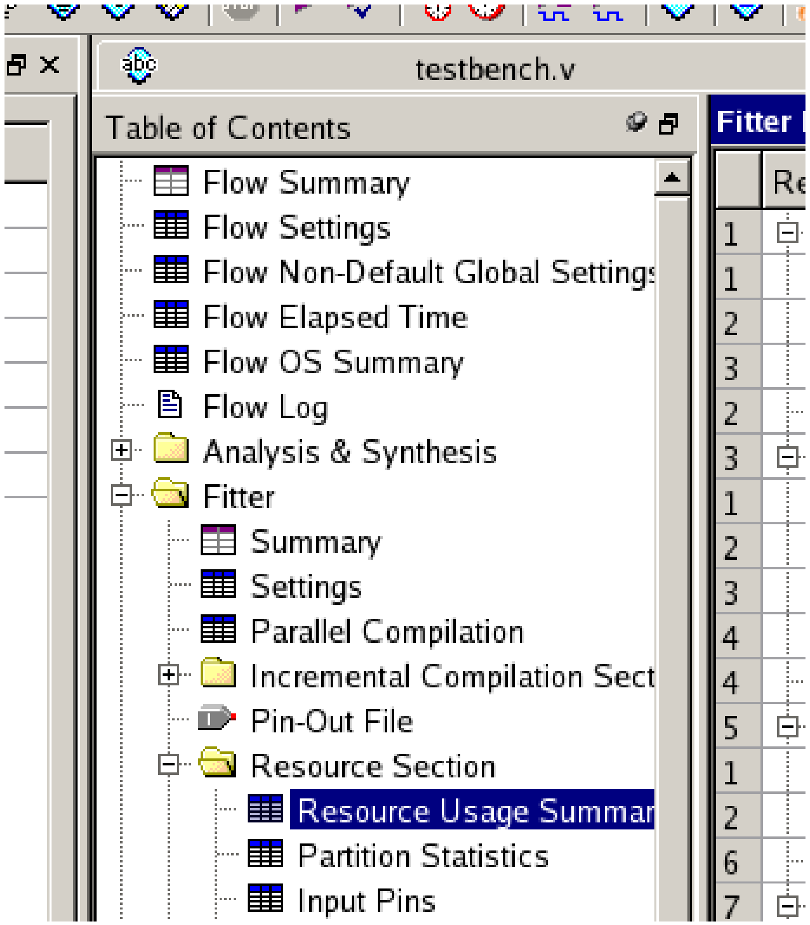
Companion device
HardCopy:
☑ Limit DSP & RAM to HardCopy device resources

# Synthesis

- To compile design, click start button 

testbench.v

- Once done, let's examine the FPGA resources required by the design



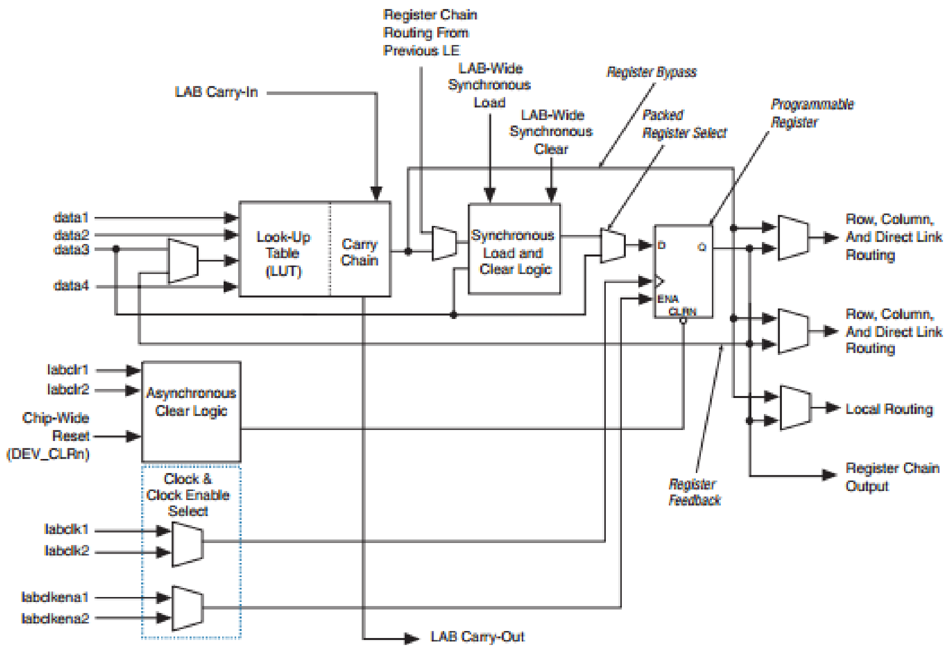
testbench.v

Table of Contents

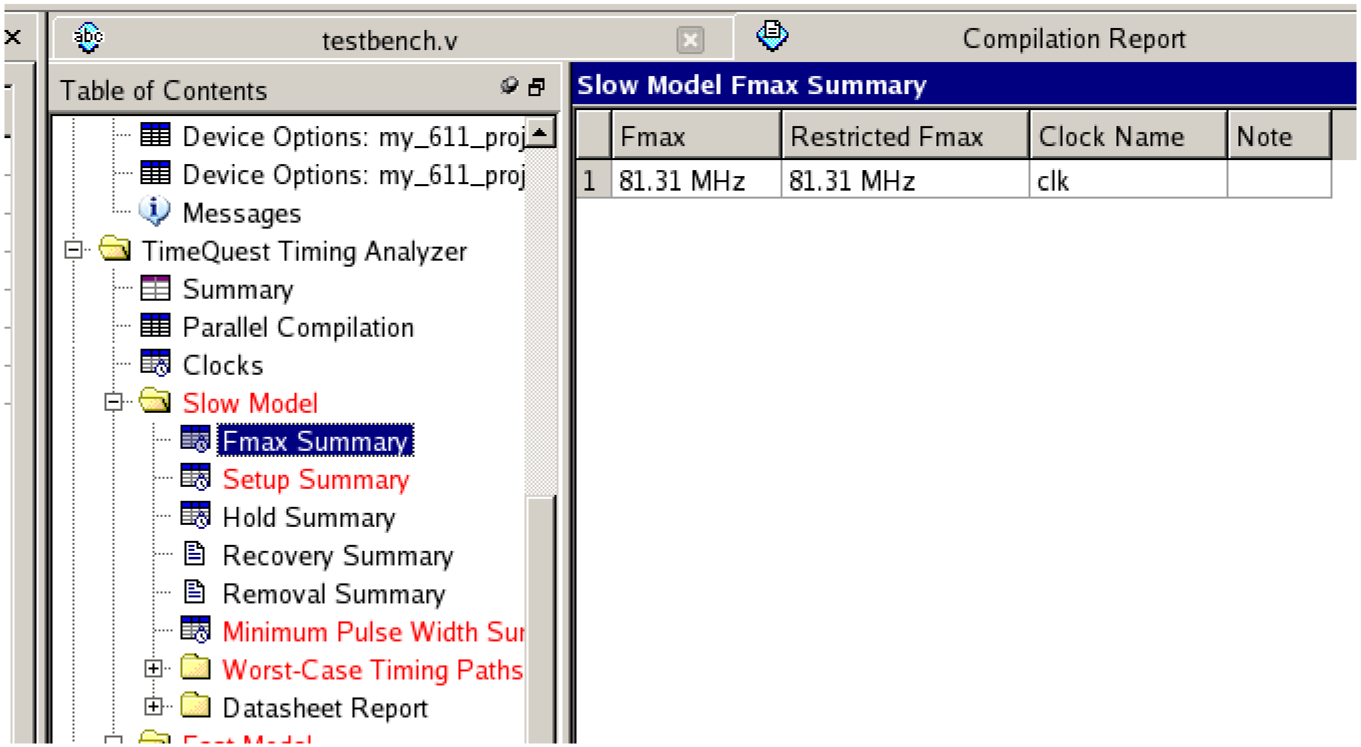
Flow Summary
Flow Settings
Flow Non-Default Global Settings
Flow Elapsed Time
Flow OS Summary
Flow Log
Analysis & Synthesis
Fitter
Summary
Settings
Parallel Compilation
Incremental Compilation Sect
Pin-Out File
Resource Section
Resource Usage Summar
Partition Statistics
Input Pins

# Resource Usage

- Logic elements
  - Cyclone 2 LE contains one 4-input LUT and one 1-bit register
  - Usage: 119 out of 33,216
    - 22 LUT only
    - 41 register only
    - 56 both
- No multiplier usage (out of 70)
- No M4K usage (out of 105)



## Clock Speed


testbench.v

Compilation Report

Table of Contents

- Device Options: my_611_proj
- Device Options: my_611_proj
- Messages
- TimeQuest Timing Analyzer
  - Summary
  - Parallel Compilation
  - Clocks
  - Slow Model
    - Fmax Summary
    - Setup Summary
    - Hold Summary
    - Recovery Summary
    - Removal Summary
    - Minimum Pulse Width Sur
    - Worst-Case Timing Paths
    - Datasheet Report

Slow Model Fmax Summary

| | Fmax | Restricted Fmax | Clock Name | Note |
|---|---|---|---|---|
| 1 | 81.31 MHz | 81.31 MHz | clk | |

## Adjust Adder

- Let's allow the synthesizer to build the adder architecture instead of us
  - Replace adder.v with:

```
module adder #(parameter width=32) (input [width-1:0] a,b,output [width-1:0] s,output
cout);

  wire [width:0] s_int;

  assign s_int = a + b;
  assign s = s_int[width-1:0];
  assign cout = s_int[width];

endmodule
```

- Re-simulate testbench to make sure it still works!
- New synthesis results:
  - 97 LEs (vs. 119)
  - 246.18 MHz (vs. 81.31 MHz!!!)

## JTAG

- Original idea: boundary scan
  - Originally used to test chip package and PCB connections
  - Chip inputs and outputs
  - “Scan chain”
- Now used as a way to source and probe internal signals
- Also used to program FPGAs
- “Joint Test Action Group” (JTAG) standard
  - Test clock (TCLK)
  - Test mode select (TMS)
  - Test data input (TDI)
  - Test data output (TDO)

## In System Testing

- Create a top-level wrapper, top.v:

```
module top (input CLOCK_50);

wire [31:0] a,b,s;
wire cout;

adder_reg(.clk(CLOCK_50),.a(a),.b(b),.s(s),.cout(cout));


endmodule
```

## Sources and Probes

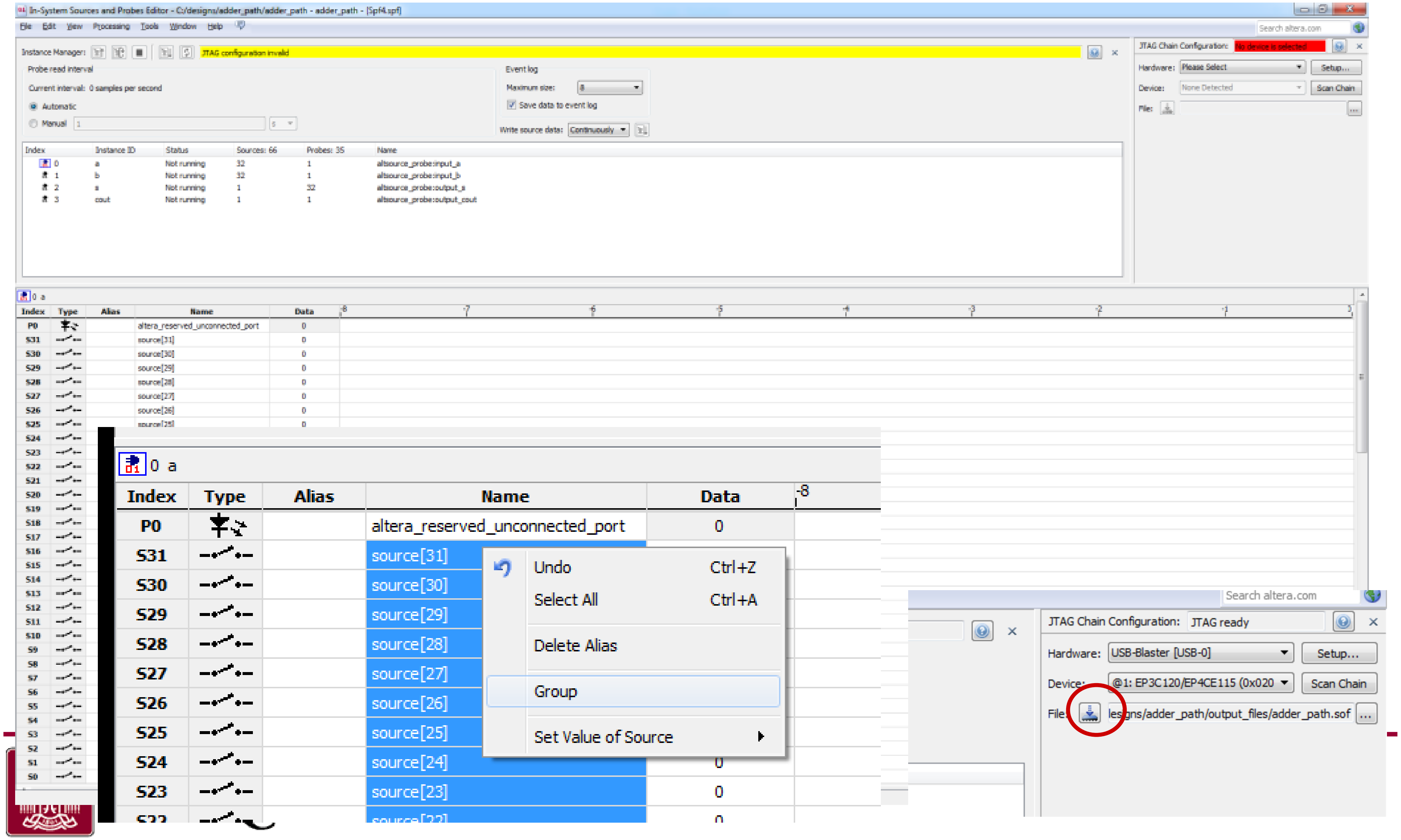
- Add sources and probes to the top-level:

```
altsource_probe #(.source_width(32),.instance_id("a")) input_a (.source(a),.probe());
altsource_probe #(.source_width(32),.instance_id("b")) input_b (.source(b),.probe());
altsource_probe #(.probe_width(32),.instance_id("s")) output_s (.source(),.probe(s));
altsource_probe #(.probe_width(1),.instance_id("cout")) output_cout (.source(),.probe(cout));
```

- Specify your top-level design and implement on FPGA

- Go to Tools | In-System Sources and Probes Editor

# Sources and Probes



In-System Sources and Probes Editor - C:/designs/adder_path/adder_path - adder_path - [Spf4.spf]

Instance Manager: JTAG configuration invalid

Probe read interval

Current interval: 0 samples per second

[✓ Automatic] Manual

Event log

Maximum size: 8

Save data to event log

Write source data: Continuously

| Index | Instance ID | Status | Sources: 66 | Probes: 35 | Name |
|---|---|---|---|---|---|
| 0 | a | Not running | 32 | 1 | altsource_probe:input_a |
| 1 | b | Not running | 32 | 1 | altsource_probe:input_b |
| 2 | s | Not running | 1 | 32 | altsource_probe:output_s |
| 3 | cout | Not running | 1 | 1 | altsource_probe:output_cout |

JTAG Chain Configuration: No device is selected

Hardware: Please Select

Device: None Detected

File:

0 a

| Index | Type | Alias | Name | Data |
|---|---|---|---|---|
| P0 | | | altera_reserved_unconnected_port | 0 |
| S31 | | | source[31] | 0 |
| S30 | | | source[30] | 0 |
| S29 | | | source[29] | 0 |
| S28 | | | source[28] | 0 |
| S27 | | | source[27] | 0 |
| S26 | | | source[26] | 0 |
| S25 | | | source[25] | 0 |
| S24 | | | source[24] | 0 |
| S23 | | | source[23] | 0 |
| S22 | | | source[22] | 0 |

JTAG Chain Configuration: JTAG ready

Hardware: USB-Blaster [USB-0]

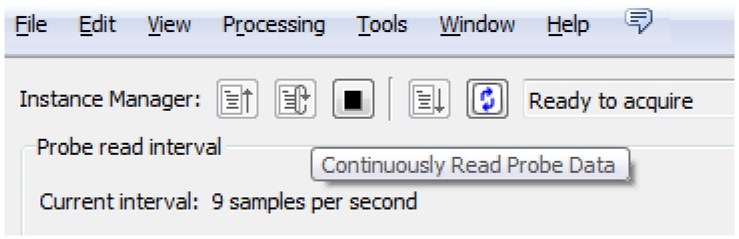
Device: @1: EP3C120/EP4CE115 (0x020

File: lesigns/adder_path/output_files/adder_path.sof

# Sources and Probes

- Highlight each of the four sources/probes and click the "continuously read probe data" button:

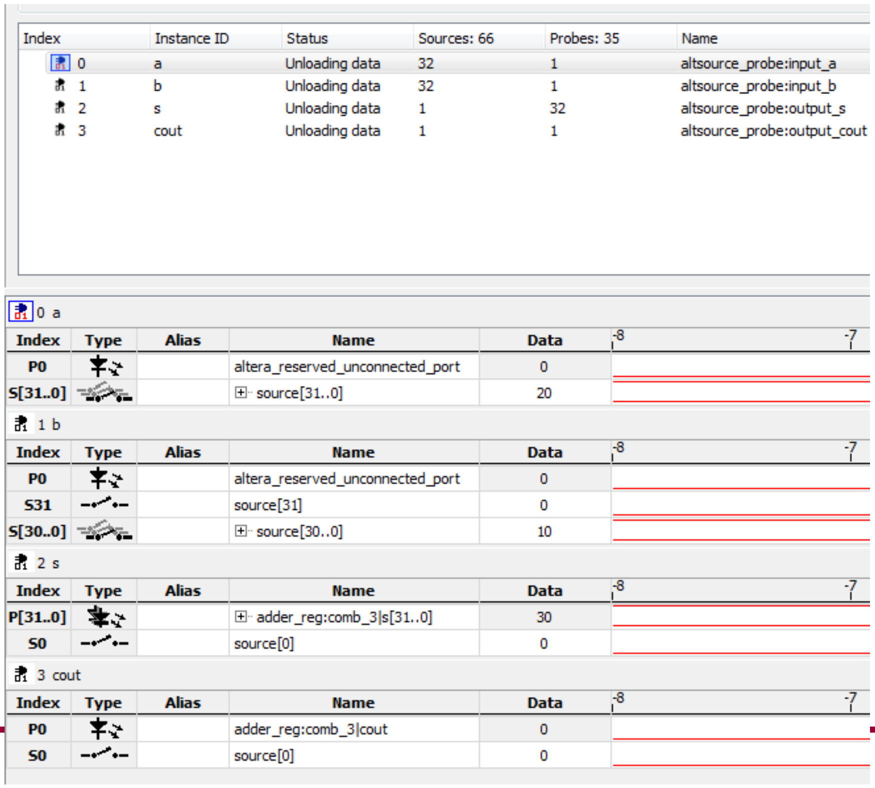
- Enter values for A and B and watch the output value update in real time



Instance Manager: Ready to acquire

Probe read interval

Current interval: 9 samples per second



| Index | Instance ID | Status | Sources: 66 | Probes: 35 | Name |
|---|---|---|---|---|---|
| 0 | a | Unloading data | 32 | 1 | altsource_probe:input_a |
| 1 | b | Unloading data | 32 | 1 | altsource_probe:input_b |
| 2 | s | Unloading data | 1 | 32 | altsource_probe:output_s |
| 3 | cout | Unloading data | 1 | 1 | altsource_probe:output_cout |

0 a

| Index | Type | Alias | Name | Data |
|---|---|---|---|---|
| P0 | | | altera_reserved_unconnected_port | 0 |
| S[31..0] | | | ⊞ source[31..0] | 20 |

1 b

| Index | Type | Alias | Name | Data |
|---|---|---|---|---|
| P0 | | | altera_reserved_unconnected_port | 0 |
| S31 | | | source[31] | 0 |
| S[30..0] | | | ⊞ source[30..0] | 10 |

2 s

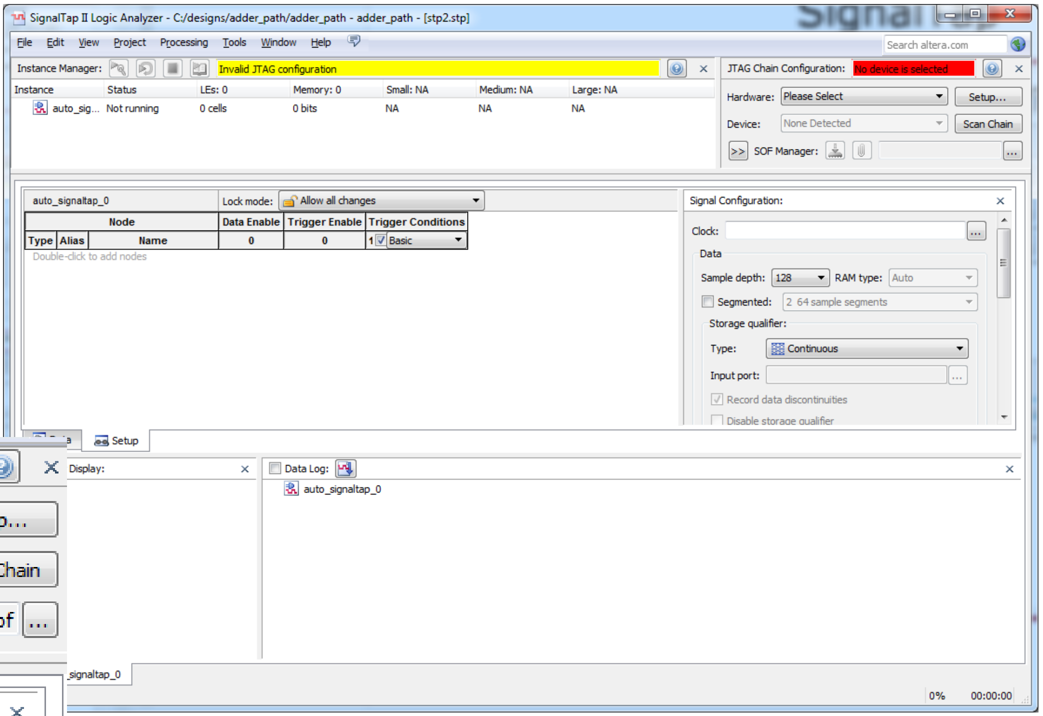
| Index | Type | Alias | Name | Data |
|---|---|---|---|---|
| P[31..0] | | | ⊞ adder_reg:comb_3\|s[31..0] | 30 |
| S0 | | | source[0] | 0 |

3 cout

| Index | Type | Alias | Name | Data |
|---|---|---|---|---|
| P0 | | | adder_reg:comb_3\|cout | 0 |
| S0 | | | source[0] | 0 |

# SignalTap

- Go to Tools | SignalTap II Logic Analyzer
- Set up sampling clock and SOF file:


SignalTap II Logic Analyzer - C:/designs/adder_path/adder_path - adder_path - [stp2.stp]

File Edit View Project Processing Tools Window Help

Instance Manager: Invalid JTAG configuration

| Instance | Status | LEs: 0 | Memory: 0 | Small: NA | Medium: NA | Large: NA |
|---|---|---|---|---|---|---|
| auto_sig... | Not running | 0 cells | 0 bits | NA | NA | NA |

JTAG Chain Configuration: No device is selected

Hardware: Please Select

Device: None Detected

SOF Manager:

auto_signaltap_0 Lock mode: Allow all changes

Double-click to add nodes

Signal Configuration:

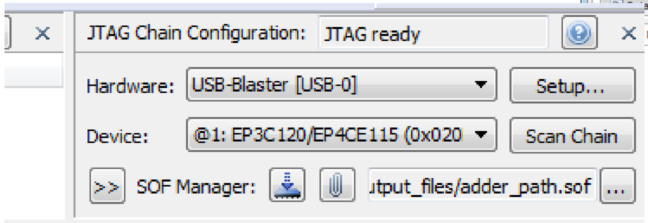
Clock:

Sample depth: 128 RAM type: Auto

Segmented: 2 64 sample segments

Storage qualifier: Type: Continuous

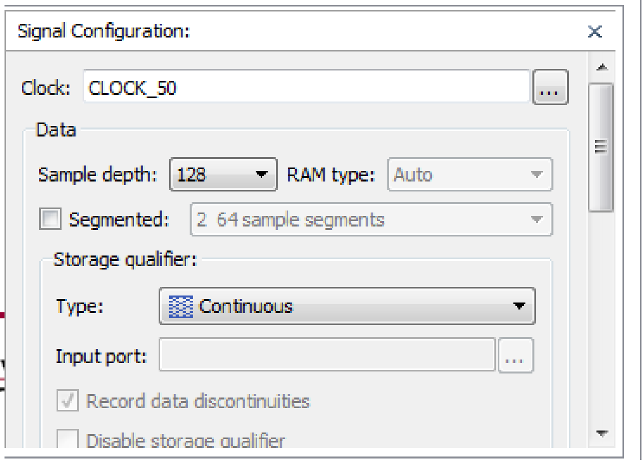
Data Log: auto_signaltap_0


JTAG Chain Configuration: JTAG ready

Hardware: USB-Blaster [USB-0]

Device: @1: EP3C120/EP4CE115 (0x020

SOF Manager: ıtput_files/adder_path.sof


Signal Configuration:

Clock: CLOCK_50

Data

Sample depth: 128 RAM type: Auto

Segmented: 2 64 sample segments

Storage qualifier:

Type: Continuous

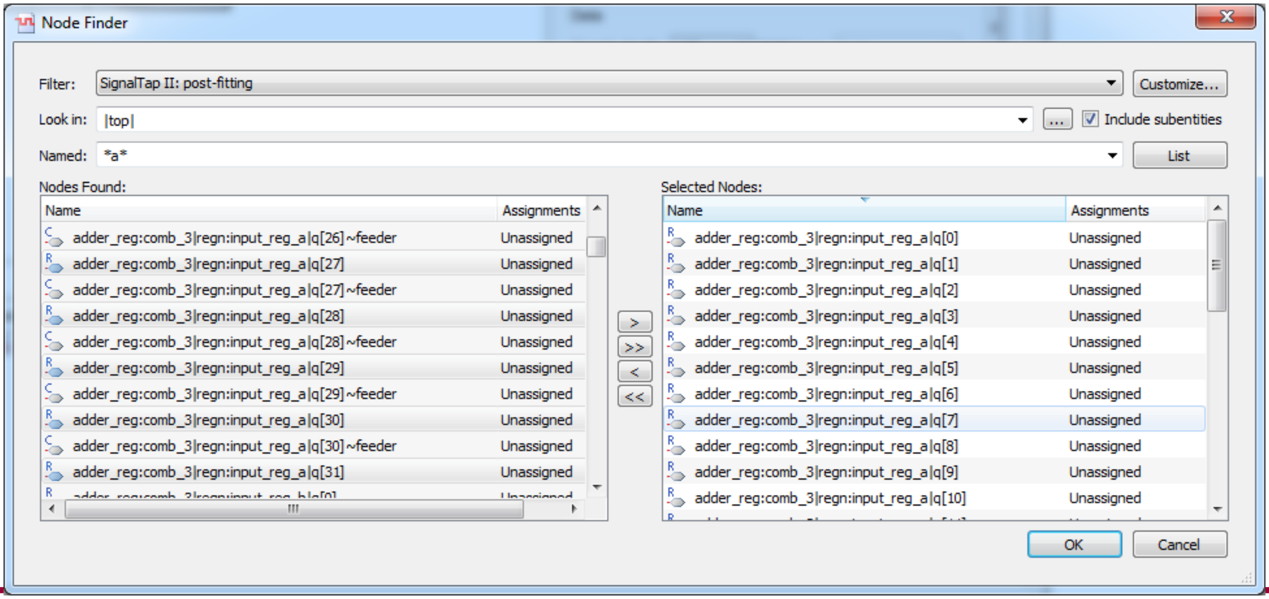
Input port:

☑ Record data discontinuities

☐ Disable storage qualifier

# SignalTap

- Right-click in middle pane and select "Add Nodes"
- Type "*a*" and click "List"
- Hold down control to select all the q bits from the a-register
- Click the ">" button
- Click OK



Filter: SignalTap II: post-fitting

Look in: |top|

[✓] Include subentities

Named: *a*

Nodes Found:

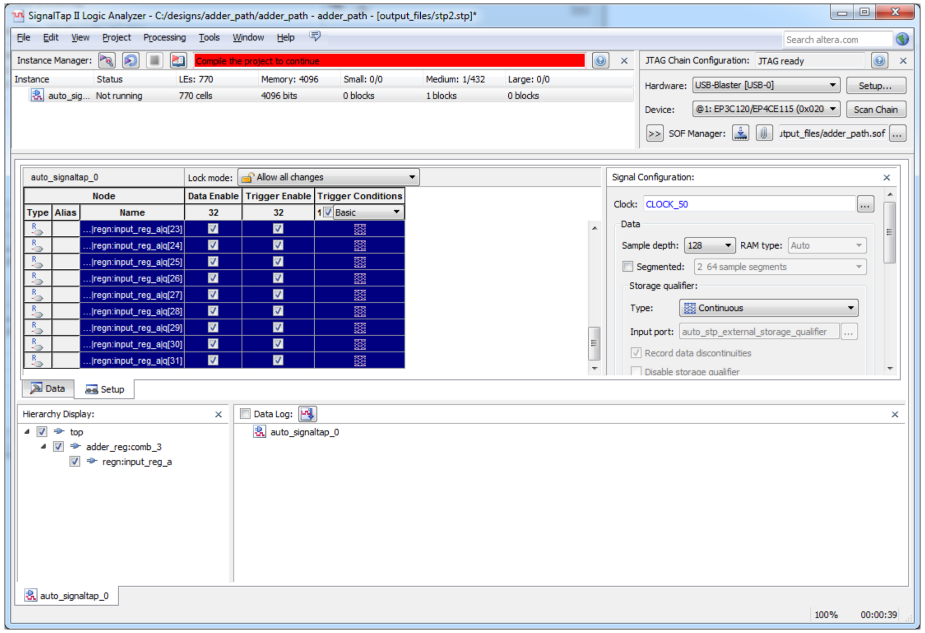
| Name | Assignments |
|---|---|
| adder_reg:comb_3\|regn:input_reg_a\|q[26]~feeder | Unassigned |
| adder_reg:comb_3\|regn:input_reg_a\|q[27] | Unassigned |
| adder_reg:comb_3\|regn:input_reg_a\|q[27]~feeder | Unassigned |
| adder_reg:comb_3\|regn:input_reg_a\|q[28] | Unassigned |
| adder_reg:comb_3\|regn:input_reg_a\|q[28]~feeder | Unassigned |
| adder_reg:comb_3\|regn:input_reg_a\|q[29] | Unassigned |
| adder_reg:comb_3\|regn:input_reg_a\|q[29]~feeder | Unassigned |
| adder_reg:comb_3\|regn:input_reg_a\|q[30] | Unassigned |
| adder_reg:comb_3\|regn:input_reg_a\|q[30]~feeder | Unassigned |
| adder_reg:comb_3\|regn:input_reg_a\|q[31] | Unassigned |
| adder_reg:comb_3\|regn:input_reg_b\|q[0] | Unassigned |

Selected Nodes:

| Name | Assignments |
|---|---|
| adder_reg:comb_3\|regn:input_reg_a\|q[0] | Unassigned |
| adder_reg:comb_3\|regn:input_reg_a\|q[1] | Unassigned |
| adder_reg:comb_3\|regn:input_reg_a\|q[2] | Unassigned |
| adder_reg:comb_3\|regn:input_reg_a\|q[3] | Unassigned |
| adder_reg:comb_3\|regn:input_reg_a\|q[4] | Unassigned |
| adder_reg:comb_3\|regn:input_reg_a\|q[5] | Unassigned |
| adder_reg:comb_3\|regn:input_reg_a\|q[6] | Unassigned |
| adder_reg:comb_3\|regn:input_reg_a\|q[7] | Unassigned |
| adder_reg:comb_3\|regn:input_reg_a\|q[8] | Unassigned |
| adder_reg:comb_3\|regn:input_reg_a\|q[9] | Unassigned |
| adder_reg:comb_3\|regn:input_reg_a\|q[10] | Unassigned |

# SignalTap

- Right-click newly-added signals and select "group"



SignalTap II Logic Analyzer - C:/designs/adder_path/adder_path - adder_path - [output_files/stp2.stp]*

Instance Manager: Compile the project to continue

JTAG Chain Configuration: JTAG ready

| Instance | Status | LEs: 770 | Memory: 4096 | Small: 0/0 | Medium: 1/432 | Large: 0/0 |
|---|---|---|---|---|---|---|
| auto_sig... | Not running | 770 cells | 4096 bits | 0 blocks | 1 blocks | 0 blocks |

Hardware: USB-Blaster [USB-0]
Device: @1: EP3C120/EP4CE115 (0x020
SOF Manager: ...tput_files/adder_path.sof

auto_signaltap_0 — Lock mode: Allow all changes

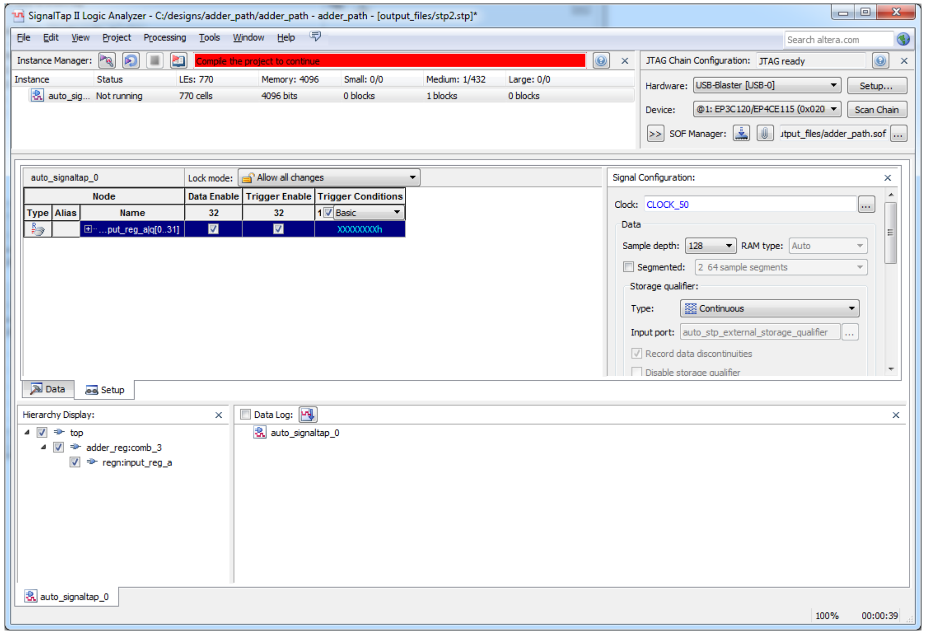
| Type | Alias | Name | Data Enable 32 | Trigger Enable 32 | Trigger Conditions 1 Basic |
|---|---|---|---|---|---|
| | | ...|regn:input_reg_a\|q[23] | ✓ | ✓ | ▨ |
| | | ...|regn:input_reg_a\|q[24] | ✓ | ✓ | ▨ |
| | | ...|regn:input_reg_a\|q[25] | ✓ | ✓ | ▨ |
| | | ...|regn:input_reg_a\|q[26] | ✓ | ✓ | ▨ |
| | | ...|regn:input_reg_a\|q[27] | ✓ | ✓ | ▨ |
| | | ...|regn:input_reg_a\|q[28] | ✓ | ✓ | ▨ |
| | | ...|regn:input_reg_a\|q[29] | ✓ | ✓ | ▨ |
| | | ...|regn:input_reg_a\|q[30] | ✓ | ✓ | ▨ |
| | | ...|regn:input_reg_a\|q[31] | ✓ | ✓ | ▨ |

Signal Configuration:
Clock: CLOCK_50
Data
Sample depth: 128 RAM type: Auto
Segmented: 2 64 sample segments
Storage qualifier:
Type: Continuous
Input port: auto_stp_external_storage_qualifier
Record data discontinuities
Disable storage qualifier

Hierarchy Display: top / adder_reg:comb_3 / regn:input_reg_a

Data Log: auto_signaltap_0

100% 00:00:39



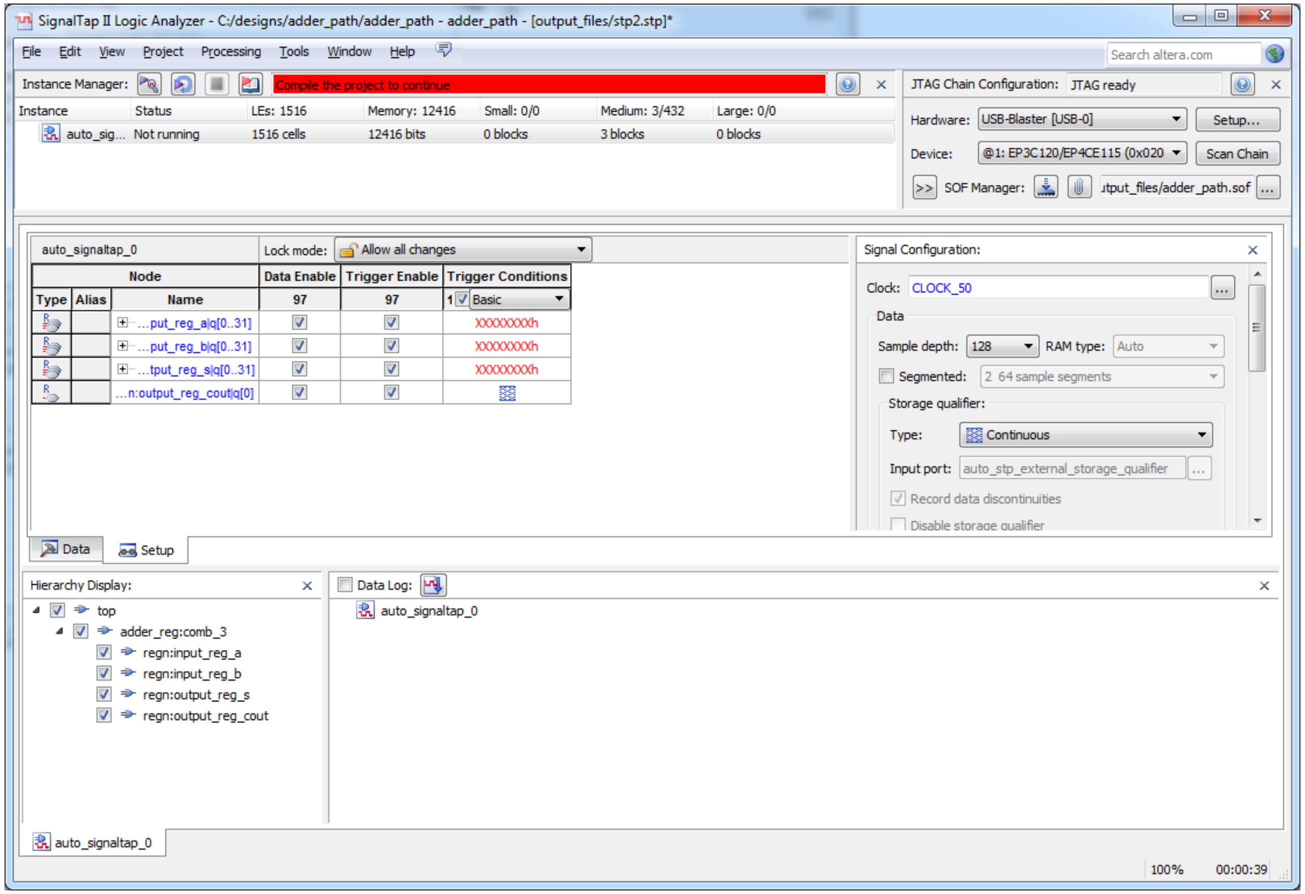
SignalTap II Logic Analyzer - C:/designs/adder_path/adder_path - adder_path - [output_files/stp2.stp]*

| Type | Alias | Name | Data Enable 32 | Trigger Enable 32 | Trigger Conditions 1 Basic |
|---|---|---|---|---|---|
| | | ⊞ ...put_reg_a\|q[0..31] | ✓ | ✓ | XXXXXXXXh |

Signal Configuration:
Clock: CLOCK_50
Sample depth: 128 RAM type: Auto
Segmented: 2 64 sample segments
Type: Continuous
Input port: auto_stp_external_storage_qualifier

# SignalTap

- Repeat process for b, s, and cout


SignalTap II Logic Analyzer - C:/designs/adder_path/adder_path - adder_path - [output_files/stp2.stp]*

Instance Manager: Compile the project to continue

| Instance | Status | LEs: 1516 | Memory: 12416 | Small: 0/0 | Medium: 3/432 | Large: 0/0 |
|---|---|---|---|---|---|---|
| auto_sig... | Not running | 1516 cells | 12416 bits | 0 blocks | 3 blocks | 0 blocks |

JTAG Chain Configuration: JTAG ready
Hardware: USB-Blaster [USB-0]
Device: @1: EP3C120/EP4CE115 (0x020
SOF Manager: .tput_files/adder_path.sof

auto_signaltap_0 — Lock mode: Allow all changes

| Type | Alias | Name | Data Enable 97 | Trigger Enable 97 | Trigger Conditions 1 ✓ Basic |
|---|---|---|---|---|---|
| | | ...put_reg_a\|q[0..31] | ✓ | ✓ | XXXXXXXXh |
| | | ...put_reg_b\|q[0..31] | ✓ | ✓ | XXXXXXXXh |
| | | ...tput_reg_s\|q[0..31] | ✓ | ✓ | XXXXXXXXh |
| | | ...n:output_reg_cout\|q[0] | ✓ | ✓ | ▨ |

Signal Configuration:
Clock: CLOCK_50
Data — Sample depth: 128 RAM type: Auto
Segmented: 2 64 sample segments
Storage qualifier: Type: Continuous
Input port: auto_stp_external_storage_qualifier
✓ Record data discontinuities
Disable storage qualifier

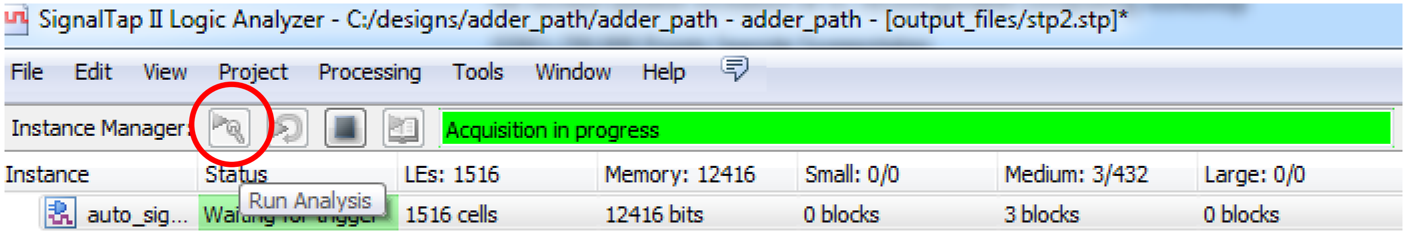
Hierarchy Display:
- top
  - adder_reg:comb_3
    - regn:input_reg_a
    - regn:input_reg_b
    - regn:output_reg_s
    - regn:output_reg_cout

Data Log: auto_signaltap_0

100% 00:00:39

# SignalTap

- Set value of "FFFFFFFF" under trigger for A and click "run analysis"

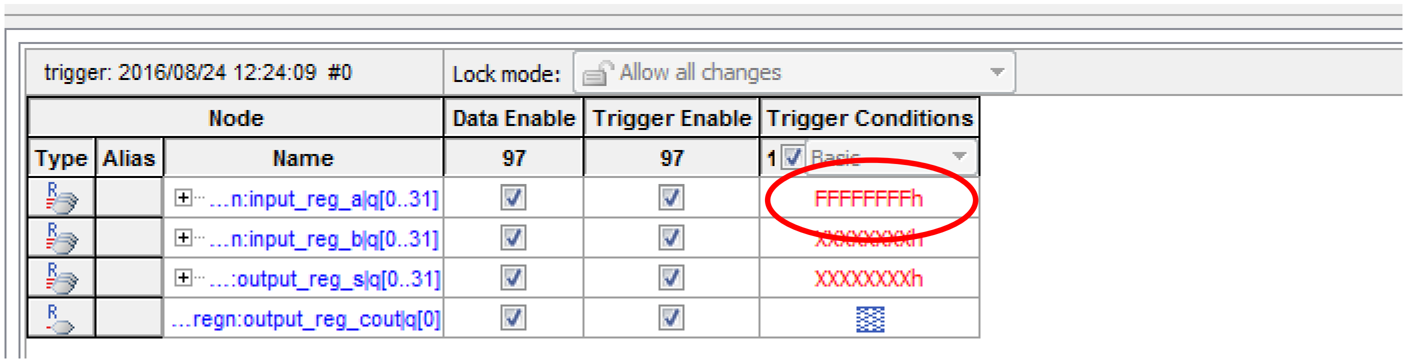


SignalTap II Logic Analyzer - C:/designs/adder_path/adder_path - adder_path - [output_files/stp2.stp]*

File Edit View Project Processing Tools Window Help

Instance Manager: Run Analysis — Acquisition in progress

| Instance | Status | LEs: 1516 | Memory: 12416 | Small: 0/0 | Medium: 3/432 | Large: 0/0 |
|---|---|---|---|---|---|---|
| auto_sig... | Waiting for trigger | 1516 cells | 12416 bits | 0 blocks | 3 blocks | 0 blocks |

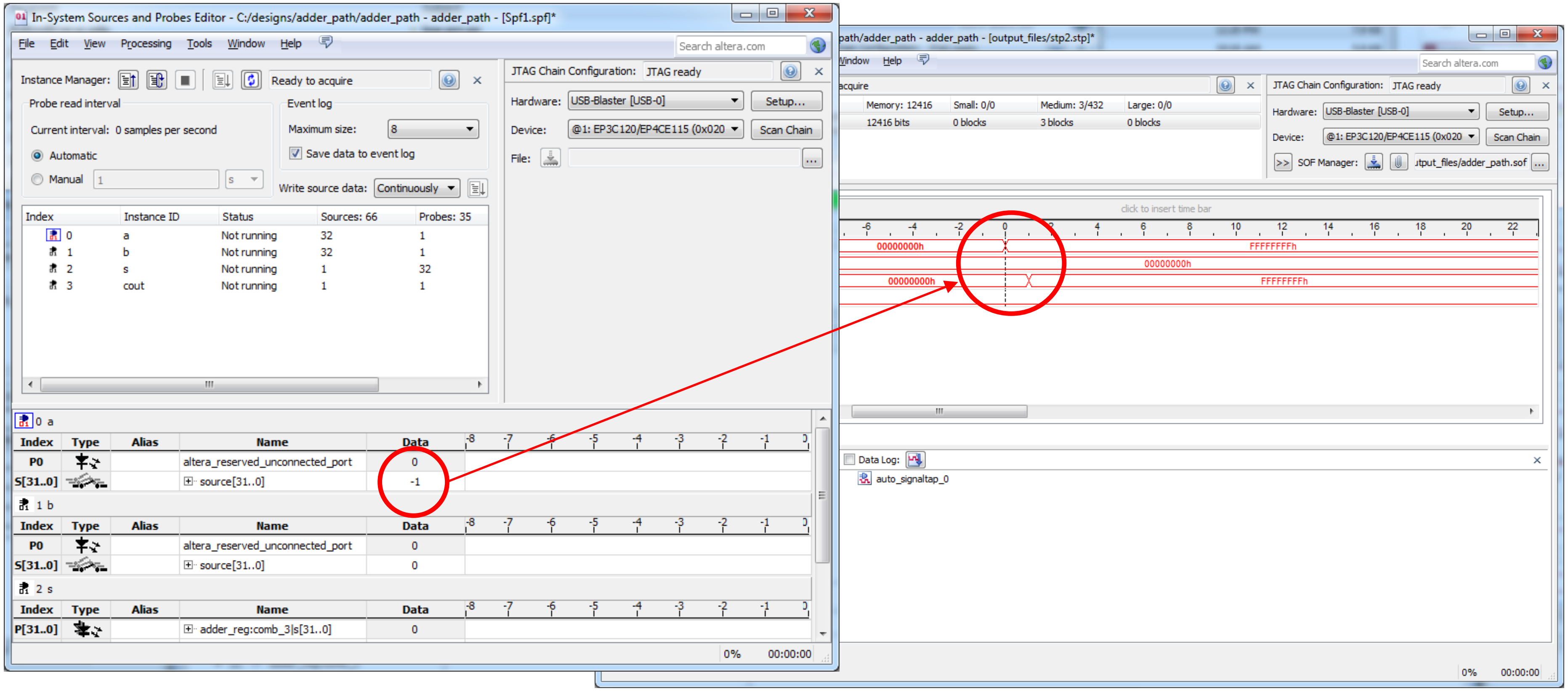


trigger: 2016/08/24 12:24:09 #0 — Lock mode: Allow all changes

| Type | Alias | Name | Data Enable 97 | Trigger Enable 97 | Trigger Conditions 1 ✓ Basic |
|---|---|---|---|---|---|
| | | ...n:input_reg_a\|q[0..31] | ✓ | ✓ | FFFFFFFFh |
| | | ...n:input_reg_b\|q[0..31] | ✓ | ✓ | XXXXXXXXh |
| | | ...:output_reg_s\|q[0..31] | ✓ | ✓ | XXXXXXXXh |
| | | ...regn:output_reg_cout\|q[0] | ✓ | ✓ | ▨ |

# SignalTap

- Back in "In System Sources and Probes", set A to -1



In-System Sources and Probes Editor - C:/designs/adder_path/adder_path - adder_path - [Spf1.spf]*

Instance Manager: Ready to acquire

Probe read interval

Current interval: 0 samples per second

Automatic

Manual 1 s

Event log

Maximum size: 8

Save data to event log

Write source data: Continuously

JTAG Chain Configuration: JTAG ready

Hardware: USB-Blaster [USB-0]

Device: @1: EP3C120/EP4CE115 (0x020

| Index | Instance ID | Status | Sources: 66 | Probes: 35 |
|---|---|---|---|---|
| 0 | a | Not running | 32 | 1 |
| 1 | b | Not running | 32 | 1 |
| 2 | s | Not running | 1 | 32 |
| 3 | cout | Not running | 1 | 1 |

0 a

| Index | Type | Alias | Name | Data |
|---|---|---|---|---|
| P0 | | | altera_reserved_unconnected_port | 0 |
| S[31..0] | | | source[31..0] | -1 |

1 b

| Index | Type | Alias | Name | Data |
|---|---|---|---|---|
| P0 | | | altera_reserved_unconnected_port | 0 |
| S[31..0] | | | source[31..0] | 0 |

2 s

| Index | Type | Alias | Name | Data |
|---|---|---|---|---|
| P[31..0] | | | adder_reg:comb_3\|s[31..0] | 0 |

0% 00:00:00

path/adder_path - adder_path - [output_files/stp2.stp]*

acquire

| Memory: 12416 | Small: 0/0 | Medium: 3/432 | Large: 0/0 |
|---|---|---|---|
| 12416 bits | 0 blocks | 3 blocks | 0 blocks |

JTAG Chain Configuration: JTAG ready

Hardware: USB-Blaster [USB-0]

Device: @1: EP3C120/EP4CE115 (0x020

SOF Manager: utput_files/adder_path.sof

click to insert time bar

00000000h FFFFFFFFh

00000000h

00000000h FFFFFFFFh

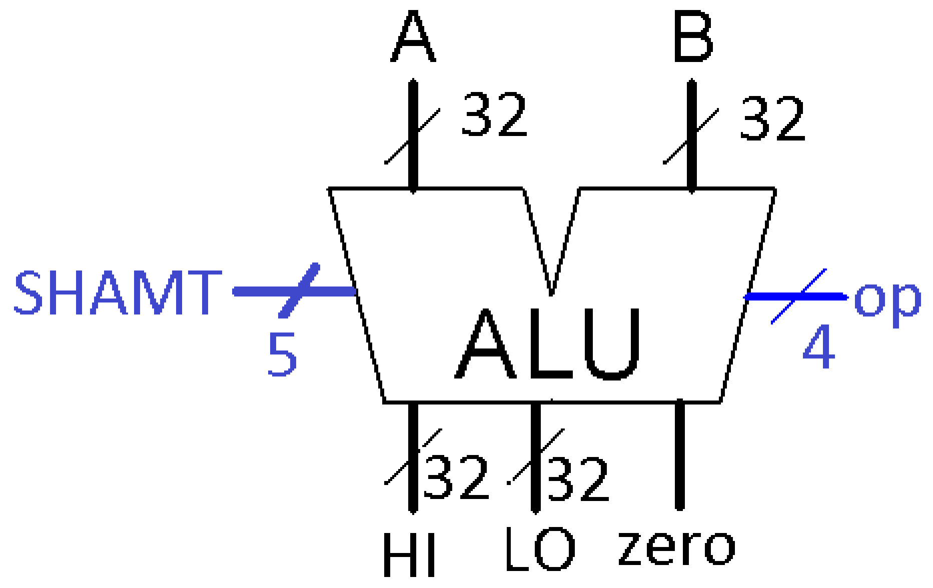
Data Log: auto_signaltap_0

0% 00:00:00

## ALU Design

- Let's create our MIPS ALU
- Requirements:
  - 32-bit
  - Implement 13 operations:
    - ADD, SUB, MULT, MULTU
    - AND, OR, NOR, XOR
    - SLT, SLTU
    - SLL, SRL, SRA
  - Interface:
    - input [31:0] A, B
    - input [3:0] op
    - input [4:0] shamt
    - output [31:0] hi, lo
    - output zero

# Arithmetic Logic Unit (ALU)



| op | Function |
|---|---|
| 0000 | A and B |
| 0001 | A or B |
| 0010 | A nor B |
| 0011 | A xor B |
| 0100 | A + B |
| 0101 | A – B |
| 0110 | A * B (signed) |
| 0111 | A* B (unsigned) |
| 1000 | B << shamt |
| 1001 | B >> shamt |
| 1010 | B >>> shamt |
| 1011 | B >>> shamt |
| 1100 | A < B (signed) |
| 1101 | A < B (unsigned) |
| 1110 | A < B (unsigned) |
| 1111 | A < B (unsigned) |

## Set Less Than

- Basic idea: set result to 00000000000000000000000000000001 if A < B else 00000000000000000000000000000000

- To do this, compute DIFF = A – B

- For signed SLT, look at sign bits of A, B, and DIFF
  - if A is negative and B is positive, set result to 1
  - if both positive and negative DIFF, set result to 1
    - Example: 2 – 4 = -2, 4 – 2 = 2
  - if both negative and negative DIFF, set result 1
    - Example -4 – (-2) = -2, -2 – (-4) = 2

## Set Less Than

- For unsigned, look at carryout
  - Example for 4-bit operands:

```
 4  =  0100 -> 0100 ->    0100
-2  =  0010 -> 1101 -> + 1110
                          0010 carryout = 1


  2  = 0010 -> 0010 ->    0010
-  4 = 0100 -> 1011 -> + 1100
                          1110 carryout = 0

 2  = 0010 -> 0010 ->    0010
-2  = 0010 -> 1101 -> + 1110
                          0000 carryout = 1
```

- If carryout = 0, A-B is < 0 therefore A < B

## Shifters

- **Logical shifter:** shifts value to left or right and fills empty spaces with 0's
  - Ex: 11001 >> 2 = 00110
  - Ex: 11001 << 2 = 00100

- **Arithmetic shifter:** same as logical shifter, but on right shift, fills empty spaces with the old most significant bit (msb).
  - Ex: 11001 >>> 2 = **11**110
  - Ex: 11001 <<< 2 = 00100

# ALU Design 1

```
module alu (input [31:0] a,b,
                         input[3:0] op,
                         input [4:0] shamt,
                         output reg [31:0] hi,lo,
                         output zero);

wire [32:0] diff = {1'b0,a}+{1'b0,~b+32'b1};

assign zero = lo==32'd0 ? 1'b1 : 1'b0;

always @(*) begin
        hi = 32'b0;
        lo = 32'b0;
```

## ALU Design 2

```
casez (op)
        // arithmetic operations
        4'b01_00: lo = a+b; // add
        4'b01_01: lo = a-b; // sub

        // mult signed
        4'b01_10: {hi,lo} = $signed(a)*$signed(b);

        // mult unsigned
        4'b01_11: {hi,lo} = a*b;
```

## ALU Design 3

```
// shifter operations
4'b10_00: lo = b << shamt; // sll
4'b10_01: lo = b >> shamt; // srl
4'b10_1?: lo = $signed(b) >>> shamt; //sra
```

## ALU Design 4

```
// comparison operations
4'b11_00:
if (a[31] & ~b[31])
        lo = 32'b1; // a neg, b pos
else
        if (a[31] == b[31] & diff[31])
                // same sign, diff is neg
                lo = 32'b1;
        else lo = 32'b0;

4'b11_??:
if (~diff[32]) lo = 32'b1; // sltu
else lo = 32'b0;
```

# ALU Design 5

```
          // logical operations
          4'b00_00: lo = a & b; // and
          4'b00_01: lo = a | b; // or
          4'b00_10: lo = ~(a | b); // nor
          4'b00_11: lo = a ^ b; // xor
      endcase

  end // always

  endmodule
```

## ALU Design 5

```
module alu_reg (input clk,input[31:0] a,b, input[3:0] op, input [4:0] shamt, output
[31:0] hi,lo, output zero);

wire [31:0] a_reg,b_reg,hi_comb,lo_comb;
wire [3:0] op_reg;
wire [4:0] shamt_reg;
wire zero_comb;

regn #(32) input_reg_a(clk,a,a_reg);
regn #(32) input_reg_b(clk,b,b_reg);
regn #(4) input_reg_op(clk,op,op_reg);
regn #(5) input_reg_shamt(clk,shamt,shamt_reg);

alu my_alu(.a(a_reg), .b(b_reg), .shamt(shamt_reg), .op(op_reg), .hi(hi_comb),
.lo(lo_comb), .zero(zero_comb));

regn #(32) output_reg_hi(clk,hi_comb,hi);
regn #(32) output_reg_lo(clk,lo_comb,lo);
regn #(1) output_reg_zero(clk,zero_comb,zero);

endmodule
```

## Synthesis Results

- ALU_reg requires:
  - 831/33216 LEs (3%)
  - 16 9-bit multipliers (signed and unsigned occur in parallel)
  - 94.95 MHz

## Lab 1

- Objective:
  - Create a self-checking testbench for the ALU
  - Create a test vectors file
  - Must cover all the test cases listed in the lab document
  - Make sure you describe any bugs you find in the design
- What to submit:
  - Test vector file
  - Corresponding file that describes each test vector
  - Testbench Verilog
  - Demo using SignalTap (using CLOCK_50)
- Due: 9/30 by midnight