ECE 429-03 (Spring 2015)

INTRODUCTION TO VLSI

PROJECT REPORT

DESIGN AND SYNTHESIS
OF
CARRY PROPAGATE ADDERS

Prof: Dr.Erdal Oruklu

TA: Dr. Cecilia Garcia Martin

Venkata Sai Sarath Marepalli

(A20319735)

CONTRIBUTIONS

All The project has been done by myself, and I assure you that I did not get any outside assistance other than those cited in the reference.

…Venkata Sai Sarath Marepalli

(A20319735)

ABSTRACTION

Addition is the basic calculation of any computation. Different kinds of adders are theoretically present in which one adder is more efficient than the other, but in order to get the performance out of these adders the investment is hardware which is a simple logic, invest more hardware to get better performance in this project we will come across 4 of those kind of adders and we will compare the performance between them at hardware's cost.

OBJECTIVE

The main aim of this project is to get familiar with the concepts like data-path circuit design, standard cell design flow, design validation and verification, so in-order to achieve that we implement Carry Propagation Adders in verilog and then verify its design using the commercial tools from 'synopsys' and Cadence design systems.

INTRODUCTION

In this project we will design 8 adders in total two for each kind using structural verilog code and then perform various simulations on it to verify its working in both physical and functional level and project the results and compare them to the expected ones. In this project we have to design both 8-bit and 32-bit adders for 4 kind of adders.

The main challenge in this design is to design a Koggstone adder because the Koggestone adder consists of a complex circuit with a number of inter connecting elements and thus instead of designing 8bit adder with two 4 bit modules we try to implement a whole 8-bit adder and then use it to design a 32 bit module.

BACKGROUND

In this project the first adder we design is a carry ripple adder in which the addition process takes place as a regular addition or in other words the human way of doing any addition we first take a bit add it to the first bit of the other number and generate a carry and a sum as outputs and thus the first iteration ends zand then the same process is carried out till the last bit and at last the interconnected cayyouts produce a final carry out and the sum is produced as 1 bit per iteration.

The second kind of adder we design in this project is called as Carry Skip Adder. This Adder is mainly used to decrease the worst case delay by decreasing the number of full adders presnt in the whole adder. This is achieved by constructing a carry skip network for each m bits of addition and the carry is propagated to a multiplexer where it selects between the carry generated by the propagate block and the full adder and takes a decision with respect to the propagate signal fed to it and then it goes to the next stage of addition.

The third kind of adder we designed in this project is called as Carry select adder. The main aim for designing this adder is the theory that the carry ripple adder's delay is mainly due to the propagation

of carry from the first stage to the last stage. As the carry we feed to any adder is binary which means it is either ‘0’ or ‘1’ so in a particular stage of addition we do the addition twice with ‘0’ as carry in and ‘1’ as a carry in and then compared the results with the actual carry in and thus the correct sum is decided for that stage of addition and later the carry-out from each adder are fed to a AND and OR gate to get the carry in for the next stage and in the last stage the final carry out is generated.

The fourth kind of adder we design in this project is called as Koggestone adder, this is a special kind of adder in which the addition is done without the regular adders but by using a series of propagate and generate blocks which does the trick for calculating the sum and carry out for the whole adder. The only thing with this kind of adders is that the circuit that has to be designed is a little bit complex and required utmost care in writing the circuit. The main advantage of this kind of adders is that they perform the addition in parallel and generate the results faster when compared to that of the regular adders.

ARCHITECTURAL EXPLORATION OF ADDERS

Any kind of adder has to follow the basic computation formula the basic computation formula for the adders is

$$Sum = \overline{A}.\overline{B}.C_{in} + \overline{A}.B.\overline{C_{in}} + A.\overline{B}.\overline{C_{in}}. + A.B.C_{in} = A \oplus B \oplus C_{in}$$

$$C_{out} = \overline{A}.B.C_{in} + A.\overline{B}.C_{in} + A.B.\overline{C_{in}}. + A.B.C_{in} = B.C_{in} + A.C_{in} + A.B$$

Inputs			Outputs	
A	B	Cin	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

The above written formula is written in verilog as shown below

```
module adder(s, co, a, b, ci);

  output s, co;
  input  a, b, ci;

  wire   o0, o1, o2;

  xor(s, a, b, ci);

  or(o0, a, b);
  or(o1, b, ci);
  or(o2, ci, a);
  and(co, o0, o1, o2);

endmodule // adder
```

The code for the full adder consists of a XOR gate which performs the addition and then the input bits are fed to a series of three or gates and later the output from or gates is fed to a and gate which calculates the carry out.

A. Carry Ripple Adder:

Carry Ripple Adder is formed using regular Full Adders, A full Adder performs addition of single bits and generate the outputs of each bit and a carry the carry is then fed to the next full adder which does the addition to the next bit and so on.

In this section, full adder we have created is used to design a 4 bit carry ripple adder and intern use that code as a module to construct a 8-bit carry ripple adder. The Verilog code for the 8 bit adder is shown below,

```
module adder8(s, co, a, b, ci);

  input [7:0] a;
  input [7:0] b;
  output[7:0] s;
  output co;
  input ci;
  wire c1, c2, c3, c4, c5, c6, c7;

  adder a0(s[0], c1, a[0], b[0], ci);
  adder a1(s[1], c2, a[1], b[1], c1);
  adder a2(s[2], c3, a[2], b[2], c2);
  adder a3(s[3], c4, a[3], b[3], c3);
  adder a4(s[4], c5, a[4], b[4], c4);
  adder a5(s[5], c6, a[5], b[5], c5);
```

```
adder a6(s[6], c7, a[6], b[6], c6);
adder a7(s[7], co, a[7], b[7], c7);

endmodule
```

The carry ripple adder designed in this project is shown in the figure below

This 8bit adders are used as a modules to make a 32 bit carry ripple adder and the code for it can be seen in the appendix section below.

B. Carry Skip Adder:

In carry skip adder there has to be a carry skip network for every m bits. So, in the 8-bit carry skip adder case it has to be designed using 2 4-bit adders, in that case we first generate a 4 bit propagate bits which are used as a selector bits for the multiplexer which drives the carry, the Verilog code for the propagate code is as follows

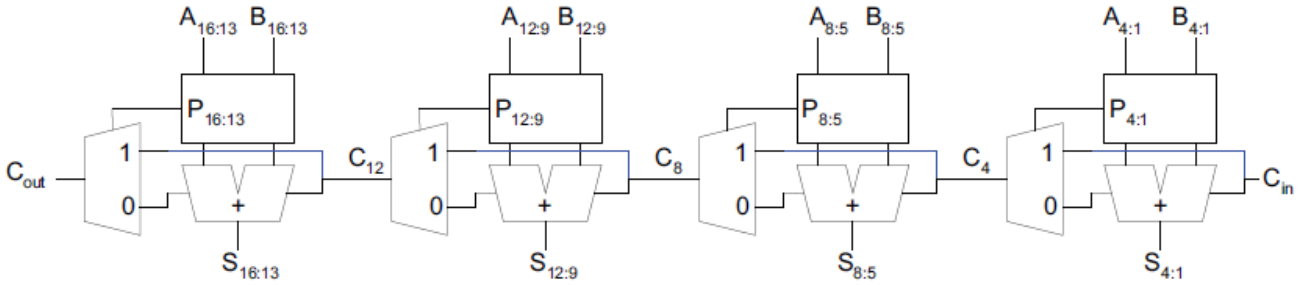
```
xor (p[0],a[0],b[0]);
xor (p[1],a[1],b[1]);
xor (p[2],a[2],b[2]);
xor (p[3],a[3],b[3]);
and (p3_0,p);
```

and the code used for the multiplexer is

```
module mux2(out,sel,a,b);
input a,b,sel;
output out;
tri out;
bufif1 (out,a,sel);
bufif0 (out,b,sel);
endmodule //mux2 end
```

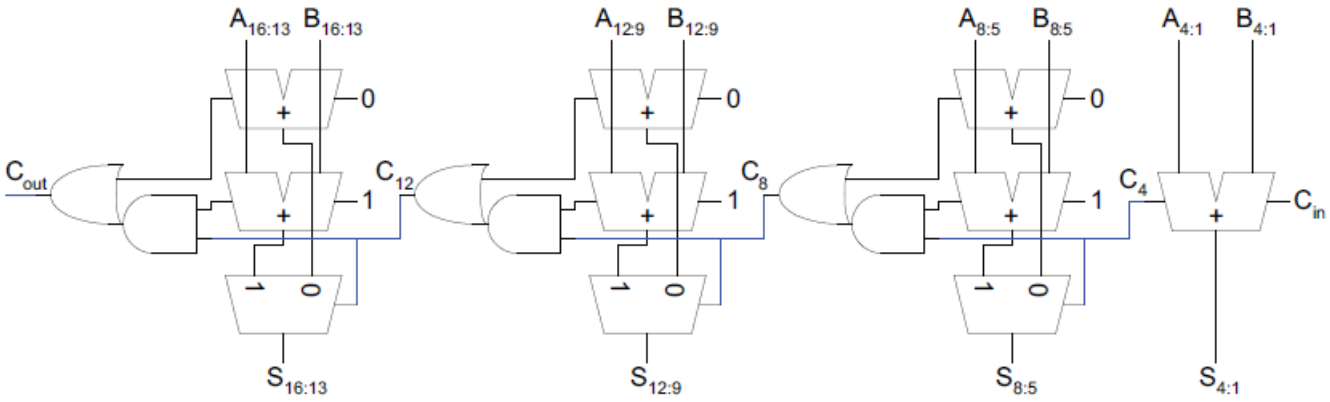
We use these modules to design a 8 bit carry skip adder and the code for the 8 and 32 bit carry skip adders are in the appendix section below.

The block diagram shown below represents the design of carry skip adder.



C. Carry Select Adder:

In a 8-bit carry select adder we do the first 4 bits addition as usually as we do in a carry ripple bit adder and the next four bits addition is done not in a regular way but there will be two adders in this stage where one adder does the addition with carry in as '0' and the other as carry in '1'. And then feed the sums both the adders are fed to a multiplexer whose select pin is the actual carry from the previous stage, thus the correct sum is selected as output. This extra set of hardware is made because the adder should not wait for the carry from the previous stage and by the time the actual carry arrives the additions have already been computed in all the next stages. The diagram below shows that the implementation of the carry select adder in this project.



The code for both the 8 and 32-bit carry select adder is shown in the appendix section below.

D. Prefix Adder – 'Koggestone' :

In this prefix tree adders such as koggestone there are propagate and generate blocks. The generate and propagate functions is as follows

$G_{i:j} = G_{i:k} + P_{i:k} * G_{k-1:j}$
$P_{i:j} = P_{i:k} * P_{k-1:j}$

And from the above functions we can easily calculate the Sum and carry which can be calculated as follows

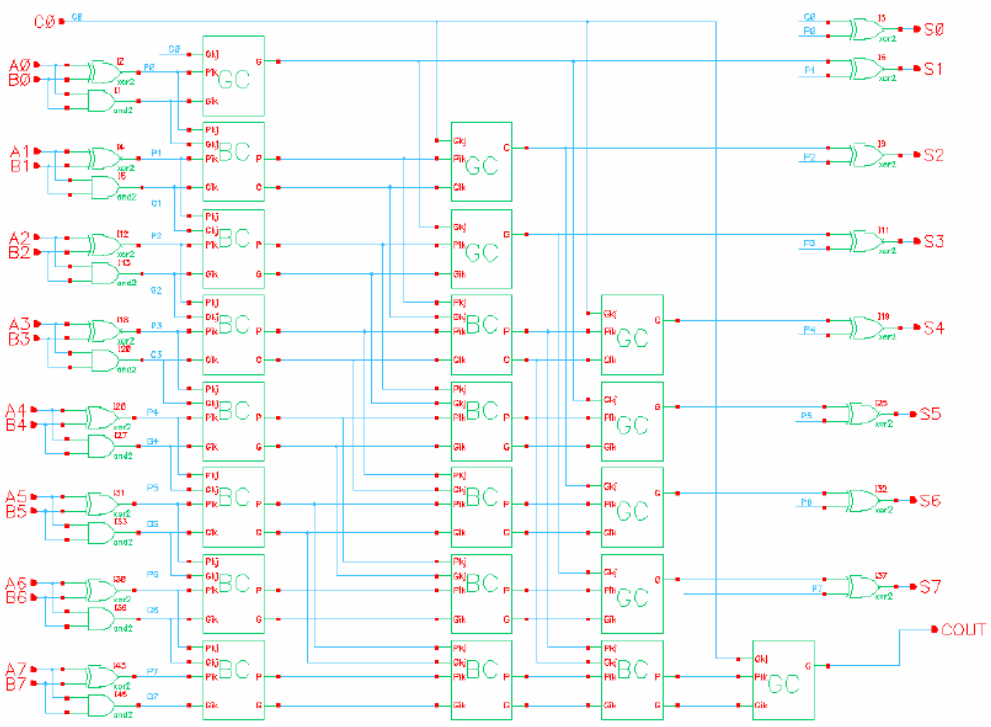
$S_i = P_i \text{ XOR } G_{i-1:0}$

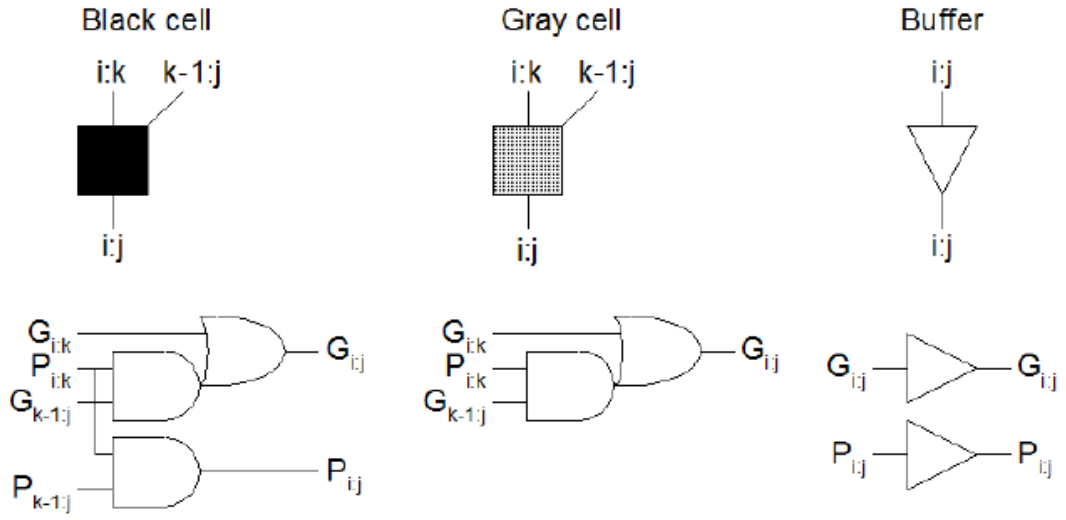
The base cases are as follows

$G_{i:i}=G_i=A_i*B_i$

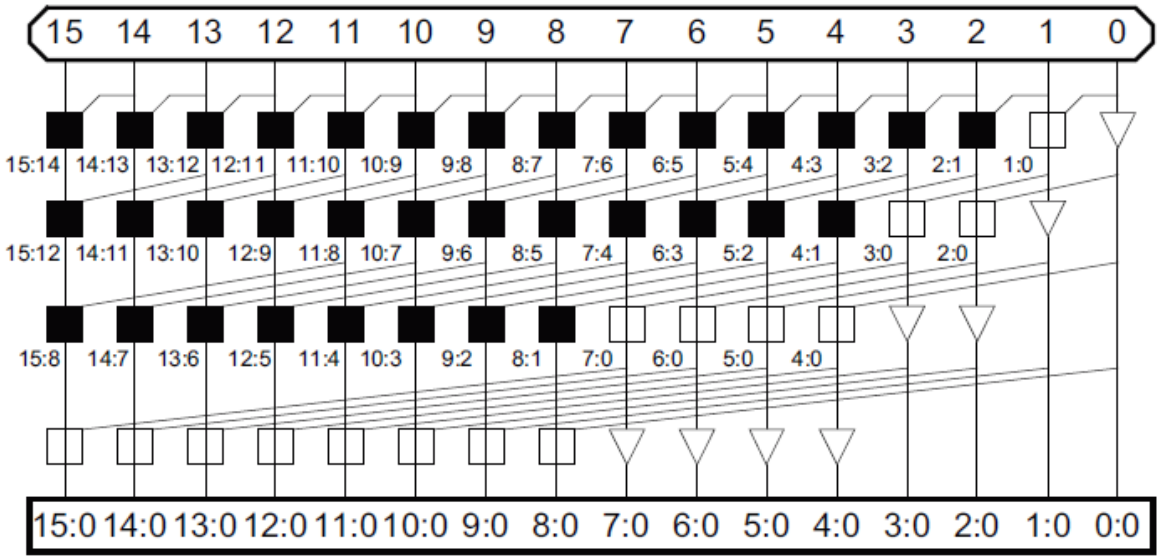
$P_{i:i}=P_i=A_i$ XOR B_i

The figure shown below is how the implementation is done in this project:





PG Diagram Notation for Prefix Tree Adders



16-bit Parallel Prefix Tree Adder - KoggeStone

MULTIPLIER DESIGN

In this array multiplier design the basic design consists of two basic modules we have defined it as square and rectangle modules, The code used in this array multiplier is as follows,

```
module square(so,co,sin,a,b,ci);
input a,b,ci,sin;
output co,so;
wire w1;
```

```
and an0(w1,a,b);
adder1 a0(so,co,w1,sin,ci);
endmodule

module rect(so,co,a,b,ci);
input a,b,ci;
output so,co;
adder1 a0(so,co,a,b,ci);
endmodule
```

and thus by following the diagram shown below the design is made, and the code for array multiplier is shown in the appendix section below.

FUNCTIONAL VALIDATION AND VERIFICATION

Functional validation and verification is the important step in any design where we check the correctness and effectiveness of the design made. The validation is done in 3 stages in the design flow.
The first validation is the pre synthesis simulation where we check the design with the testbench where we give inputs to the design made and check its results manually.
In the next stage synthesis is performed to transform the RTL code to gate level code. And then we conduct a place and route in the encounter tool and get the results.
At the end of each stage we make a conventional Verilog simulation and check that the results are same at the end of each stage.
To check the outputs correctly I have given same inputs for each case and have seen that the validation has been passed and the screenshots are placed in the appendix section showing that the design validations are correct.

As the last stage of verification I have done a formality check and the screenshots showing the formality check in the appendix section below.

SYNTHESIS RESULTS

A standard cell library is logic gate of particular fabrication process type having a group of different kinds of logic gates. We use this standard cell layout and thus donot need to create a layout manally we can automatically use the basic logic gates used in the library and make the design as needed, by doing this we can calculate the area required for the design and number of gates required and cell count and also on the delay level we can also calculate the required time arrival time and slack for a particular implementation design the tables presented in this section are the results after the design compile simulation and encounter simulation we can compare the results of a particular design before and after the place and router in a particular design and also the best possible adder design can be seen by considering all the design constrains and finding the optimal adder.

Results of 8-bit Carry Ripple Adder:

	Timing			Area		
	Required Time	Arrival Time	Slack Time	Area(μm)	Gate Count	Cell Count
Synthesis	3.90	0.75	3.15	138.912		48
Place and Route	3.90	1.173	2.727	131.4	46	48

Results of 32-bit Carry Ripple Adder:

	Timing			Area		
	Required Time	Arrival Time	Slack Time	Area(μm)	Gate Count	Cell Count
Synthesis	3.90	2.59	1.31	555.651		192
Place and Route	3.90	3.98	-0.08	525.6	186	192

Results of 8-bit Carry Skip Adder:

	Timing			Area		
	Required Time	Arrival Time	Slack Time	Area(μm)	Gate Count	Cell Count
Synthesis	3.90	0.92	2.98	168.94		58
Place and Route	3.90	1.261	2.639	158.6	56	58

Results of 32-bit Carry Skip Adder:

	Timing			Area		
	Required Time	Arrival Time	Slack Time	Area(μm)	Gate Count	Cell Count
Synthesis	3.9	3.28	0.62	675.791		232
Place and Route	3.9	4.81	-0.918	634.5	225	232

Results of 8-bit Carry Select Adder:

	Timing			Area		
	Required Time	Arrival Time	Slack Time	Area(μm)	Gate Count	Cell Count
Synthesis	3.9	0.62	3.28	233.711		76
Place and Route	3.9	0.921	2.979	214.9	76	76

Results of 32-bit Carry Select Adder:

	Timing			Area		
	Required Time	Arrival Time	Slack Time	Area(μm)	Gate Count	Cell Count
Synthesis	3.9	1.28	2.62	1096.754		354
Place and Route	3.9	1.965	1.935	1006.6	357	354

Results of 8-bit Koggestone Adder:

	Timing			Area		
	Required Time	Arrival Time	Slack Time	Area(μm)	Gate Count	Cell Count
Synthesis	3.9	0.5	3.4	262.33		100
Place and Route	3.9	0.873	3.027	245	87	100

Results of 32-bit Koggestone Adder:

	Timing			Area		
	Required Time	Arrival Time	Slack Time	Area(μm)	Gate Count	Cell Count
Synthesis	3.9	1.1	2.8	1049.354		400
Place and Route	3.9	2.009	1.891	979.9	348	400

Results of Array Multiplier:

	Timing			Area		
	Required Time	Arrival Time	Slack Time	Area(μm)	Gate Count	Cell Count
Synthesis	3.9	1.06	2.84	383.887		134
Place and Route	3.9	1.446	2.454	371.7	132	134

We can see that the slack time is getting better from the adders of the same class or in all the 4 8-bit adders the koggestone Adder is found to have the better slack time and similarly this is the case for the 32-bit adders as well, Thus we can say that the Koggestone Adders are the best kind of adders when delay is the main constraint that has be made better. If Area is the main concern the Carry ripple adder is the best adder.

As it has already been stated out that the trade-off between the delay and hardware.

CONCLUSION AND FUTURE WORK

Thus, we have designed all the four adders along with the bonus project and have come to a conclusion that if delay is considered the Koggestone Adder is the best adder amoung the 4 adders and if cost is the main concern the Carry Ripple Adder is the best Adder.

There may be possibility for the adder which has the optimal stage between the carry ripple adder and Koggestone-adder that has the better slack time and less adder.

APPENDIX

A. Verilog code for the Architectures:

i. Code for Carry Ripple Adder 8-bit:

```
// Adder

module adder(s, co, a, b, ci);

  output s, co;
  input  a, b, ci;

  wire   o0, o1, o2;

  xor(s, a, b, ci);

  or(o0, a, b);
  or(o1, b, ci);
  or(o2, ci, a);
  and(co, o0, o1, o2);

endmodule // adder


module adder8(s, co, a, b, ci);

  input [7:0] a;
  input [7:0] b;
  output[7:0] s;
  output co;
  input ci;
  wire c1, c2, c3, c4, c5, c6, c7;

  adder a0(s[0], c1, a[0], b[0], ci);
  adder a1(s[1], c2, a[1], b[1], c1);
  adder a2(s[2], c3, a[2], b[2], c2);
  adder a3(s[3], c4, a[3], b[3], c3);
  adder a4(s[4], c5, a[4], b[4], c4);
  adder a5(s[5], c6, a[5], b[5], c5);
  adder a6(s[6], c7, a[6], b[6], c6);
  adder a7(s[7], co, a[7], b[7], c7);
```

```
endmodule // adder8
```

ii. Test-Bench For Carry Ripple Adder 8-bit:

```
`timescale 1ns/10ps

module stimulus;

reg ci;
reg [7:0] a,b;

wire co;
wire [7:0] s;

adder8 adder1(s,co,a,b,ci);


initial begin
    $shm_open("shm.db",1); // Opens a waveform database
    $shm_probe("AS");    // Saves all signals to database
    #1000 $finish;
    $shm_close();   // Closes the waveform database
end

// Stimulate the Input Signals
initial begin
    a = 5;
    b = 10;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 37;
    b = 48;
    ci = 0;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 125;
    b = 110;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 63;
```

```
    b = 211;
    ci = 0;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 122;
    b = 11;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 245;
    b = 2;
    ci = 0;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 3;
    b = 90;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 100;
    b = 200;
    ci = 0;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 127;
    b = 127;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);
end

endmodule // stimulus
```

iii. Code for Carry Ripple Adder 32-bit:

```
//adder 32

module adder32(s, co, a, b, ci);
  input [31:0] a;
  input [31:0] b;
  output[31:0] s;
  output co;
  input ci;
  wire c1, c2, c3;
```

```
  adder8 a0(s[7:0], c1, a[7:0], b[7:0], ci);
  adder8 a1(s[15:8], c2, a[15:8], b[15:8], c1);
  adder8 a2(s[23:16], c3, a[23:16], b[23:16], c2);
  adder8 a3(s[31:24], co, a[31:24], b[31:24], c3);
endmodule // adder32 end
```

iv. Test-Bench For Carry Ripple Adder 32-bit:

```
`timescale 1ns/10ps

module stimulus;

reg ci;
reg [31:0] a,b;

wire co;
wire [31:0] s;

adder32 adder1(s,co,a,b,ci);


initial begin
    $shm_open("shm.db",1); // Opens a waveform database
    $shm_probe("AS");    // Saves all signals to database
    #1000 $finish;
    $shm_close();   // Closes the waveform database
end

// Stimulate the Input Signals
initial begin
    a = 5;
    b = 10;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 37;
    b = 48;
    ci = 0;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 125;
    b = 110;
    ci = 1;
```

```
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 63;
    b = 211;
    ci = 0;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 122;
    b = 11;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 245;
    b = 2;
    ci = 0;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 3;
    b = 90;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 100;
    b = 200;
    ci = 0;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 127;
    b = 127;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);
end

endmodule // stimulus
```

v. Code for Carry Skip Adder 8-bit:

```
//full adder 1bit
module adder1( s,co,a,b,ci);
output s,co;
input a,b,ci;
wire o0,o1,o2;
xor(s,a,b,ci);
or(o0,a,b);
```

```
or(o1,b,ci);
or(o2,ci,a);
and(co,o0,o1,o2);
endmodule //fulladder 1bit

//full adder 4bit

// mux2
module mux2(out,sel,a,b);
input a,b,sel;
output out;
tri out;
bufif1 (out,a,sel);
bufif0 (out,b,sel);
endmodule //mux2 end


// carryskip adder 4bit
module carryskip4(s,co,a,b,ci);
wire [3:0] p;
output [3:0] s;
output co;
input [3:0] a;
input [3:0] b;
input ci;
wire p3_0,w1,w2,w3,w4;
xor (p[0],a[0],b[0]);
xor (p[1],a[1],b[1]);
xor (p[2],a[2],b[2]);
xor (p[3],a[3],b[3]);
and (p3_0,p);
adder1 a0(s[0],w1,a[0],b[0],ci);
adder1 a1(s[1],w2,a[1],b[1],w1);
adder1 a2(s[2],w3,a[2],b[2],w2);
adder1 a3(s[3],w4,a[3],b[3],w3);
mux2 m0(co,p3_0,ci,w4);

endmodule
//carryskip adder 4bit end

module carryskip8 (s,co,a,b,ci);
output [7:0] s;
output co;
```

```
input [7:0] a;
input [7:0] b;
input ci;
wire c4;

carryskip4 cs0(s[3:0],c4,a[3:0],b[3:0],ci);
carryskip4 cs1(s[7:4],co,a[7:4],b[7:4],c4);
endmodule
```

vi. Test-Bench for Carry Skip Adder 8-bit:

```
`timescale 1ns/10ps

module stimulus;

reg ci;
reg [7:0] a,b;

wire co;
wire [7:0] s;

carryskip8 adder1(s,co,a,b,ci);


initial begin
    $shm_open("shm.db",1); // Opens a waveform database
    $shm_probe("AS");    // Saves all signals to database
    #1000 $finish;
    $shm_close();   // Closes the waveform database
end

// Stimulate the Input Signals
initial begin
    a = 5;
    b = 10;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 37;
    b = 48;
    ci = 0;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);
```

```
    a = 125;
    b = 110;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 63;
    b = 211;
    ci = 0;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 122;
    b = 11;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 245;
    b = 2;
    ci = 0;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 3;
    b = 90;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 100;
    b = 200;
    ci = 0;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 127;
    b = 127;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);
end

endmodule // stimulus
```

vii. Code for Carry Skip Adder 32-bit:

```
module carryskip32 (s,co,a,b,ci);
output [31:0] s;
output co;
```

```
input [31:0] a;
input [31:0] b;
input ci;
wire c8,c16,c24;

carryskip8 cs0(s[7:0],c8,a[7:0],b[7:0],ci);
carryskip8 cs1(s[15:8],c16,a[15:8],b[15:8],c8);
carryskip8 cs3(s[23:16],c24,a[23:16],b[23:16],c16);
carryskip8 cs4(s[31:24],co,a[31:24],b[31:24],c24);

endmodule
```

viii. Test-Bench for Carry Skip Adder 32-bit:

```
`timescale 1ns/10ps

module stimulus;

reg ci;
reg [31:0] a,b;

wire co;
wire [31:0] s;

carryskip32 adder1(s,co,a,b,ci);


initial begin
    $shm_open("shm.db",1); // Opens a waveform database
    $shm_probe("AS");    // Saves all signals to database
    #1000 $finish;
    $shm_close();   // Closes the waveform database
end

// Stimulate the Input Signals
initial begin
    a = 5;
    b = 10;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 37;
    b = 48;
    ci = 0;
```

```
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 125;
    b = 110;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 63;
    b = 211;
    ci = 0;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 122;
    b = 11;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 245;
    b = 2;
    ci = 0;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 3;
    b = 90;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 100;
    b = 200;
    ci = 0;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 127;
    b = 127;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);
end

endmodule // stimulus
```

ix. Code for Carry Select Adder 8-bit:

```
// carry select adder 8bit
```

```
// adder 1bit
module adder1(s, co, a, b, ci);

output s, co;
input  a, b, ci;
wire   o0, o1, o2;

xor(s, a, b, ci);
or(o0, a, b);
or(o1, b, ci);
or(o2, ci, a);
and(co, o0, o1, o2);

endmodule // adder

// adder 2 bit
module adder2(s,co,a,b,ci);

output[1:0] s;
output co;
input [1:0] a,b;
input ci;
wire w1;

adder1 a0(s[0],w1,a[0],b[0],ci);
adder1 a1(s[1],co,a[1],b[1],w1);

endmodule

//adder 2bit end

// multiplexer 2
module mux2(out,sel,a,b);
input a,b,sel;
output out;
tri out;
bufif1 (out,a,sel);
bufif0 (out,b,sel);
endmodule
//mux2 end

// carry ripple adder 4bit
module carryselect4(s,co,a,b,ci,c_0,c_1);
```

```
//reg c_0=0;
//reg c_1=1;
output [3:0] s;
output co;
input [3:0] a,b;
input ci,c_0,c_1;
wire c2,w1,w2,w3;
wire [1:0] s_0,s_1;

adder2 a0(s[1:0],c2,a[1:0],b[1:0],ci);
adder2 a1_0(s_0,w1,a[3:2],b[3:2],c_0);
adder2 a1_1(s_1,w2,a[3:2],b[3:2],c_1);

mux2 m0(s[2],c2,s_1[0],s_0[0]);
mux2 m1(s[3],c2,s_1[1],s_0[1]);

and(w3,w2,c2);
or(co,w1,w3);
endmodule

module carryselect8(s,co,a,b,ci,c_0,c_1);
output [7:0] s;
output co;
input [7:0] a,b;
input ci,c_0,c_1;
wire  c4;
carryselect4 cs0(s[3:0],c4,a[3:0],b[3:0],ci,c_0,c_1);
carryselect4 cs1(s[7:4],co,a[7:4],b[7:4],c4,c_0,c_1);
endmodule
```

x. Test-Bench for Carry Select Adder 8-bit:

```
`timescale 1ns/10ps

module stimulus;
reg c_0,c_1;
reg ci;
reg [7:0] a,b;

wire co;
wire [7:0] s;

carryselect8 adder1(s,co,a,b,ci,c_0,c_1);
```

```
initial begin
    $shm_open("shm.db",1); // Opens a waveform database
    $shm_probe("AS");    // Saves all signals to database
    #1000 $finish;
    $shm_close();   // Closes the waveform database
end

// Stimulate the Input Signals
initial begin
            a = 5;
    b = 10;
    ci = 1;
    c_0=0;
    c_1=1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 37;
    b = 48;
    ci = 0;
    c_0=0;
    c_1=1;

    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 125;
    b = 110;
    ci = 1;
    c_0=0;
    c_1=1;

    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 63;
    b = 211;
    ci = 0;
    c_0=0;
    c_1=1;

    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 122;
    b = 11;
```

```
        ci = 1;
    c_0=0;
        c_1=1;

        #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

        a = 245;
        b = 2;
        ci = 0;
        c_0=0;
        c_1=1;

        #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

        a = 3;
        b = 90;
        ci = 1;
        c_0=0;
        c_1=1;

        #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

        a = 100;
        b = 200;
        ci = 0;
        c_0=0;
        c_1=1;

        #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

        a = 127;
        b = 127;
        ci = 1;
        c_0=0;
        c_1=1;

        #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);
    end

    endmodule // stimulus
```

xi. Code for Carry Select Adder 32-bit:

```
module adder1(s, co, a, b, ci);

output s, co;
input  a, b, ci;
wire   o0, o1, o2;

xor(s, a, b, ci);
or(o0, a, b);
or(o1, b, ci);
or(o2, ci, a);
and(co, o0, o1, o2);

endmodule // adder


// multiplexer 2
module mux2(out,sel,a,b);
input a,b,sel;
output out;
tri out;
bufif1 (out,a,sel);
bufif0 (out,b,sel);
endmodule
//mux2 end

module carryripple8(s,co,a,b,ci);
input [7:0] a,b;
input ci;
output [7:0] s;
output co;
wire c1,c2,c3,c4,c5,c6,c7;

adder1 a1(s[0],c1,a[0],b[0],ci);
adder1 a2(s[1],c2,a[1],b[1],c1);
adder1 a3(s[2],c3,a[2],b[2],c2);
adder1 a4(s[3],c4,a[3],b[3],c3);
adder1 a5(s[4],c5,a[4],b[4],c4);
adder1 a6(s[5],c6,a[5],b[5],c5);
adder1 a7(s[6],c7,a[6],b[6],c6);
adder1 a8(s[7],co,a[7],b[7],c7);

endmodule
```

```
module carryselect32(s,co,a,b,ci,c_0,c_1);

output [31:0] s;
output co;
input [31:0] a,b;
input ci,c_0,c_1;
wire [23:0] s_0,s_1;
wire c16_0,c16_1,c24_0,c24_1,c32_0,c32_1;
wire c8,c16,c24;
wire w16,w24,w32;
carryripple8 cr0(s[7:0],c8,a[7:0],b[7:0],ci);

carryripple8 cr1_0(s_0[7:0],c16_0,a[15:8],b[15:8],c_0);
carryripple8 cr1_1(s_1[7:0],c16_1,a[15:8],b[15:8],c_1);

mux2 m1_8(s[8],c8,s_1[0],s_0[0]);
mux2 m1_9(s[9],c8,s_1[1],s_0[1]);
mux2 m1_10(s[10],c8,s_1[2],s_0[2]);
mux2 m1_11(s[11],c8,s_1[3],s_0[3]);
mux2 m1_12(s[12],c8,s_1[4],s_0[4]);
mux2 m1_13(s[13],c8,s_1[5],s_0[5]);
mux2 m1_14(s[14],c8,s_1[6],s_0[6]);
mux2 m1_15(s[15],c8,s_1[7],s_0[7]);
and an1(w16,c16_1,c8);
or or1(c16,w16,c16_0);

carryripple8 cr2_0(s_0[15:8],c24_0,a[23:16],b[23:16],c_0);
carryripple8 cr2_1(s_1[15:8],c24_1,a[23:16],b[23:16],c_1);

mux2 m1_16(s[16],c16,s_1[8],s_0[8]);
mux2 m1_17(s[17],c16,s_1[9],s_0[9]);
mux2 m1_18(s[18],c16,s_1[10],s_0[10]);
mux2 m1_19(s[19],c16,s_1[11],s_0[11]);
mux2 m1_20(s[20],c16,s_1[12],s_0[12]);
mux2 m1_21(s[21],c16,s_1[13],s_0[13]);
mux2 m1_22(s[22],c16,s_1[14],s_0[14]);
mux2 m1_23(s[23],c16,s_1[15],s_0[15]);

and an2(w24,c24_1,c16);
or or2(c24,w24,c24_0);

carryripple8 cr3_0(s_0[23:16],c32_0,a[31:24],b[31:24],c_0);
carryripple8 cr3_1(s_1[23:16],c32_1,a[31:24],b[31:24],c_1);
```

```
mux2 m1_24(s[24],c24,s_1[16],s_0[16]);
mux2 m1_25(s[25],c24,s_1[17],s_0[17]);
mux2 m1_26(s[26],c24,s_1[18],s_0[18]);
mux2 m1_27(s[27],c24,s_1[19],s_0[19]);
mux2 m1_28(s[28],c24,s_1[20],s_0[20]);
mux2 m1_29(s[29],c24,s_1[21],s_0[21]);
mux2 m1_30(s[30],c24,s_1[22],s_0[22]);
mux2 m1_31(s[31],c24,s_1[23],s_0[23]);

and an3(w32,c32_1,c24);
or or3(co,w32,c32_0);
endmodule
```

xii. Test-Bench for Carry Select Adder 32-bit:

```
`timescale 1ns/10ps

module stimulus;

reg ci,c_0,c_1;
reg [31:0] a,b;


wire co;
wire [31:0] s;

carryselect32 adder1(s,co,a,b,ci,c_0,c_1);


initial begin
    $shm_open("shm.db",1); // Opens a waveform database
    $shm_probe("AS");    // Saves all signals to database
    #1000 $finish;
    $shm_close();   // Closes the waveform database
end

// Stimulate the Input Signals
initial begin
    a = 5;
    b = 10;
```

```
    ci = 1;
    c_0=0;
    c_1=1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 37;
    b = 48;
    ci = 0;
    c_0=0;
    c_1=1;

    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 125;
    b = 110;
    ci = 1;
    c_0=0;
    c_1=1;

    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 63;
    b = 211;
    ci = 0;
    c_0=0;
    c_1=1;

    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 122;
    b = 11;
    ci = 1;
c_0=0;
    c_1=1;

    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 245;
    b = 2;
    ci = 0;
    c_0=0;
    c_1=1;
```

```
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 3;
    b = 90;
    ci = 1;
    c_0=0;
    c_1=1;

    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 100;
    b = 200;
    ci = 0;
    c_0=0;
    c_1=1;

    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 127;
    b = 127;
    ci = 1;
    c_0=0;
    c_1=1;

    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);
end

endmodule // stimulus
```

xiii. Code for Koggestone Adder 8-bit:

```
//generate and propagete block
module gandp(g,p,a,b);
input a,b;
output g,p;
xor(p,a,b);
and(g,a,b);
endmodule
// end of generate and propagate

//declaring a universal grey cell
module greycell(g,g_kj,p_ik,g_ik);
input g_kj,p_ik,g_ik;
output g;
```

```
wire w1;

and(w1,g_kj,p_ik);
or(g,w1,g_ik);
endmodule
//end of universal grey cell


// declaring a universal black cell
module blackcell(g,p,p_kj,g_kj,p_ik,g_ik);
input p_kj,g_kj,p_ik,g_ik;
output g,p;
wire w1;

and(w1,g_kj,p_ik);
or(g,w1,g_ik);
and(p,p_ik,p_kj);
endmodule
//end of universal black cell

module koggstone8(s,co,a,b,ci);
input [7:0] a,b;
input ci;
output co;
output [7:0] s;

wire [7:0] g,p;
wire gc_s1_g;
wire [1:0] gc_s2_g;
wire [3:0] gc_s3_g;
wire [6:0] bc_s1_g,bc_s1_p;
wire [4:0] bc_s2_g,bc_s2_p;
wire bc_s3_g,bc_s3_p;


// generating g and p for input bits
gandp gp0(g[0],p[0],a[0],b[0]);
gandp gp1(g[1],p[1],a[1],b[1]);
gandp gp2(g[2],p[2],a[2],b[2]);
gandp gp3(g[3],p[3],a[3],b[3]);
gandp gp4(g[4],p[4],a[4],b[4]);
gandp gp5(g[5],p[5],a[5],b[5]);
gandp gp6(g[6],p[6],a[6],b[6]);
```

```
gandp gp7(g[7],p[7],a[7],b[7]);


// using grey and black cells and grey cells to generate co

//stage 1
greycell gc0(gc_s1_g,ci,p[0],g[0]);
blackcell bc0(bc_s1_g[0],bc_s1_p[0],p[0],g[0],p[1],g[1]);
blackcell bc1(bc_s1_g[1],bc_s1_p[1],p[1],g[1],p[2],g[2]);
blackcell bc2(bc_s1_g[2],bc_s1_p[2],p[2],g[2],p[3],g[3]);
blackcell bc3(bc_s1_g[3],bc_s1_p[3],p[3],g[3],p[4],g[4]);
blackcell bc4(bc_s1_g[4],bc_s1_p[4],p[4],g[4],p[5],g[5]);
blackcell bc5(bc_s1_g[5],bc_s1_p[5],p[5],g[5],p[6],g[6]);
blackcell bc6(bc_s1_g[6],bc_s1_p[6],p[6],g[6],p[7],g[7]);

// stage 2

greycell gc1(gc_s2_g[0],ci,bc_s1_p[0],bc_s1_g[0]);
greycell gc2(gc_s2_g[1],gc_s1_g,bc_s1_p[1],bc_s1_g[1]);
blackcell
bc7(bc_s2_g[0],bc_s2_p[0],bc_s1_p[0],bc_s1_g[0],bc_s1_p[2],bc_s1_g[2]);
blackcell
bc8(bc_s2_g[1],bc_s2_p[1],bc_s1_p[1],bc_s1_g[1],bc_s1_p[3],bc_s1_g[3]);
blackcell
bc9(bc_s2_g[2],bc_s2_p[2],bc_s1_p[2],bc_s1_g[2],bc_s1_p[4],bc_s1_g[4]);
blackcell
bc10(bc_s2_g[3],bc_s2_p[3],bc_s1_p[3],bc_s1_g[3],bc_s1_p[5],bc_s1_g[5]);
blackcell
bc11(bc_s2_g[4],bc_s2_p[4],bc_s1_p[4],bc_s1_g[4],bc_s1_p[6],bc_s1_g[6]);

//stage 3
greycell gc3(gc_s3_g[0],ci,bc_s2_p[0],bc_s2_g[0]);
greycell gc4(gc_s3_g[1],gc_s1_g,bc_s2_p[1],bc_s2_g[1]);
greycell gc5(gc_s3_g[2],gc_s2_g[0],bc_s2_p[2],bc_s2_g[2]);
greycell gc6(gc_s3_g[3],gc_s2_g[1],bc_s2_p[3],bc_s2_g[3]);
blackcell
bc12(bc_s3_g,bc_s3_p,bc_s2_p[0],bc_s2_g[0],bc_s2_p[4],bc_s2_g[4]);


// carry out
greycell gc7(co,ci,bc_s3_p,bc_s3_g);

// sum
```

```
xor x0(s[0],ci,p[0]);
xor x1(s[1],gc_s1_g,p[1]);
xor x2(s[2],gc_s2_g[0],p[2]);
xor x3(s[3],gc_s2_g[1],p[3]);
xor x4(s[4],gc_s3_g[0],p[4]);
xor x5(s[5],gc_s3_g[1],p[5]);
xor x6(s[6],gc_s3_g[2],p[6]);
xor x7(s[7],gc_s3_g[3],p[7]);

endmodule
```

xiv. Test-Bench for Koggestone 8-bit:

```
`timescale 1ns/10ps

module stimulus;

reg ci;
reg [7:0] a,b;

wire co;
wire [7:0] s;

koggstone8 adder1(s,co,a,b,ci);


initial begin
    $shm_open("shm.db",1); // Opens a waveform database
    $shm_probe("AS");    // Saves all signals to database
    #1000 $finish;
    $shm_close();   // Closes the waveform database
end

// Stimulate the Input Signals
initial begin
    a = 5;
    b = 10;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 37;
    b = 48;
    ci = 0;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);
```

```
    a = 125;
    b = 110;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 63;
    b = 211;
    ci = 0;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 122;
    b = 11;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 245;
    b = 2;
    ci = 0;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 3;
    b = 90;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 100;
    b = 200;
    ci = 0;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 127;
    b = 127;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);
end

endmodule // stimulus
```

xv. Code for Koggestone 32-bit:

```
//generate and propagete block
module gandp(g,p,a,b);
input a,b;
```

```
output g,p;
xor(p,a,b);
and(g,a,b);
endmodule
// end of generate and propagate

//declaring a universal grey cell
module greycell(g,g_kj,p_ik,g_ik);
input g_kj,p_ik,g_ik;
output g;
wire w1;

and(w1,g_kj,p_ik);
or(g,w1,g_ik);
endmodule
//end of universal grey cell


// declaring a universal black cell
module blackcell(g,p,p_kj,g_kj,p_ik,g_ik);
input p_kj,g_kj,p_ik,g_ik;
output g,p;
wire w1;

and(w1,g_kj,p_ik);
or(g,w1,g_ik);
and(p,p_ik,p_kj);
endmodule
//end of universal black cell

module koggstone8(s,co,a,b,ci);
input [7:0] a,b;
input ci;
output co;
output [7:0] s;

wire [7:0] g,p;
wire gc_s1_g;
wire [1:0] gc_s2_g;
wire [3:0] gc_s3_g;
wire [6:0] bc_s1_g,bc_s1_p;
wire [4:0] bc_s2_g,bc_s2_p;
wire bc_s3_g,bc_s3_p;
```

```
// generating g and p for input bits
gandp gp0(g[0],p[0],a[0],b[0]);
gandp gp1(g[1],p[1],a[1],b[1]);
gandp gp2(g[2],p[2],a[2],b[2]);
gandp gp3(g[3],p[3],a[3],b[3]);
gandp gp4(g[4],p[4],a[4],b[4]);
gandp gp5(g[5],p[5],a[5],b[5]);
gandp gp6(g[6],p[6],a[6],b[6]);
gandp gp7(g[7],p[7],a[7],b[7]);


// using grey and black cells and grey cells to generate co

//stage 1
greycell gc0(gc_s1_g,ci,p[0],g[0]);
blackcell bc0(bc_s1_g[0],bc_s1_p[0],p[0],g[0],p[1],g[1]);
blackcell bc1(bc_s1_g[1],bc_s1_p[1],p[1],g[1],p[2],g[2]);
blackcell bc2(bc_s1_g[2],bc_s1_p[2],p[2],g[2],p[3],g[3]);
blackcell bc3(bc_s1_g[3],bc_s1_p[3],p[3],g[3],p[4],g[4]);
blackcell bc4(bc_s1_g[4],bc_s1_p[4],p[4],g[4],p[5],g[5]);
blackcell bc5(bc_s1_g[5],bc_s1_p[5],p[5],g[5],p[6],g[6]);
blackcell bc6(bc_s1_g[6],bc_s1_p[6],p[6],g[6],p[7],g[7]);

// stage 2

greycell gc1(gc_s2_g[0],ci,bc_s1_p[0],bc_s1_g[0]);
greycell gc2(gc_s2_g[1],gc_s1_g,bc_s1_p[1],bc_s1_g[1]);
blackcell
bc7(bc_s2_g[0],bc_s2_p[0],bc_s1_p[0],bc_s1_g[0],bc_s1_p[2],bc_s1_g[2]);
blackcell
bc8(bc_s2_g[1],bc_s2_p[1],bc_s1_p[1],bc_s1_g[1],bc_s1_p[3],bc_s1_g[3]);
blackcell
bc9(bc_s2_g[2],bc_s2_p[2],bc_s1_p[2],bc_s1_g[2],bc_s1_p[4],bc_s1_g[4]);
blackcell
bc10(bc_s2_g[3],bc_s2_p[3],bc_s1_p[3],bc_s1_g[3],bc_s1_p[5],bc_s1_g[5]);
blackcell
bc11(bc_s2_g[4],bc_s2_p[4],bc_s1_p[4],bc_s1_g[4],bc_s1_p[6],bc_s1_g[6]);

//stage 3
greycell gc3(gc_s3_g[0],ci,bc_s2_p[0],bc_s2_g[0]);
greycell gc4(gc_s3_g[1],gc_s1_g,bc_s2_p[1],bc_s2_g[1]);
```

```
greycell gc5(gc_s3_g[2],gc_s2_g[0],bc_s2_p[2],bc_s2_g[2]);
greycell gc6(gc_s3_g[3],gc_s2_g[1],bc_s2_p[3],bc_s2_g[3]);
blackcell
bc12(bc_s3_g,bc_s3_p,bc_s2_p[0],bc_s2_g[0],bc_s2_p[4],bc_s2_g[4]);


// carry out
greycell gc7(co,ci,bc_s3_p,bc_s3_g);

// sum
xor x0(s[0],ci,p[0]);
xor x1(s[1],gc_s1_g,p[1]);
xor x2(s[2],gc_s2_g[0],p[2]);
xor x3(s[3],gc_s2_g[1],p[3]);
xor x4(s[4],gc_s3_g[0],p[4]);
xor x5(s[5],gc_s3_g[1],p[5]);
xor x6(s[6],gc_s3_g[2],p[6]);
xor x7(s[7],gc_s3_g[3],p[7]);

endmodule

//koggstone adder 32bit adder with 4 8bit koggstone adder modules
module koggstone32(s,co,a,b,ci);
input [31:0] a,b;
input ci;
output co;
output [31:0] s;

koggstone8 ks0(s[7:0],c8,a[7:0],b[7:0],ci);
koggstone8 ks1(s[15:8],c16,a[15:8],b[15:8],c8);
koggstone8 ks2(s[23:16],c24,a[23:16],b[23:16],c16);
koggstone8 ks3(s[31:24],co,a[31:24],b[31:24],c24);
endmodule
```

xvi. Test-Bench for Koggestone 32-bit:

```
`timescale 1ns/10ps

module stimulus;

reg ci;
reg [31:0] a,b;
```

```
wire co;
wire [31:0] s;

koggstone32 adder1(s,co,a,b,ci);


initial begin
    $shm_open("shm.db",1); // Opens a waveform database
    $shm_probe("AS");    // Saves all signals to database
    #1000 $finish;
    $shm_close();   // Closes the waveform database
end

// Stimulate the Input Signals
initial begin
    a = 5;
    b = 10;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 37;
    b = 48;
    ci = 0;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 125;
    b = 110;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 63;
    b = 211;
    ci = 0;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 122;
    b = 11;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 245;
    b = 2;
    ci = 0;
```

```
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 3;
    b = 90;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 100;
    b = 200;
    ci = 0;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);

    a = 127;
    b = 127;
    ci = 1;
    #100 $display("At Time: %d  Sum=%d Carry=%d",$time,s,co);
end

endmodule // stimulus
```

xvii. Code for Array Multiplier:

```
// adder 1 bit
module adder1(s, co, a, b, ci);

output s, co;
input  a, b, ci;
wire   o0, o1, o2;

xor(s, a, b, ci);
or(o0, a, b);
or(o1, b, ci);
or(o2, ci, a);
and(co, o0, o1, o2);

endmodule // adder
// 1st block in a array multiplier
module square(so,co,sin,a,b,ci);

input a,b,ci,sin;
output co,so;

wire w1;
```

```
and an0(w1,a,b);
adder1 a0(so,co,w1,sin,ci);

endmodule

module rect(so,co,a,b,ci);
input a,b,ci;
output so,co;



adder1 a0(so,co,a,b,ci);
endmodule

module arraymultiplier(p,co,a,b,si,ci);

input [3:0] a,b;
input ci,si;
output [7:0] p;
output co;
wire [11:0] s_interm;
wire c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,c13,c14,c15,c16,c17,c118,c19;

//1st line
square sq1(p[0],c1,si,a[0],b[0],ci);
square sq2(s_interm[0],c2,si,a[1],b[0],ci);
square sq3(s_interm[1],c3,si,a[2],b[0],ci);
square sq4(s_interm[2],c4,si,a[3],b[0],ci);
//2nd line
square sq5(p[1],c5,s_interm[0],a[0],b[1],c1);
square sq6(s_interm[3],c6,s_interm[1],a[1],b[1],c2);
square sq7(s_interm[4],c7,s_interm[2],a[2],b[1],c3);
square sq8(s_interm[5],c8,si,a[3],b[1],c4);
//3rd line
square sq9(p[2],c9,s_interm[3],a[0],b[2],c5);
square sq10(s_interm[6],c10,s_interm[4],a[1],b[2],c6);
square sq11(s_interm[7],c11,s_interm[5],a[2],b[2],c7);
square sq12(s_interm[8],c12,si,a[3],b[2],c8);
//4th line
square sq13(p[3],c13,s_interm[6],a[0],b[3],c9);
square sq14(s_interm[9],c14,s_interm[7],a[1],b[3],c10);
square sq15(s_interm[10],c15,s_interm[8],a[2],b[3],c11);
square sq16(s_interm[11],c16,si,a[3],b[3],c12);
```

```
//5th line
rect rc1(p[4],c17,s_interm[9],c13,ci);
rect rc2(p[5],c18,s_interm[10],c14,c17);
rect rc3(p[6],c19,s_interm[11],c15,c18);
rect rc4(p[7],co,si,c16,c19);
endmodule
```

xviii. Test-Bench for Array Multiplier:

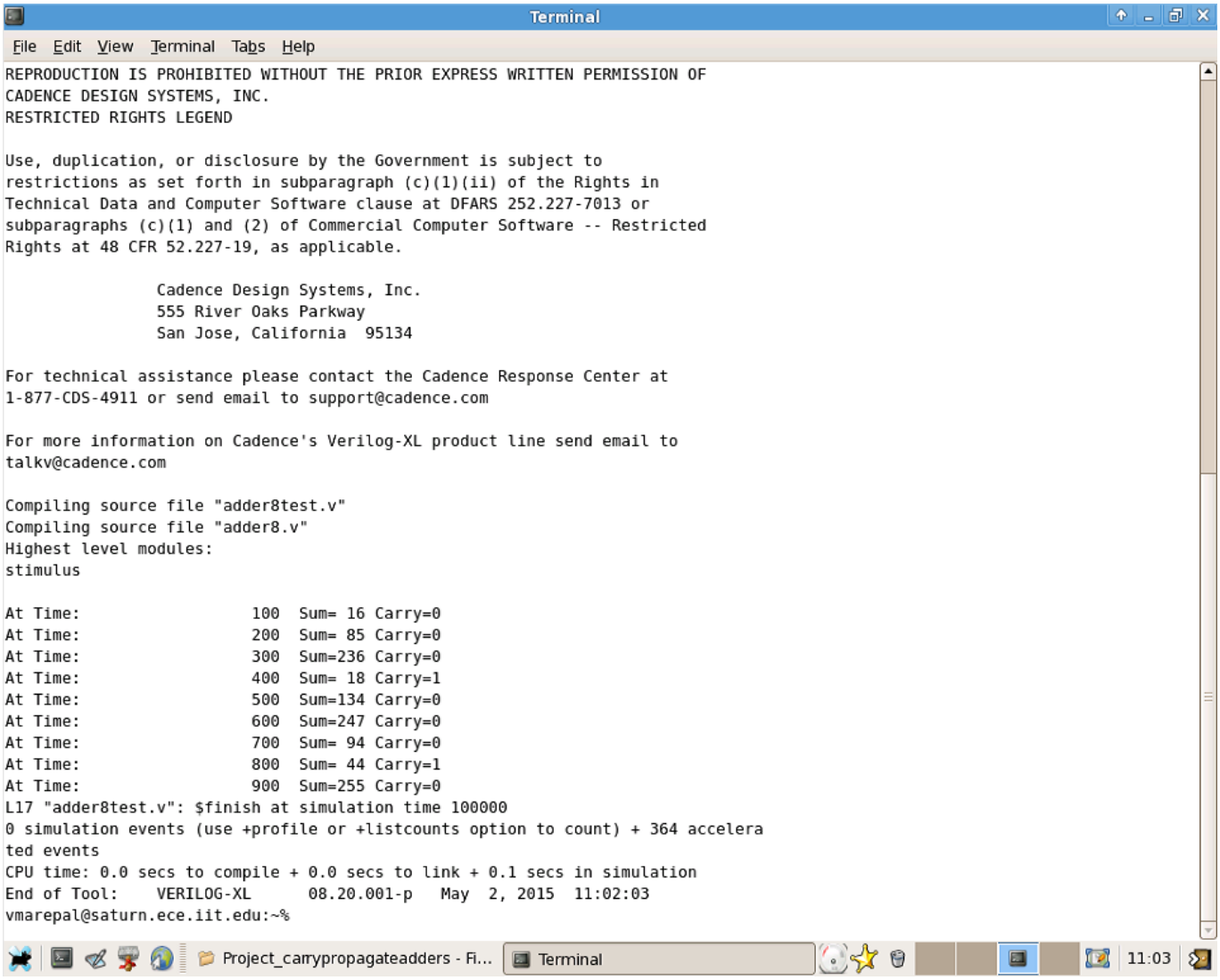
```
`timescale 1ns/10ps

module stimulus;

reg ci;
reg si;
reg [3:0] a,b;

wire co;
wire [7:0] p;

arraymultiplier mul1(p,co,a,b,si,ci);


initial begin
    $shm_open("shm.db",1); // Opens a waveform database
    $shm_probe("AS");    // Saves all signals to database
    #1000 $finish;
    $shm_close();   // Closes the waveform database
end

// Stimulate the Input Signals
initial begin
    a = 5;
    b = 10;
    ci =0;
    si =0;
    #100 $display("At Time: %d  Product=%d Carry=%d",$time,p,co);

    a = 37;
    b = 48;
    ci = 0;
    si =0;
    #100 $display("At Time: %d  Product=%d Carry=%d",$time,p,co);
```

```
    a = 125;
    b = 110;
    ci = 1;
    si =0;
    #100 $display("At Time: %d  Product=%d Carry=%d",$time,p,co);

    a = 63;
    b = 211;
    ci = 0;
    si =0;
    #100 $display("At Time: %d  Product=%d Carry=%d",$time,p,co);

    a = 122;
    b = 11;
    ci = 1;
    si =0;
    #100 $display("At Time: %d  Product=%d Carry=%d",$time,p,co);

    a = 245;
    b = 2;
    ci = 0;
    si =0;
    #100 $display("At Time: %d  Product=%d Carry=%d",$time,p,co);

    a = 3;
    b = 90;
    ci = 1;
    si =0;
    #100 $display("At Time: %d  Product=%d Carry=%d",$time,p,co);

    a = 100;
    b = 200;
    ci = 0;
    si =0;
    #100 $display("At Time: %d  Product=%d Carry=%d",$time,p,co);

    a = 127;
    b = 127;
    ci = 1;
    si =0;
    #100 $display("At Time: %d  Product=%d Carry=%d",$time,p,co);
end
```

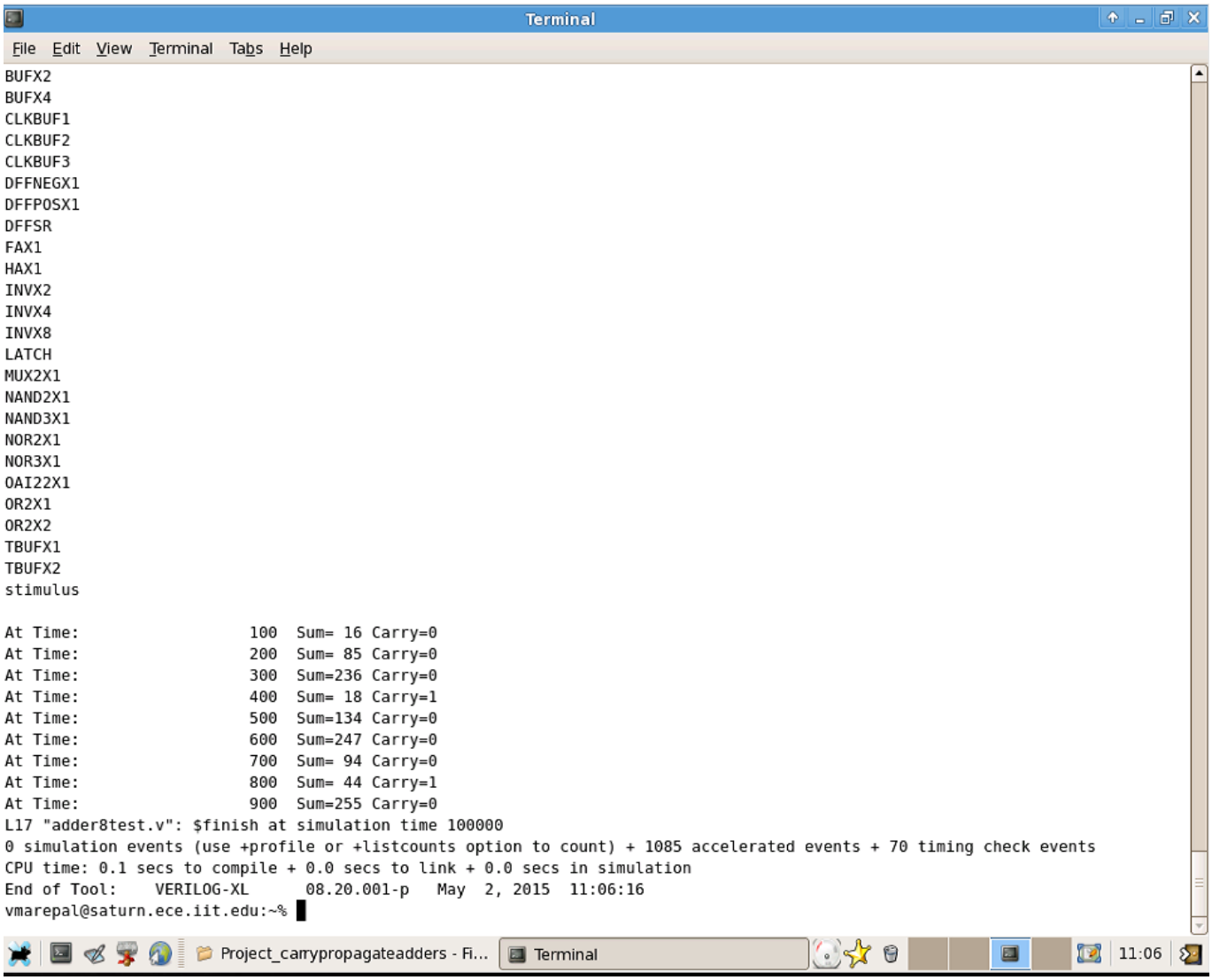
endmodule // stimulus

B. Functional Validation of Carry Ripple Adder 8-bit:

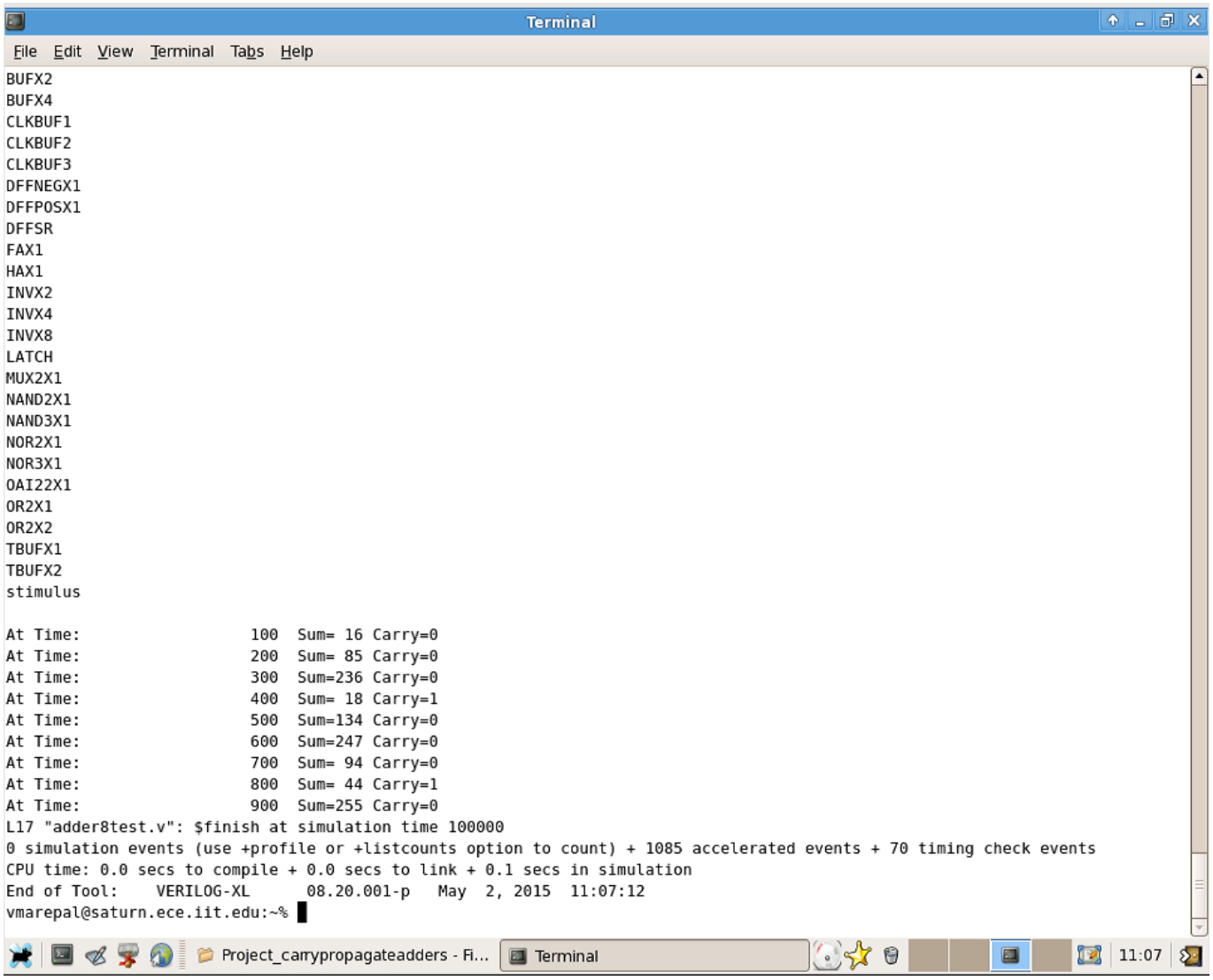
a. Verilog simulation:



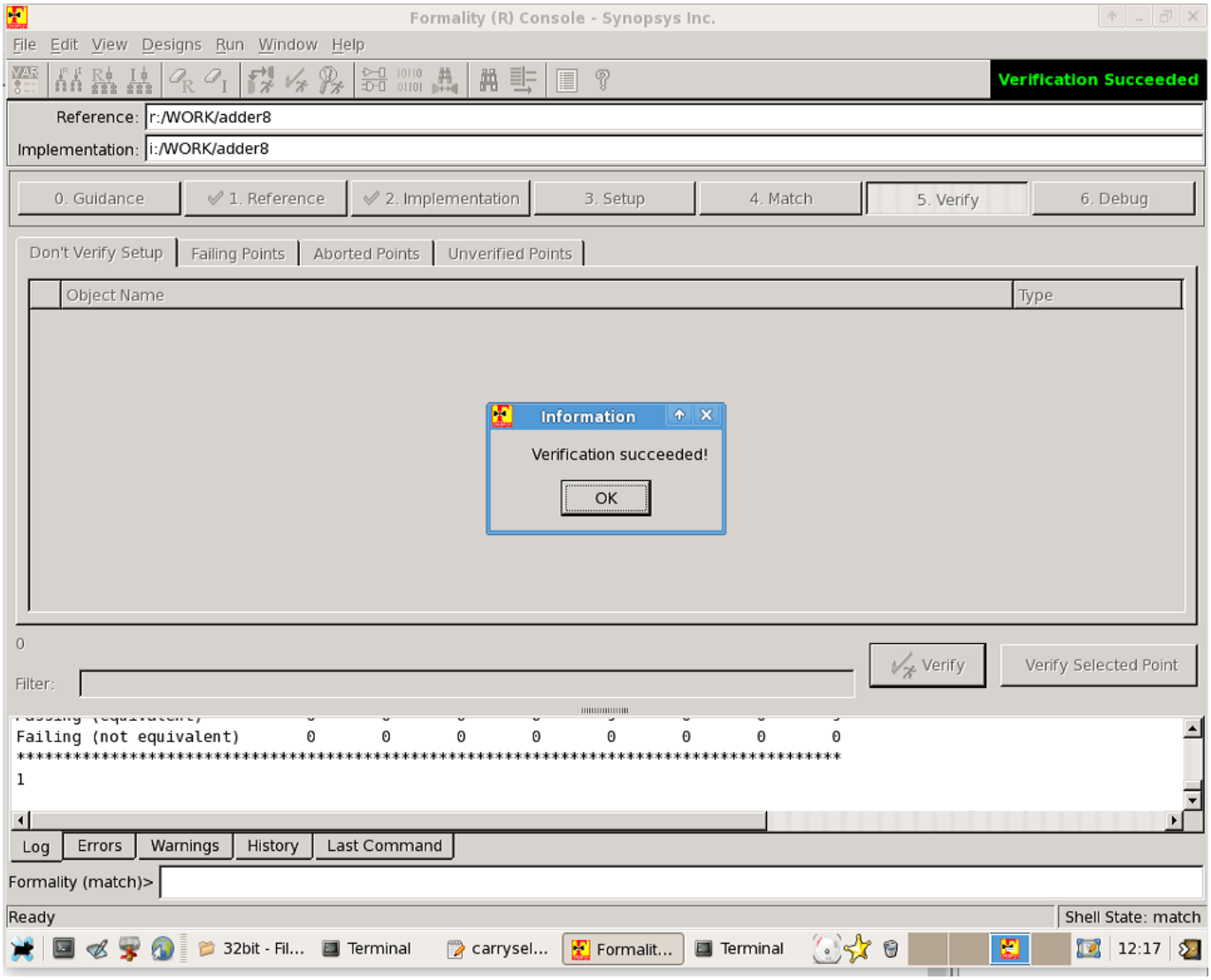
b. DC Compile:


Terminal
File Edit View Terminal Tabs Help
BUFX2
BUFX4
CLKBUF1
CLKBUF2
CLKBUF3
DFFNEGX1
DFFPOSX1
DFFSR
FAX1
HAX1
INVX2
INVX4
INVX8
LATCH
MUX2X1
NAND2X1
NAND3X1
NOR2X1
NOR3X1
OAI22X1
OR2X1
OR2X2
TBUFX1
TBUFX2
stimulus
At Time: 100 Sum= 16 Carry=0
At Time: 200 Sum= 85 Carry=0
At Time: 300 Sum=236 Carry=0
At Time: 400 Sum= 18 Carry=1
At Time: 500 Sum=134 Carry=0
At Time: 600 Sum=247 Carry=0
At Time: 700 Sum= 94 Carry=0
At Time: 800 Sum= 44 Carry=1
At Time: 900 Sum=255 Carry=0
L17 "adder8test.v": $finish at simulation time 100000
0 simulation events (use +profile or +listcounts option to count) + 1085 accelerated events + 70 timing check events
CPU time: 0.1 secs to compile + 0.0 secs to link + 0.0 secs in simulation
End of Tool: VERILOG-XL 08.20.001-p May 2, 2015 11:06:16
vmarepal@saturn.ece.iit.edu:~%
Project_carrypropagateadders - Fi...
Terminal
11:06

c. Encounter:

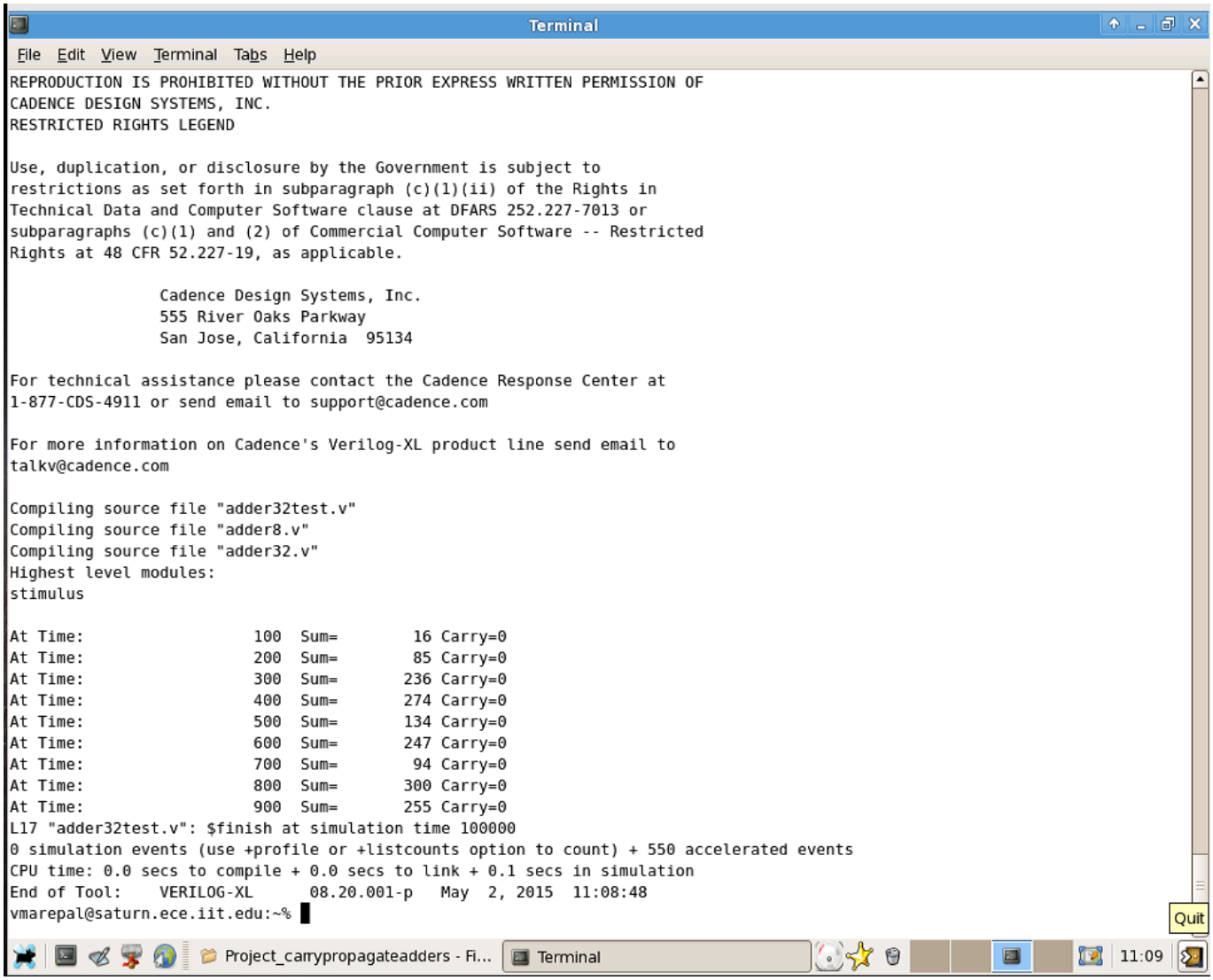

Terminal
File Edit View Terminal Tabs Help
BUFX2
BUFX4
CLKBUF1
CLKBUF2
CLKBUF3
DFFNEGX1
DFFPOSX1
DFFSR
FAX1
HAX1
INVX2
INVX4
INVX8
LATCH
MUX2X1
NAND2X1
NAND3X1
NOR2X1
NOR3X1
OAI22X1
OR2X1
OR2X2
TBUFX1
TBUFX2
stimulus
At Time: 100 Sum= 16 Carry=0
At Time: 200 Sum= 85 Carry=0
At Time: 300 Sum=236 Carry=0
At Time: 400 Sum= 18 Carry=1
At Time: 500 Sum=134 Carry=0
At Time: 600 Sum=247 Carry=0
At Time: 700 Sum= 94 Carry=0
At Time: 800 Sum= 44 Carry=1
At Time: 900 Sum=255 Carry=0
L17 "adder8test.v": $finish at simulation time 100000
0 simulation events (use +profile or +listcounts option to count) + 1085 accelerated events + 70 timing check events
CPU time: 0.0 secs to compile + 0.0 secs to link + 0.1 secs in simulation
End of Tool: VERILOG-XL 08.20.001-p May 2, 2015 11:07:12
vmarepal@saturn.ece.iit.edu:~%
Project_carrypropagateadders - Fi...
Terminal
11:07

d. Formality

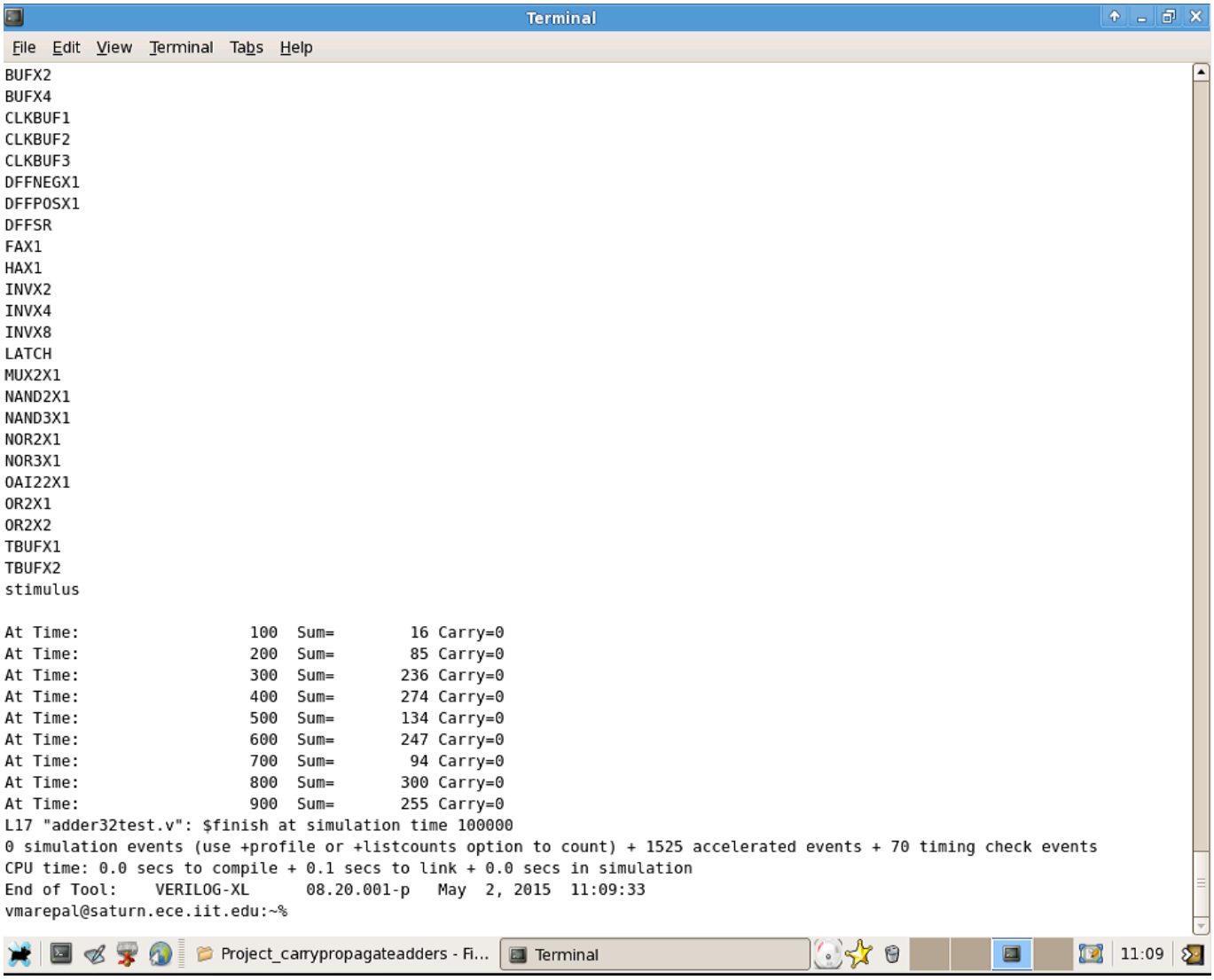

Formality (R) Console - Synopsys Inc.
File Edit View Designs Run Window Help
Verification Succeeded
Reference: r:/WORK/adder8
Implementation: i:/WORK/adder8
0. Guidance
1. Reference
2. Implementation
3. Setup
4. Match
5. Verify
6. Debug
Don't Verify Setup
Failing Points
Aborted Points
Unverified Points
Object Name
Type
Information
Verification succeeded!
OK
0
Filter:
Verify
Verify Selected Point
Failing (not equivalent) 0 0 0 0 0 0 0 0
1
Log
Errors
Warnings
History
Last Command
Formality (match)>
Ready
Shell State: match
32bit - Fil...
Terminal
carrysel...
Formalit...
Terminal
12:17

C. Functional Validation of Carry Ripple Adder 32-bit:

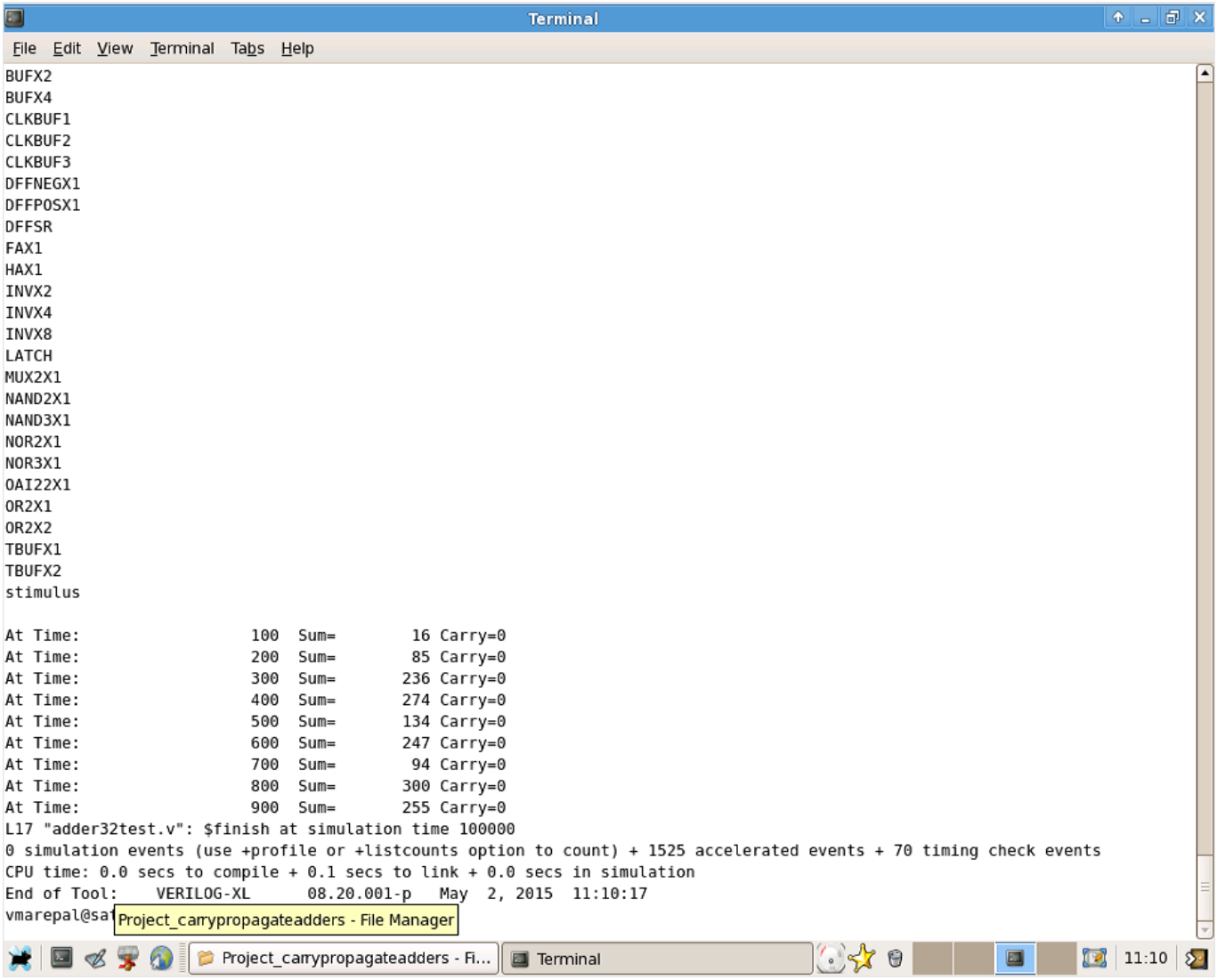
a. Verilog simulation:



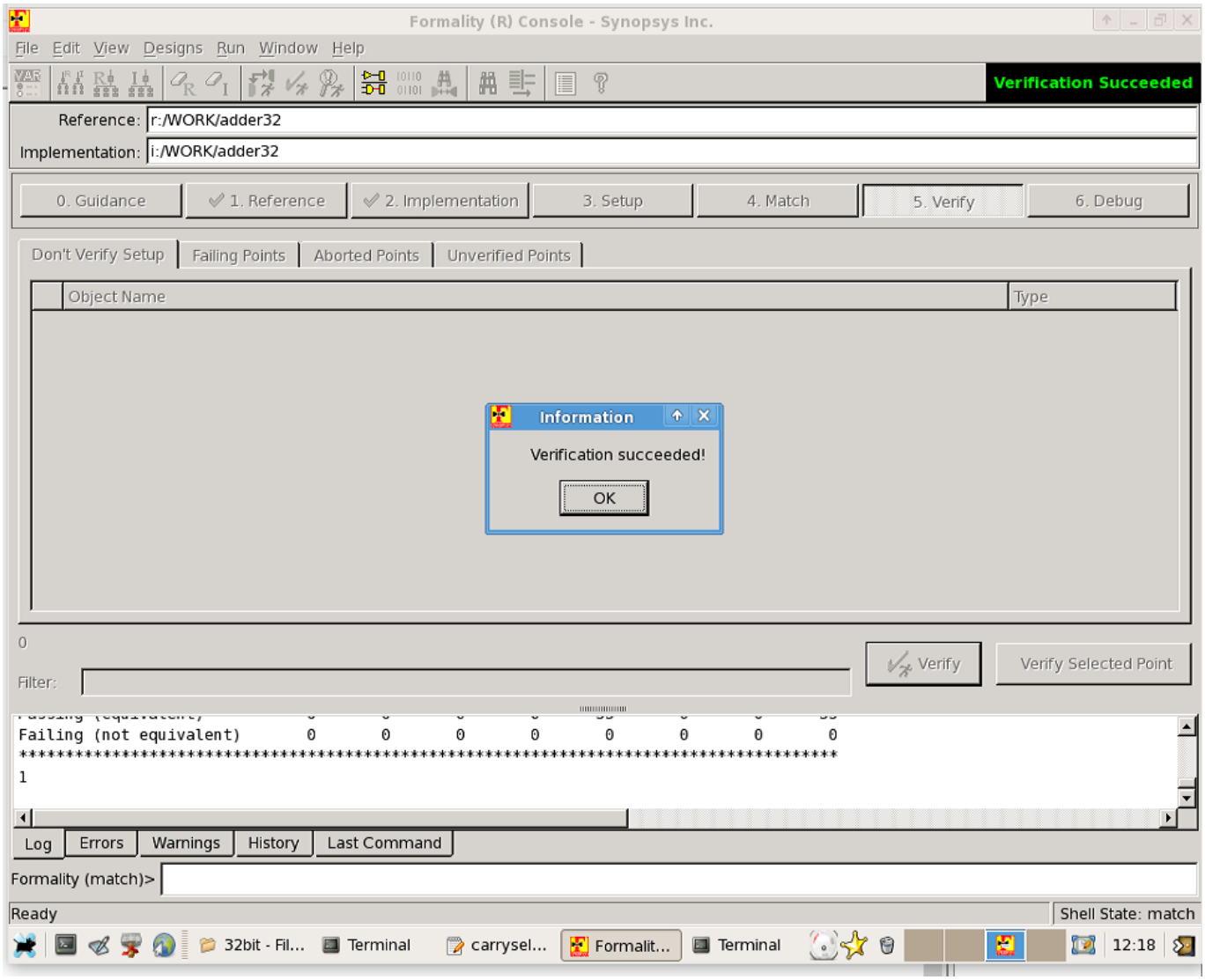
b. DC Compile:


Terminal
File Edit View Terminal Tabs Help
BUFX2
BUFX4
CLKBUF1
CLKBUF2
CLKBUF3
DFFNEGX1
DFFPOSX1
DFFSR
FAX1
HAX1
INVX2
INVX4
INVX8
LATCH
MUX2X1
NAND2X1
NAND3X1
NOR2X1
NOR3X1
OAI22X1
OR2X1
OR2X2
TBUFX1
TBUFX2
stimulus
At Time: 100 Sum= 16 Carry=0
At Time: 200 Sum= 85 Carry=0
At Time: 300 Sum= 236 Carry=0
At Time: 400 Sum= 274 Carry=0
At Time: 500 Sum= 134 Carry=0
At Time: 600 Sum= 247 Carry=0
At Time: 700 Sum= 94 Carry=0
At Time: 800 Sum= 300 Carry=0
At Time: 900 Sum= 255 Carry=0
L17 "adder32test.v": $finish at simulation time 100000
0 simulation events (use +profile or +listcounts option to count) + 1525 accelerated events + 70 timing check events
CPU time: 0.0 secs to compile + 0.1 secs to link + 0.0 secs in simulation
End of Tool: VERILOG-XL 08.20.001-p May 2, 2015 11:09:33
vmarepal@saturn.ece.iit.edu:~%
Project_carrypropagateadders - Fi...
Terminal
11:09

c. Encounter:

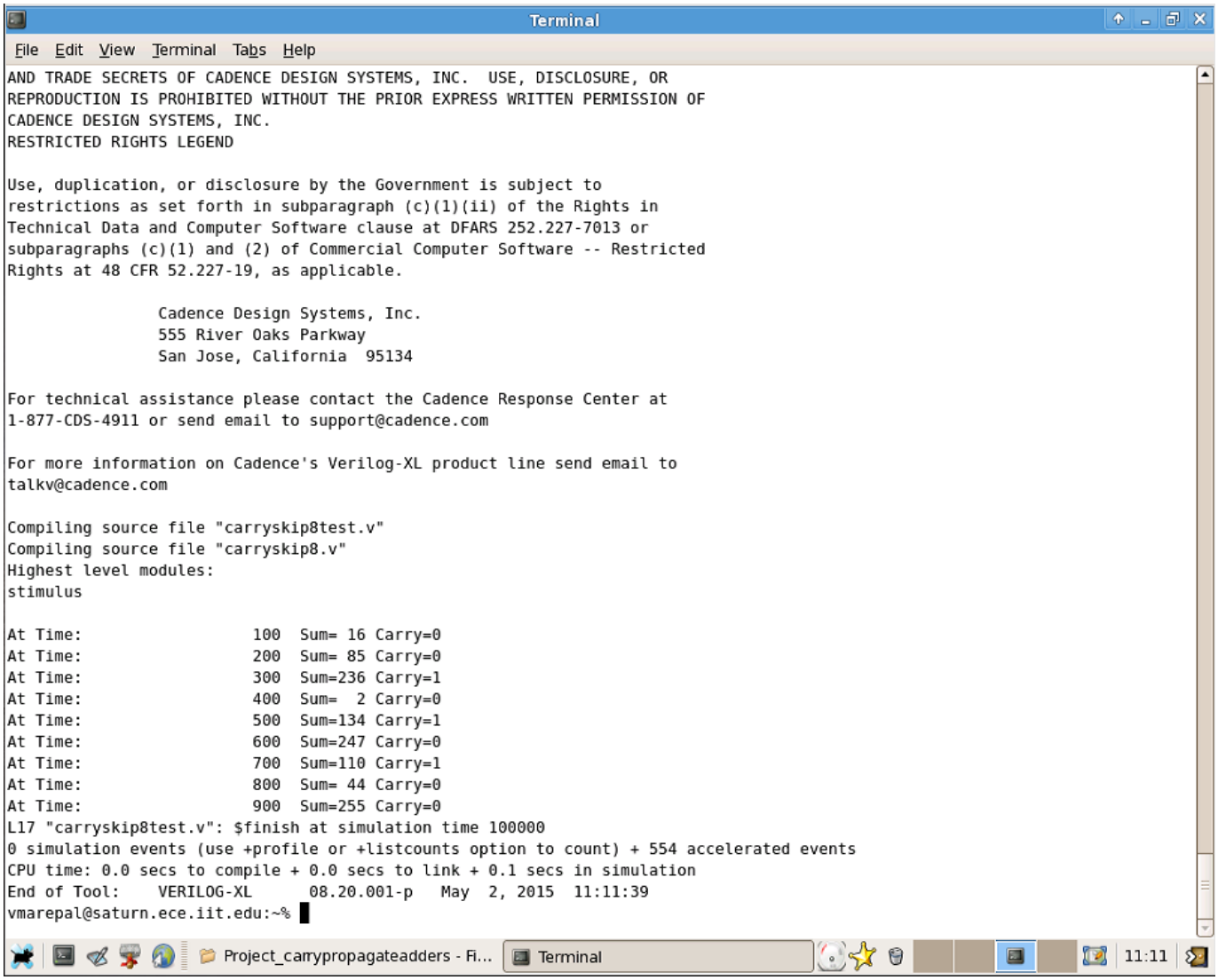

Terminal
File Edit View Terminal Tabs Help
BUFX2
BUFX4
CLKBUF1
CLKBUF2
CLKBUF3
DFFNEGX1
DFFPOSX1
DFFSR
FAX1
HAX1
INVX2
INVX4
INVX8
LATCH
MUX2X1
NAND2X1
NAND3X1
NOR2X1
NOR3X1
OAI22X1
OR2X1
OR2X2
TBUFX1
TBUFX2
stimulus
At Time: 100 Sum= 16 Carry=0
At Time: 200 Sum= 85 Carry=0
At Time: 300 Sum= 236 Carry=0
At Time: 400 Sum= 274 Carry=0
At Time: 500 Sum= 134 Carry=0
At Time: 600 Sum= 247 Carry=0
At Time: 700 Sum= 94 Carry=0
At Time: 800 Sum= 300 Carry=0
At Time: 900 Sum= 255 Carry=0
L17 "adder32test.v": $finish at simulation time 100000
0 simulation events (use +profile or +listcounts option to count) + 1525 accelerated events + 70 timing check events
CPU time: 0.0 secs to compile + 0.1 secs to link + 0.0 secs in simulation
End of Tool: VERILOG-XL 08.20.001-p May 2, 2015 11:10:17
vmarepal@sa
Project_carrypropagateadders - File Manager
Project_carrypropagateadders - Fi...
Terminal
11:10

d. Formality

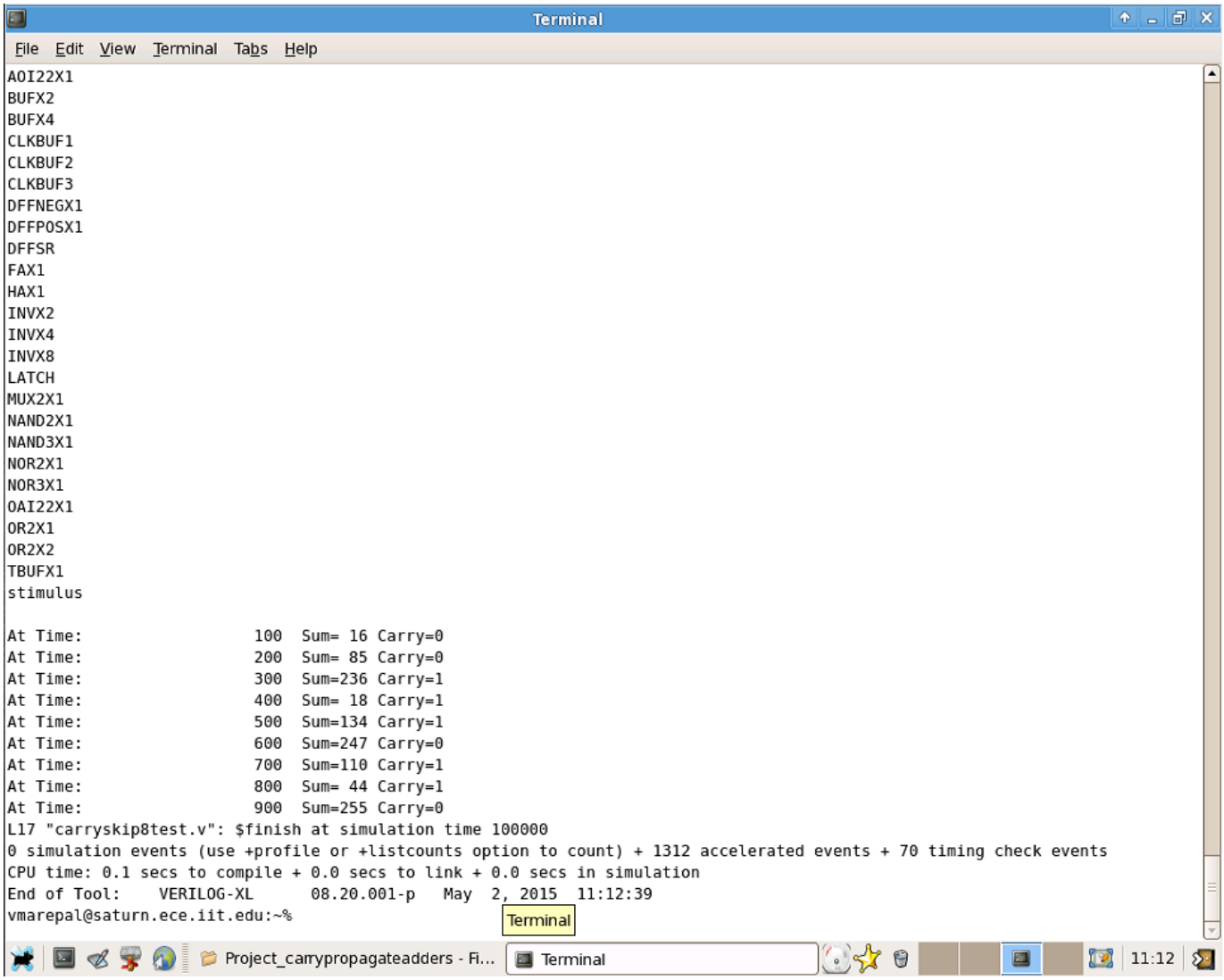

Formality (R) Console - Synopsys Inc.
File Edit View Designs Run Window Help
Verification Succeeded
Reference: r:/WORK/adder32
Implementation: i:/WORK/adder32
0. Guidance
1. Reference
2. Implementation
3. Setup
4. Match
5. Verify
6. Debug
Don't Verify Setup
Failing Points
Aborted Points
Unverified Points
Object Name
Type
Information
Verification succeeded!
OK
0
Filter:
Verify
Verify Selected Point
Failing (not equivalent) 0 0 0 0 0 0 0 0
1
Log
Errors
Warnings
History
Last Command
Formality (match)>
Ready
Shell State: match
32bit - Fil...
Terminal
carrysel...
Formalit...
Terminal
12:18

D. Functional Validation of Carry Skip Adder 8-bit:

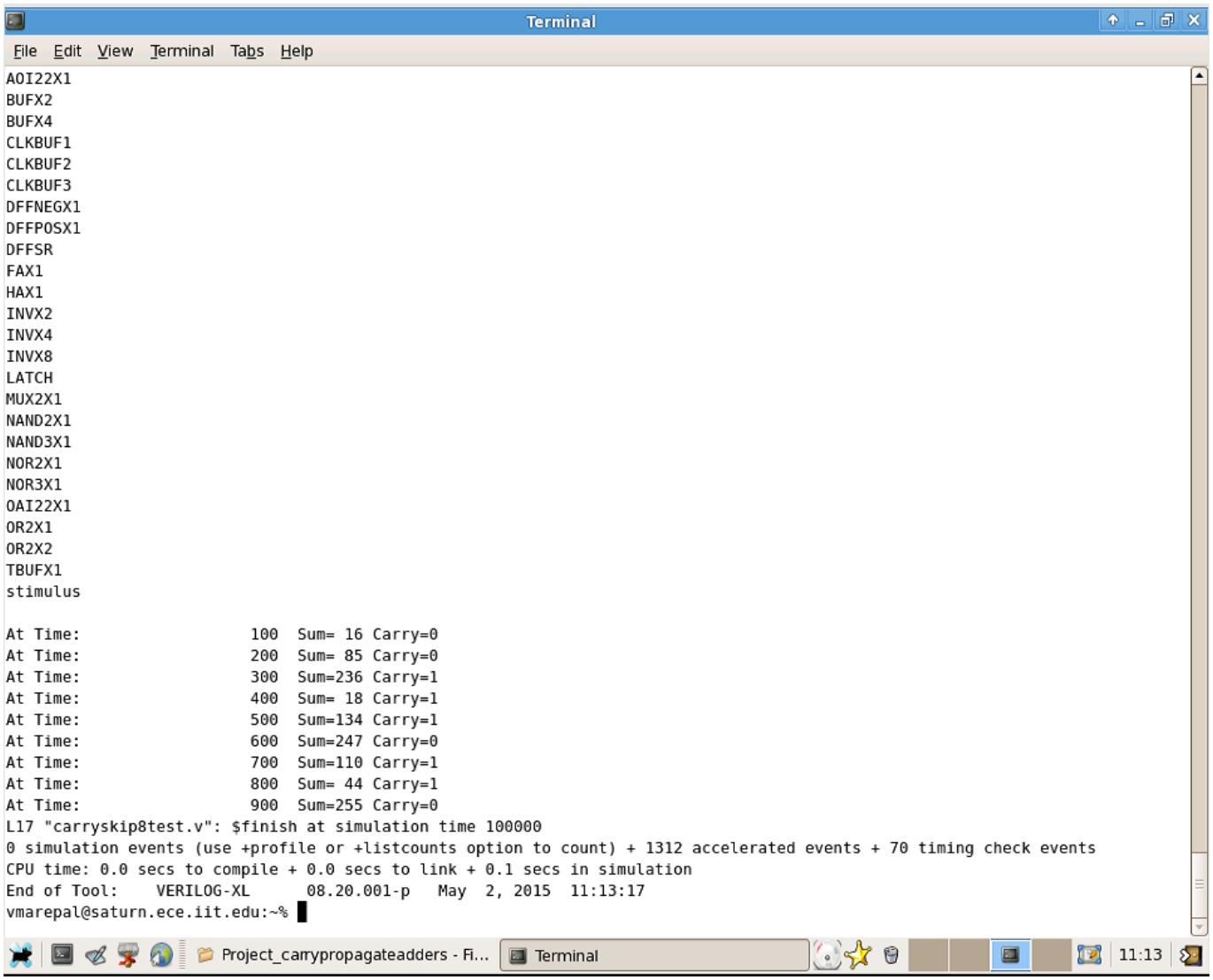
a. Verilog simulation:



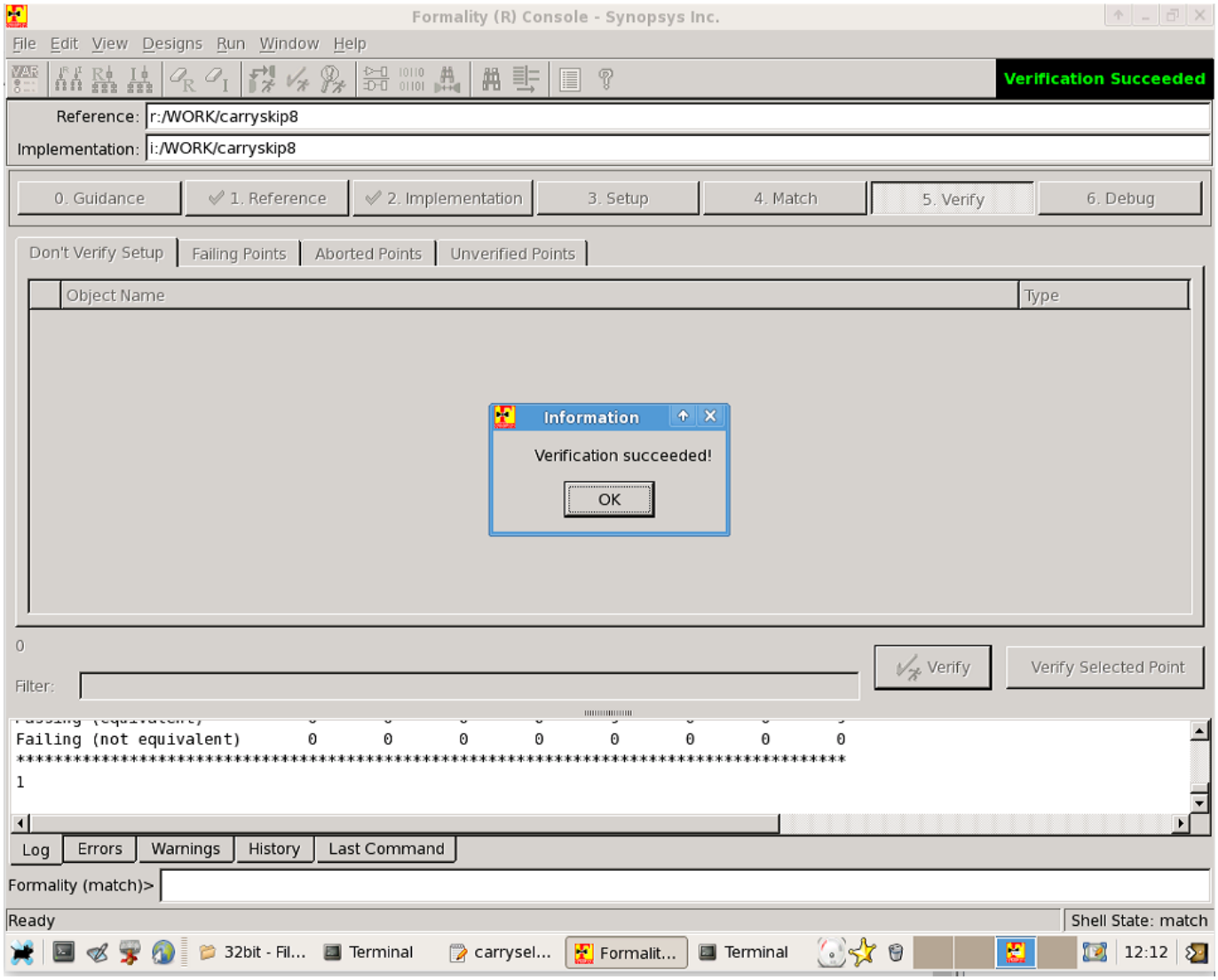
b. DC Compile:


Terminal
File Edit View Terminal Tabs Help
AOI22X1
BUFX2
BUFX4
CLKBUF1
CLKBUF2
CLKBUF3
DFFNEGX1
DFFPOSX1
DFFSR
FAX1
HAX1
INVX2
INVX4
INVX8
LATCH
MUX2X1
NAND2X1
NAND3X1
NOR2X1
NOR3X1
OAI22X1
OR2X1
OR2X2
TBUFX1
stimulus
At Time: 100 Sum= 16 Carry=0
At Time: 200 Sum= 85 Carry=0
At Time: 300 Sum=236 Carry=1
At Time: 400 Sum= 18 Carry=1
At Time: 500 Sum=134 Carry=1
At Time: 600 Sum=247 Carry=0
At Time: 700 Sum=110 Carry=1
At Time: 800 Sum= 44 Carry=1
At Time: 900 Sum=255 Carry=0
L17 "carryskip8test.v": $finish at simulation time 100000
0 simulation events (use +profile or +listcounts option to count) + 1312 accelerated events + 70 timing check events
CPU time: 0.1 secs to compile + 0.0 secs to link + 0.0 secs in simulation
End of Tool: VERILOG-XL 08.20.001-p May 2, 2015 11:12:39
vmarepal@saturn.ece.iit.edu:~%
Terminal
Project_carrypropagateadders - Fi...
Terminal
11:12

c. Encounter:

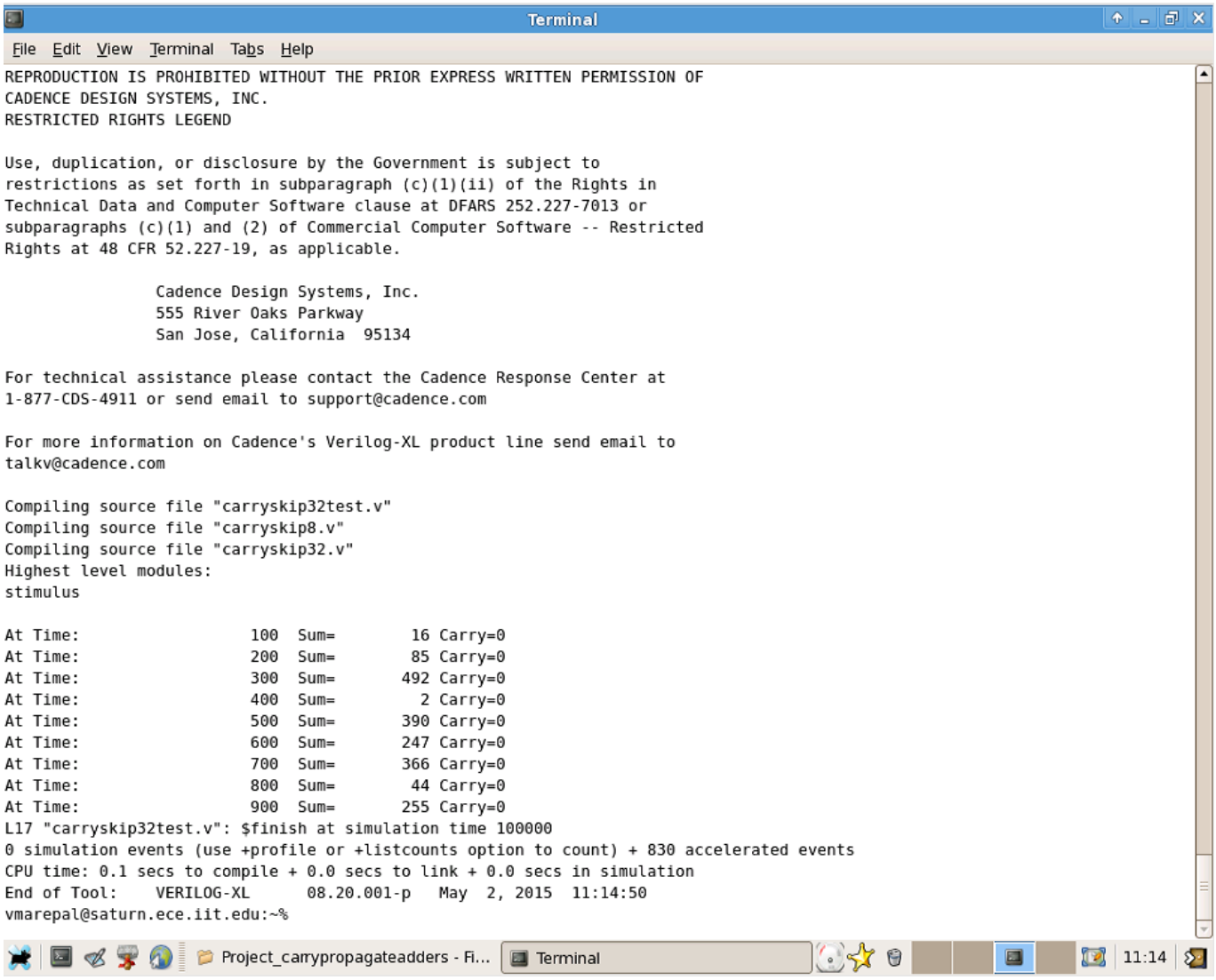

Terminal
File Edit View Terminal Tabs Help
AOI22X1
BUFX2
BUFX4
CLKBUF1
CLKBUF2
CLKBUF3
DFFNEGX1
DFFPOSX1
DFFSR
FAX1
HAX1
INVX2
INVX4
INVX8
LATCH
MUX2X1
NAND2X1
NAND3X1
NOR2X1
NOR3X1
OAI22X1
OR2X1
OR2X2
TBUFX1
stimulus
At Time: 100 Sum= 16 Carry=0
At Time: 200 Sum= 85 Carry=0
At Time: 300 Sum=236 Carry=1
At Time: 400 Sum= 18 Carry=1
At Time: 500 Sum=134 Carry=1
At Time: 600 Sum=247 Carry=0
At Time: 700 Sum=110 Carry=1
At Time: 800 Sum= 44 Carry=1
At Time: 900 Sum=255 Carry=0
L17 "carryskip8test.v": $finish at simulation time 100000
0 simulation events (use +profile or +listcounts option to count) + 1312 accelerated events + 70 timing check events
CPU time: 0.0 secs to compile + 0.0 secs to link + 0.1 secs in simulation
End of Tool: VERILOG-XL 08.20.001-p May 2, 2015 11:13:17
vmarepal@saturn.ece.iit.edu:~%
Project_carrypropagateadders - Fi...
Terminal
11:13

d. Formality

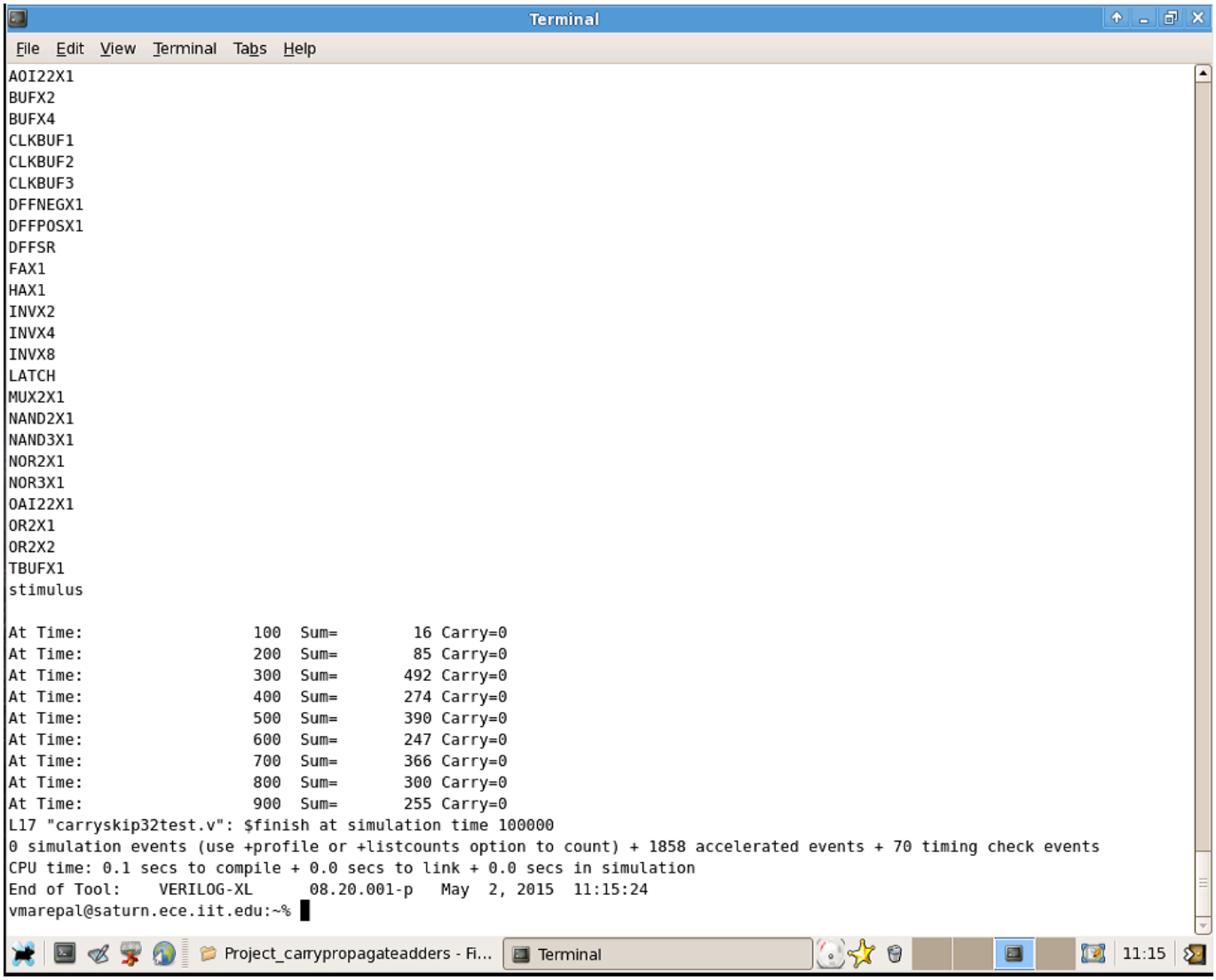

Formality (R) Console - Synopsys Inc.
File Edit View Designs Run Window Help
Verification Succeeded
Reference: r:/WORK/carryskip8
Implementation: i:/WORK/carryskip8
0. Guidance
1. Reference
2. Implementation
3. Setup
4. Match
5. Verify
6. Debug
Don't Verify Setup
Failing Points
Aborted Points
Unverified Points
Object Name
Type
Information
Verification succeeded!
OK
0
Filter:
Verify
Verify Selected Point
Failing (not equivalent) 0 0 0 0 0 0 0 0
1
Log
Errors
Warnings
History
Last Command
Formality (match)>
Ready
Shell State: match
32bit - Fil...
Terminal
carrysel...
Formalit...
Terminal
12:12

E. Functional Validation of Carry Skip Adder 32-bit:

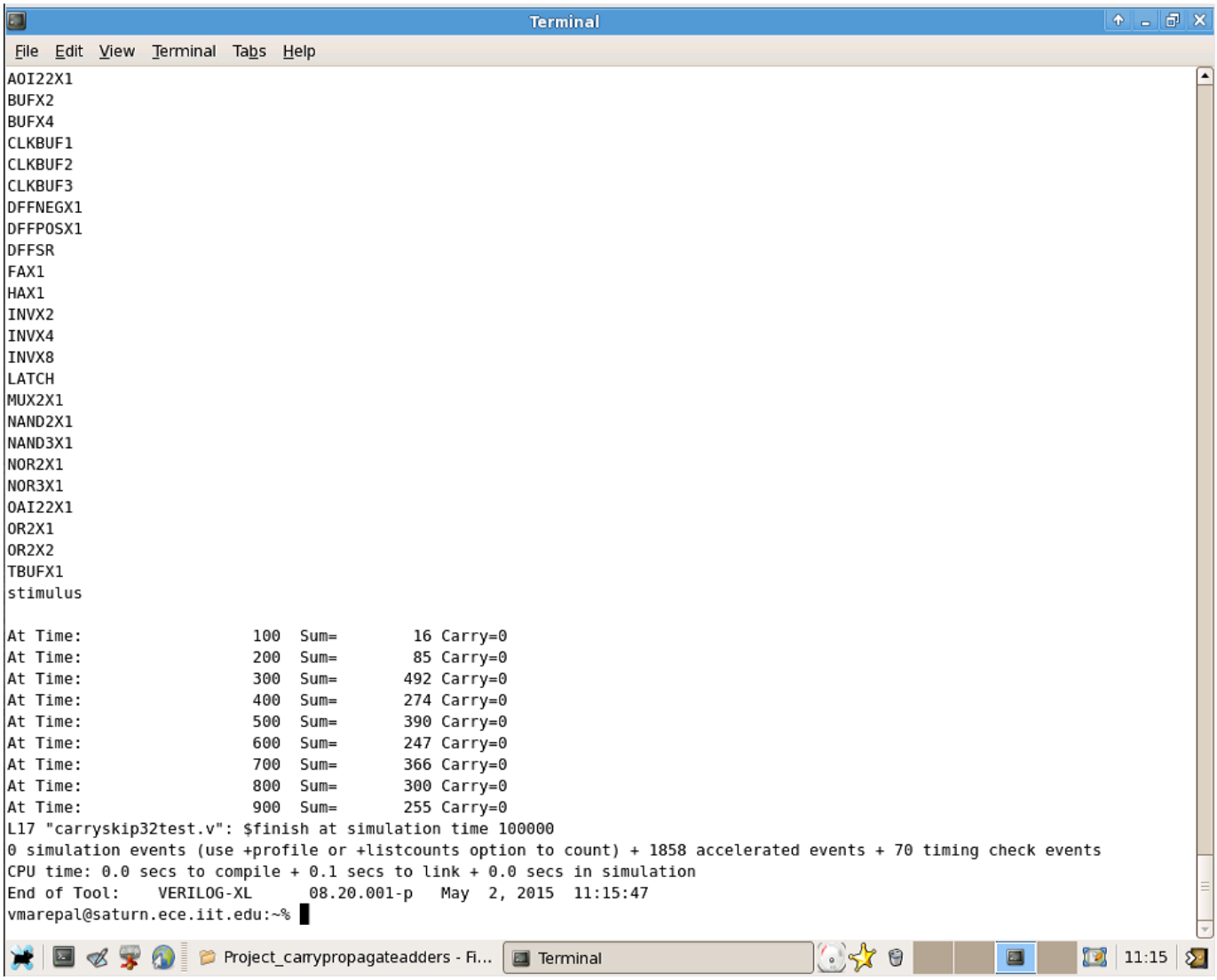
a. Verilog simulation:



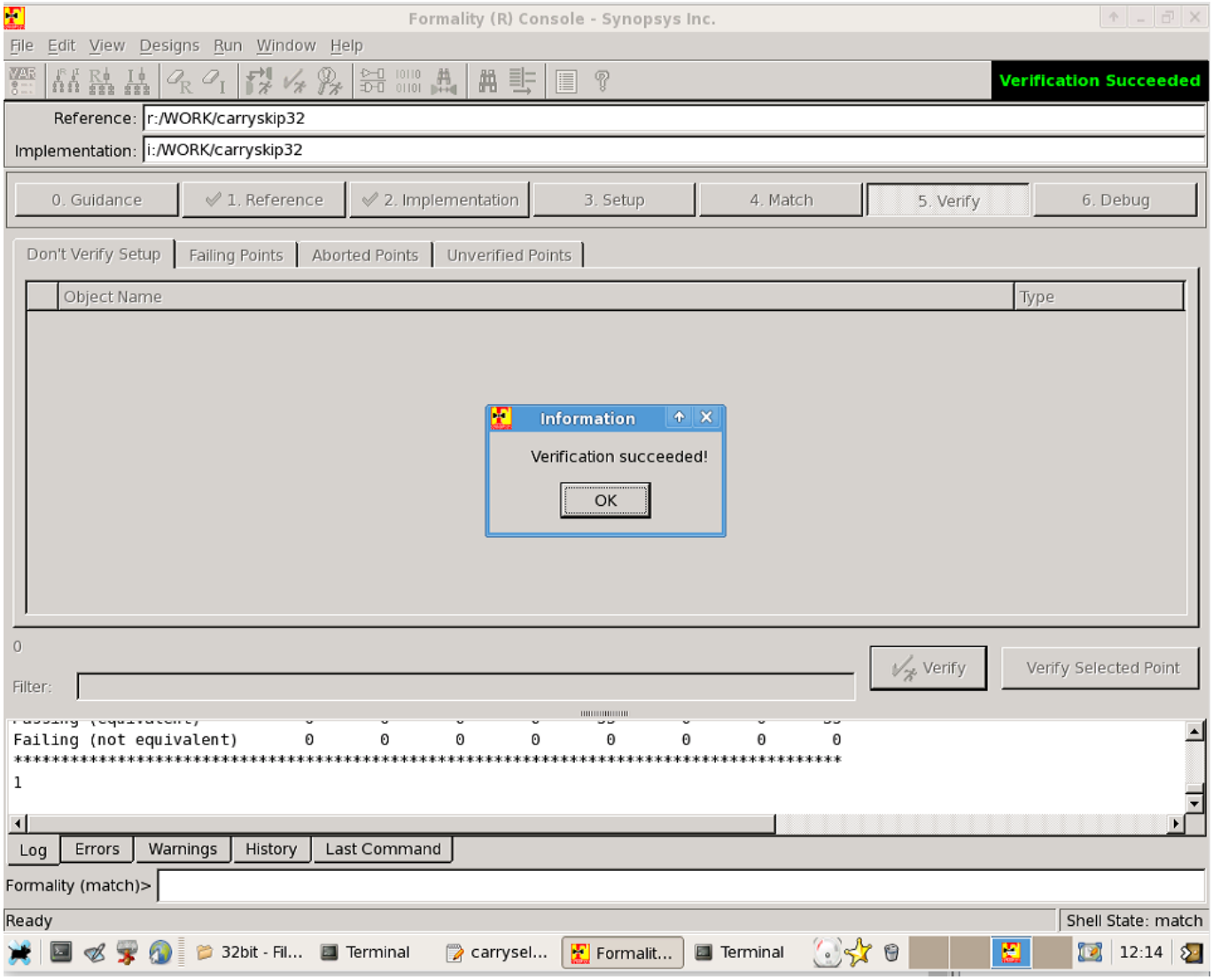
b. DC Compile:


Terminal
File Edit View Terminal Tabs Help
AOI22X1
BUFX2
BUFX4
CLKBUF1
CLKBUF2
CLKBUF3
DFFNEGX1
DFFPOSX1
DFFSR
FAX1
HAX1
INVX2
INVX4
INVX8
LATCH
MUX2X1
NAND2X1
NAND3X1
NOR2X1
NOR3X1
OAI22X1
OR2X1
OR2X2
TBUFX1
stimulus
At Time: 100 Sum= 16 Carry=0
At Time: 200 Sum= 85 Carry=0
At Time: 300 Sum= 492 Carry=0
At Time: 400 Sum= 274 Carry=0
At Time: 500 Sum= 390 Carry=0
At Time: 600 Sum= 247 Carry=0
At Time: 700 Sum= 366 Carry=0
At Time: 800 Sum= 300 Carry=0
At Time: 900 Sum= 255 Carry=0
L17 "carryskip32test.v": $finish at simulation time 100000
0 simulation events (use +profile or +listcounts option to count) + 1858 accelerated events + 70 timing check events
CPU time: 0.1 secs to compile + 0.0 secs to link + 0.0 secs in simulation
End of Tool: VERILOG-XL 08.20.001-p May 2, 2015 11:15:24
vmarepal@saturn.ece.iit.edu:~%
Project_carrypropagateadders - Fi...
Terminal
11:15

c. Encounter:

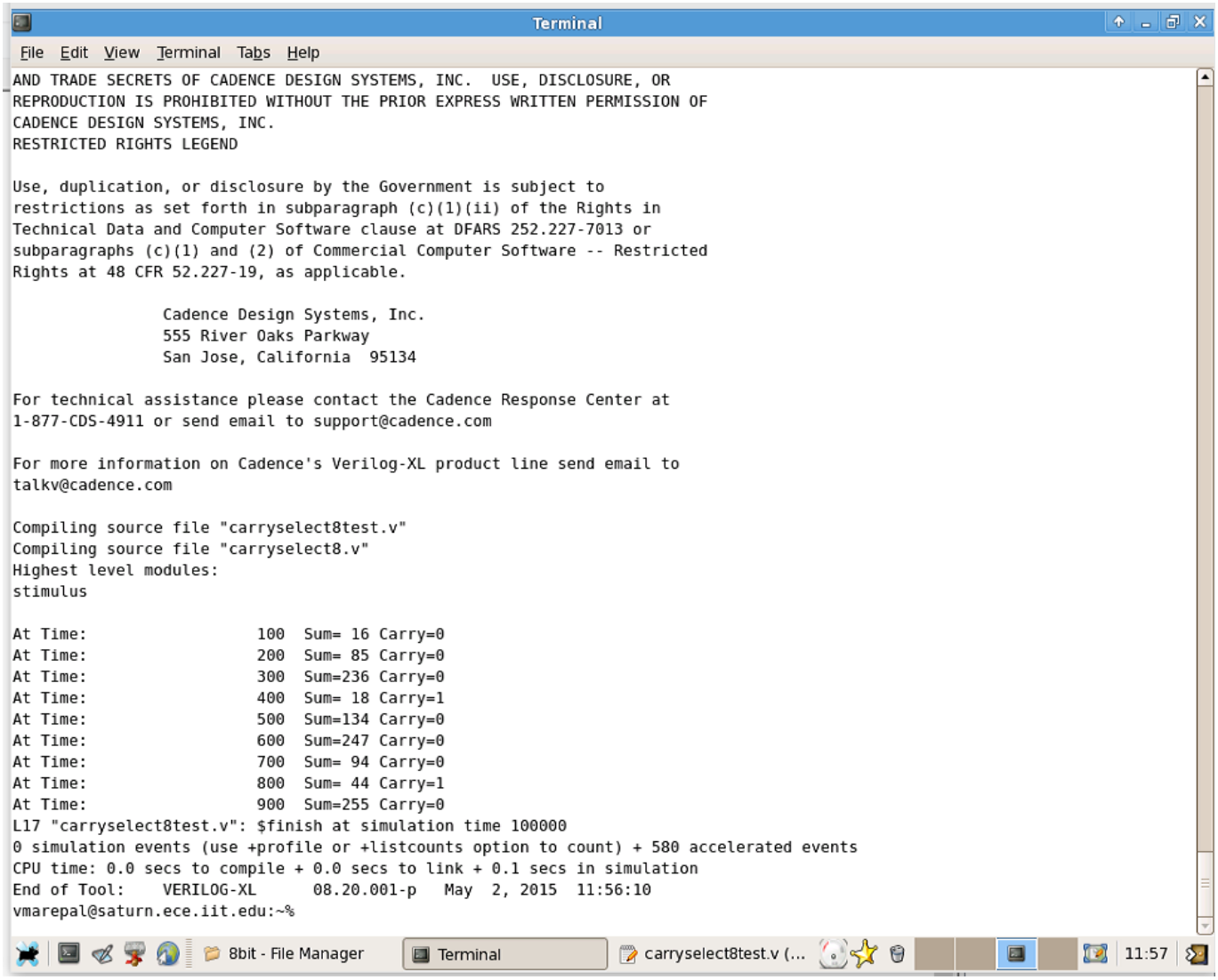

Terminal
File Edit View Terminal Tabs Help
AOI22X1
BUFX2
BUFX4
CLKBUF1
CLKBUF2
CLKBUF3
DFFNEGX1
DFFPOSX1
DFFSR
FAX1
HAX1
INVX2
INVX4
INVX8
LATCH
MUX2X1
NAND2X1
NAND3X1
NOR2X1
NOR3X1
OAI22X1
OR2X1
OR2X2
TBUFX1
stimulus
At Time: 100 Sum= 16 Carry=0
At Time: 200 Sum= 85 Carry=0
At Time: 300 Sum= 492 Carry=0
At Time: 400 Sum= 274 Carry=0
At Time: 500 Sum= 390 Carry=0
At Time: 600 Sum= 247 Carry=0
At Time: 700 Sum= 366 Carry=0
At Time: 800 Sum= 300 Carry=0
At Time: 900 Sum= 255 Carry=0
L17 "carryskip32test.v": $finish at simulation time 100000
0 simulation events (use +profile or +listcounts option to count) + 1858 accelerated events + 70 timing check events
CPU time: 0.0 secs to compile + 0.1 secs to link + 0.0 secs in simulation
End of Tool: VERILOG-XL 08.20.001-p May 2, 2015 11:15:47
vmarepal@saturn.ece.iit.edu:~%
Project_carrypropagateadders - Fi...
Terminal
11:15

d. Formality

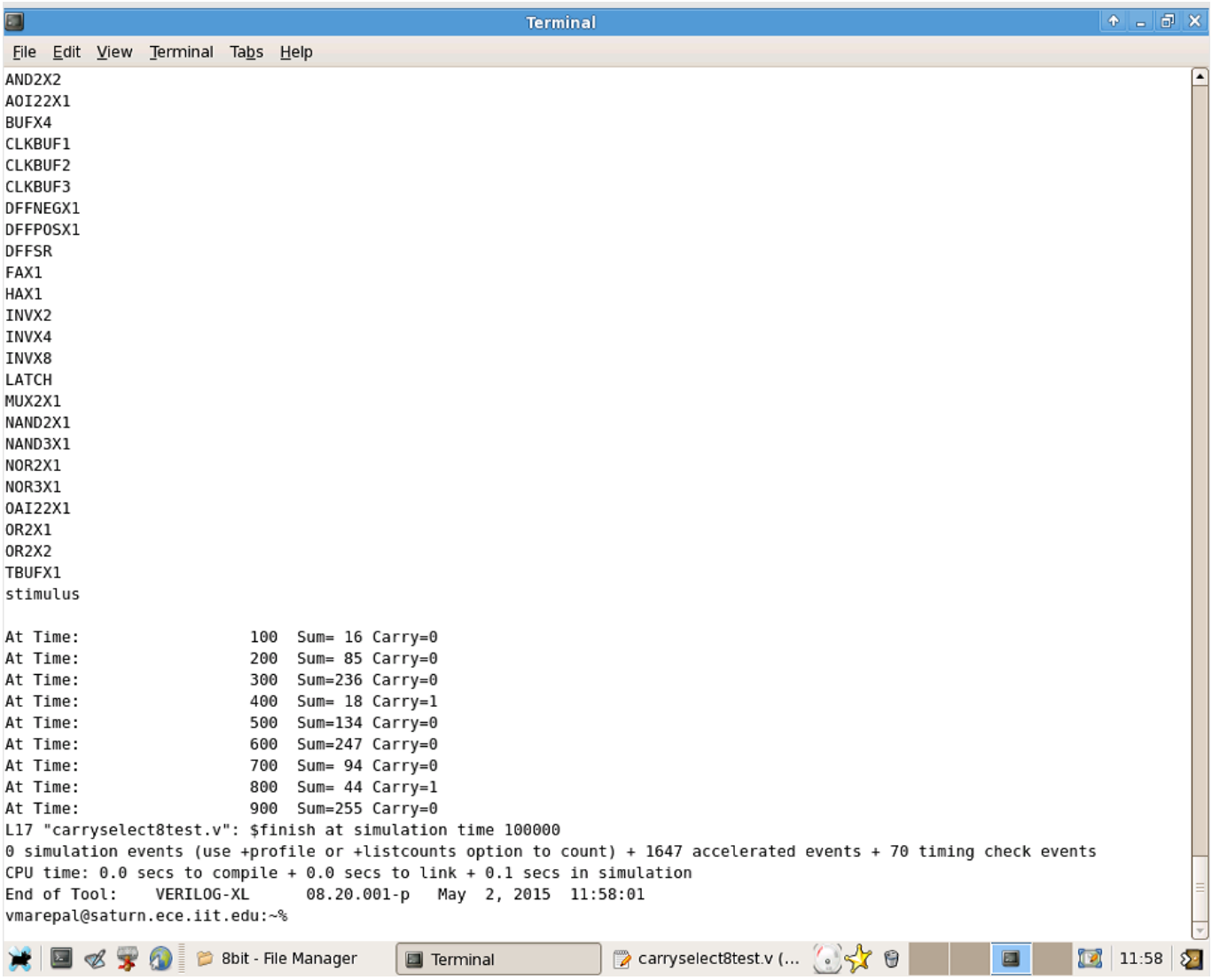

Formality (R) Console - Synopsys Inc.
File Edit View Designs Run Window Help
Verification Succeeded
Reference: r:/WORK/carryskip32
Implementation: i:/WORK/carryskip32
0. Guidance
1. Reference
2. Implementation
3. Setup
4. Match
5. Verify
6. Debug
Don't Verify Setup
Failing Points
Aborted Points
Unverified Points
Object Name
Type
Information
Verification succeeded!
OK
0
Filter:
Verify
Verify Selected Point
Failing (not equivalent) 0 0 0 0 0 0 0 0
1
Log
Errors
Warnings
History
Last Command
Formality (match)>
Ready
Shell State: match
32bit - Fil...
Terminal
carrysel...
Formalit...
Terminal
12:14

F. Functional Validation of Carry Select Adder 8-bit:

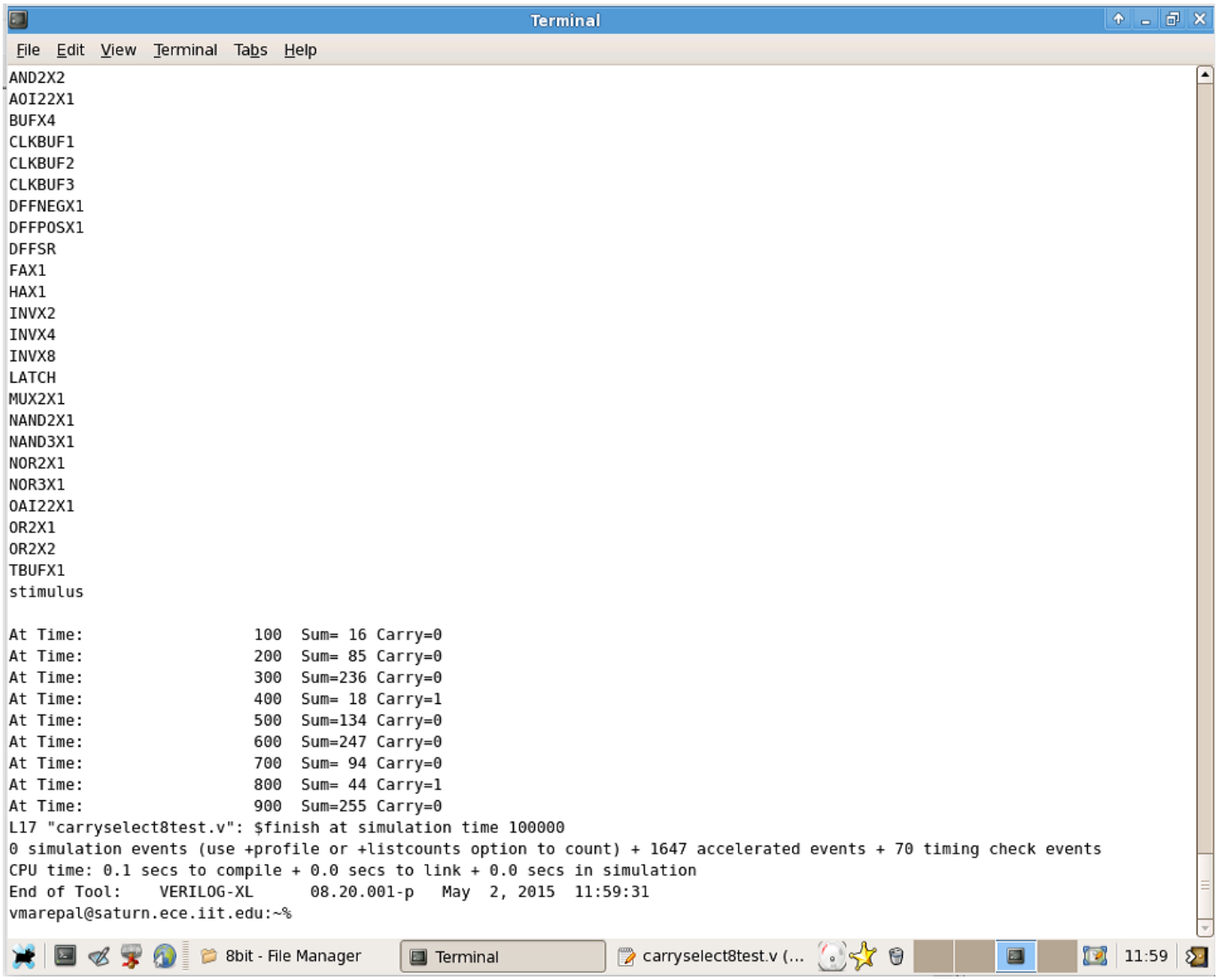
a. Verilog simulation:



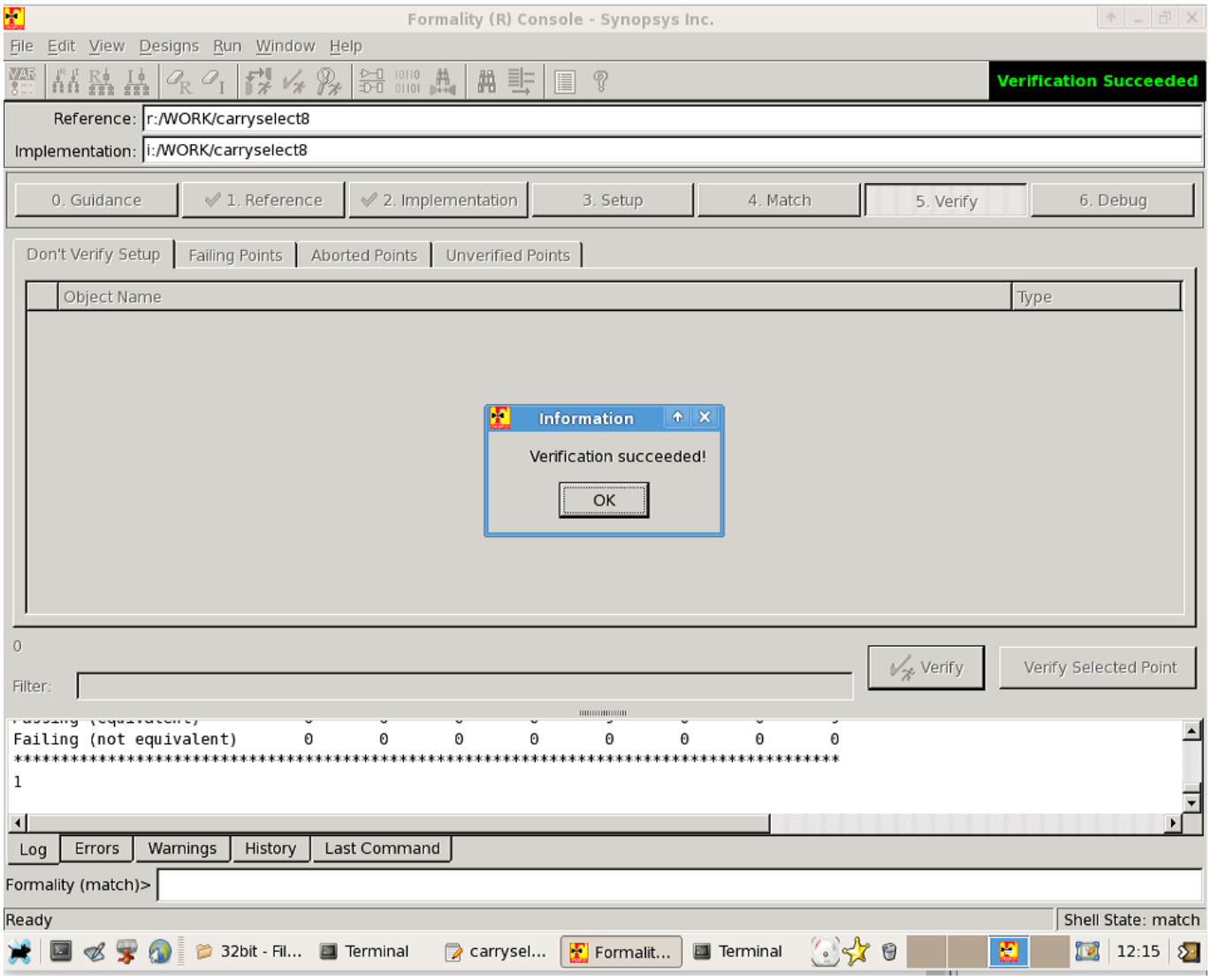
b. DC Compile:


Terminal
File Edit View Terminal Tabs Help
AND2X2
AOI22X1
BUFX4
CLKBUF1
CLKBUF2
CLKBUF3
DFFNEGX1
DFFPOSX1
DFFSR
FAX1
HAX1
INVX2
INVX4
INVX8
LATCH
MUX2X1
NAND2X1
NAND3X1
NOR2X1
NOR3X1
OAI22X1
OR2X1
OR2X2
TBUFX1
stimulus
At Time: 100 Sum= 16 Carry=0
At Time: 200 Sum= 85 Carry=0
At Time: 300 Sum=236 Carry=0
At Time: 400 Sum= 18 Carry=1
At Time: 500 Sum=134 Carry=0
At Time: 600 Sum=247 Carry=0
At Time: 700 Sum= 94 Carry=0
At Time: 800 Sum= 44 Carry=1
At Time: 900 Sum=255 Carry=0
L17 "carryselect8test.v": $finish at simulation time 100000
0 simulation events (use +profile or +listcounts option to count) + 1647 accelerated events + 70 timing check events
CPU time: 0.0 secs to compile + 0.0 secs to link + 0.1 secs in simulation
End of Tool: VERILOG-XL 08.20.001-p May 2, 2015 11:58:01
vmarepal@saturn.ece.iit.edu:~%
8bit - File Manager
Terminal
carryselect8test.v (...
11:58

c. Encounter:

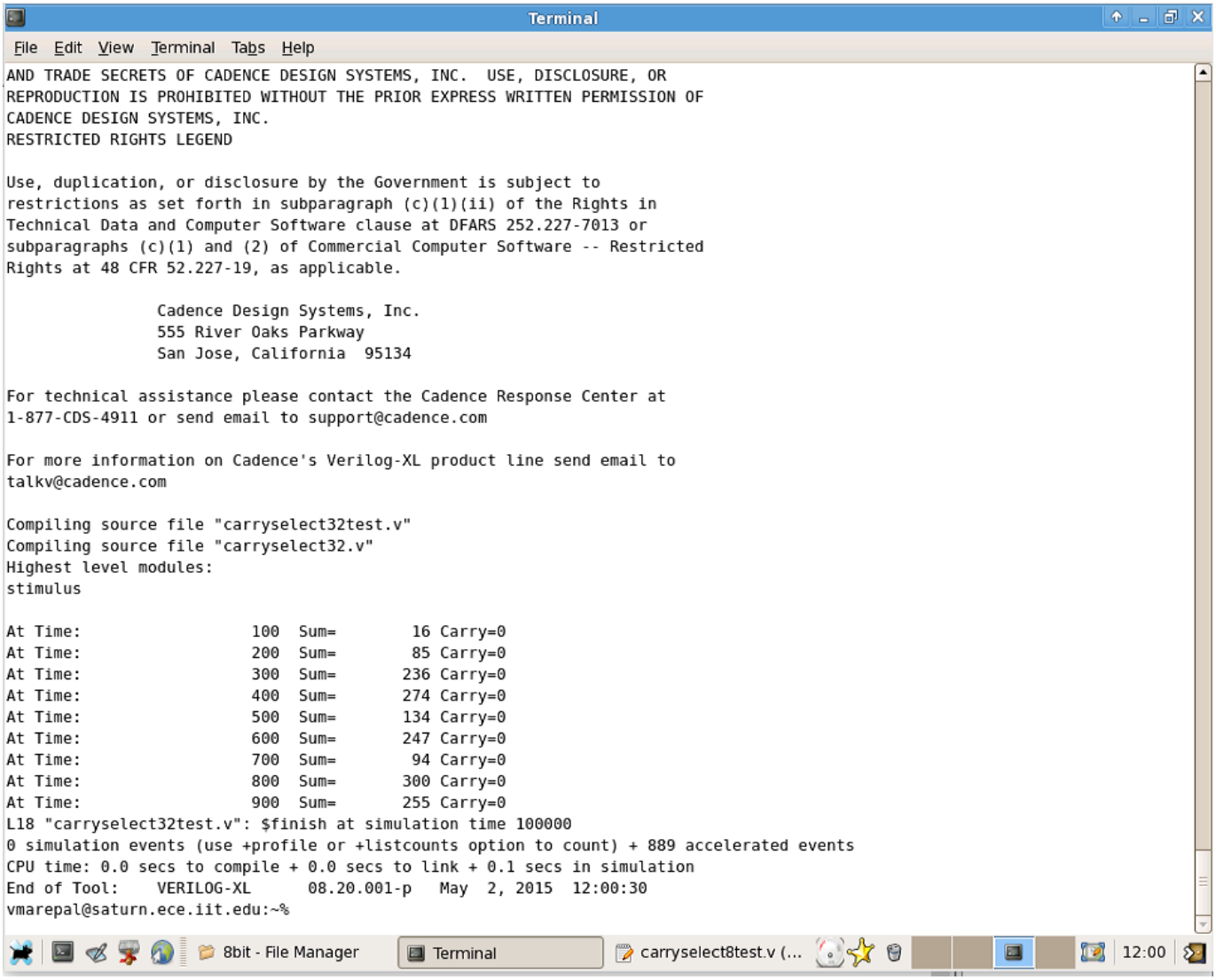

Terminal
File Edit View Terminal Tabs Help
AND2X2
AOI22X1
BUFX4
CLKBUF1
CLKBUF2
CLKBUF3
DFFNEGX1
DFFPOSX1
DFFSR
FAX1
HAX1
INVX2
INVX4
INVX8
LATCH
MUX2X1
NAND2X1
NAND3X1
NOR2X1
NOR3X1
OAI22X1
OR2X1
OR2X2
TBUFX1
stimulus
At Time: 100 Sum= 16 Carry=0
At Time: 200 Sum= 85 Carry=0
At Time: 300 Sum=236 Carry=0
At Time: 400 Sum= 18 Carry=1
At Time: 500 Sum=134 Carry=0
At Time: 600 Sum=247 Carry=0
At Time: 700 Sum= 94 Carry=0
At Time: 800 Sum= 44 Carry=1
At Time: 900 Sum=255 Carry=0
L17 "carryselect8test.v": $finish at simulation time 100000
0 simulation events (use +profile or +listcounts option to count) + 1647 accelerated events + 70 timing check events
CPU time: 0.1 secs to compile + 0.0 secs to link + 0.0 secs in simulation
End of Tool: VERILOG-XL 08.20.001-p May 2, 2015 11:59:31
vmarepal@saturn.ece.iit.edu:~%
8bit - File Manager
Terminal
carryselect8test.v (...
11:59

d. Formality

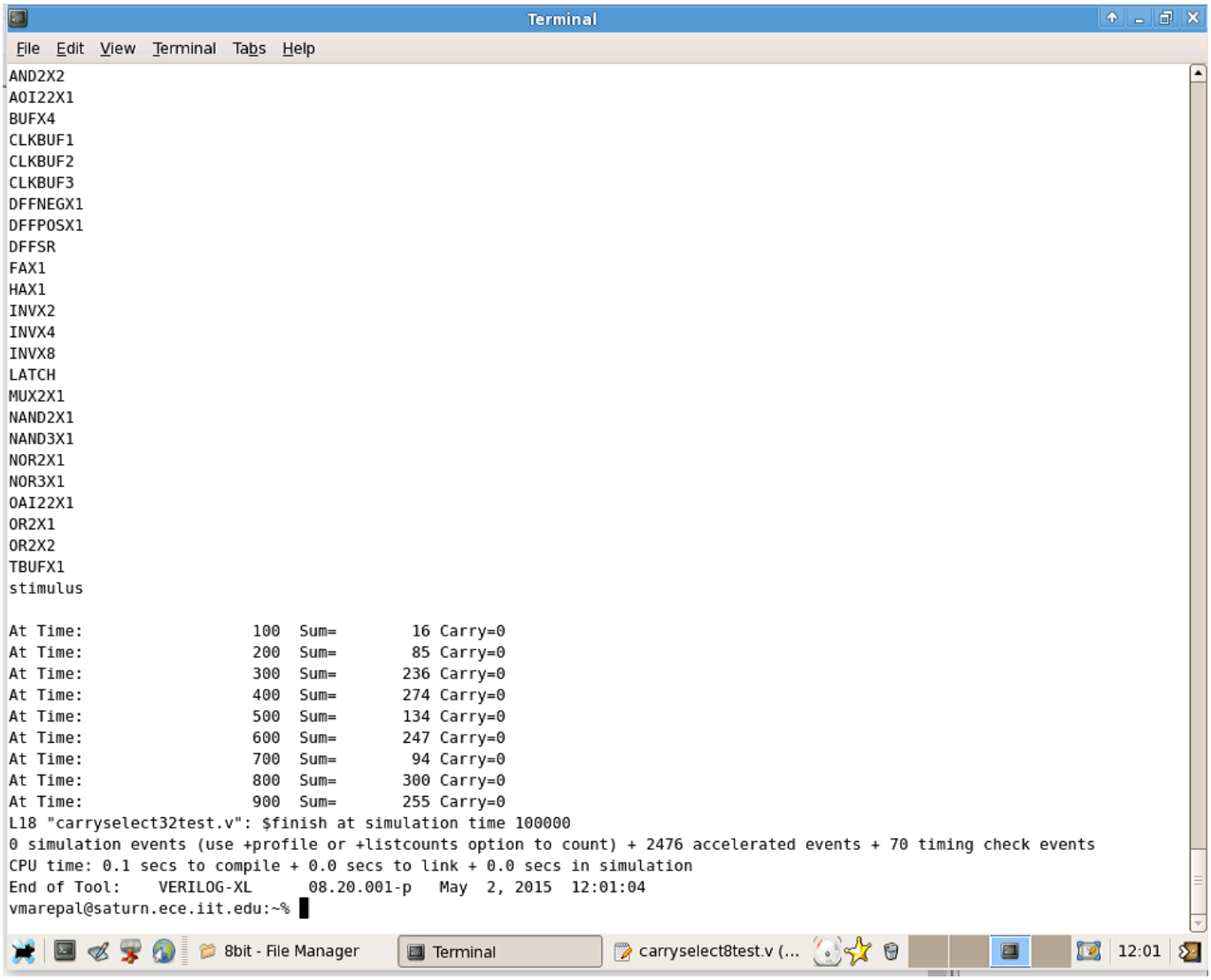

Formality (R) Console - Synopsys Inc.
File Edit View Designs Run Window Help
Verification Succeeded
Reference: r:/WORK/carryselect8
Implementation: i:/WORK/carryselect8
0. Guidance
1. Reference
2. Implementation
3. Setup
4. Match
5. Verify
6. Debug
Don't Verify Setup
Failing Points
Aborted Points
Unverified Points
Object Name
Type
Information
Verification succeeded!
OK
0
Filter:
Verify
Verify Selected Point
Failing (not equivalent) 0 0 0 0 0 0 0 0
1
Log
Errors
Warnings
History
Last Command
Formality (match)>
Ready
Shell State: match
32bit - Fil...
Terminal
carrysel...
Formalit...
Terminal
12:15

G. Functional Validation of Carry Select Adder 32-bit:

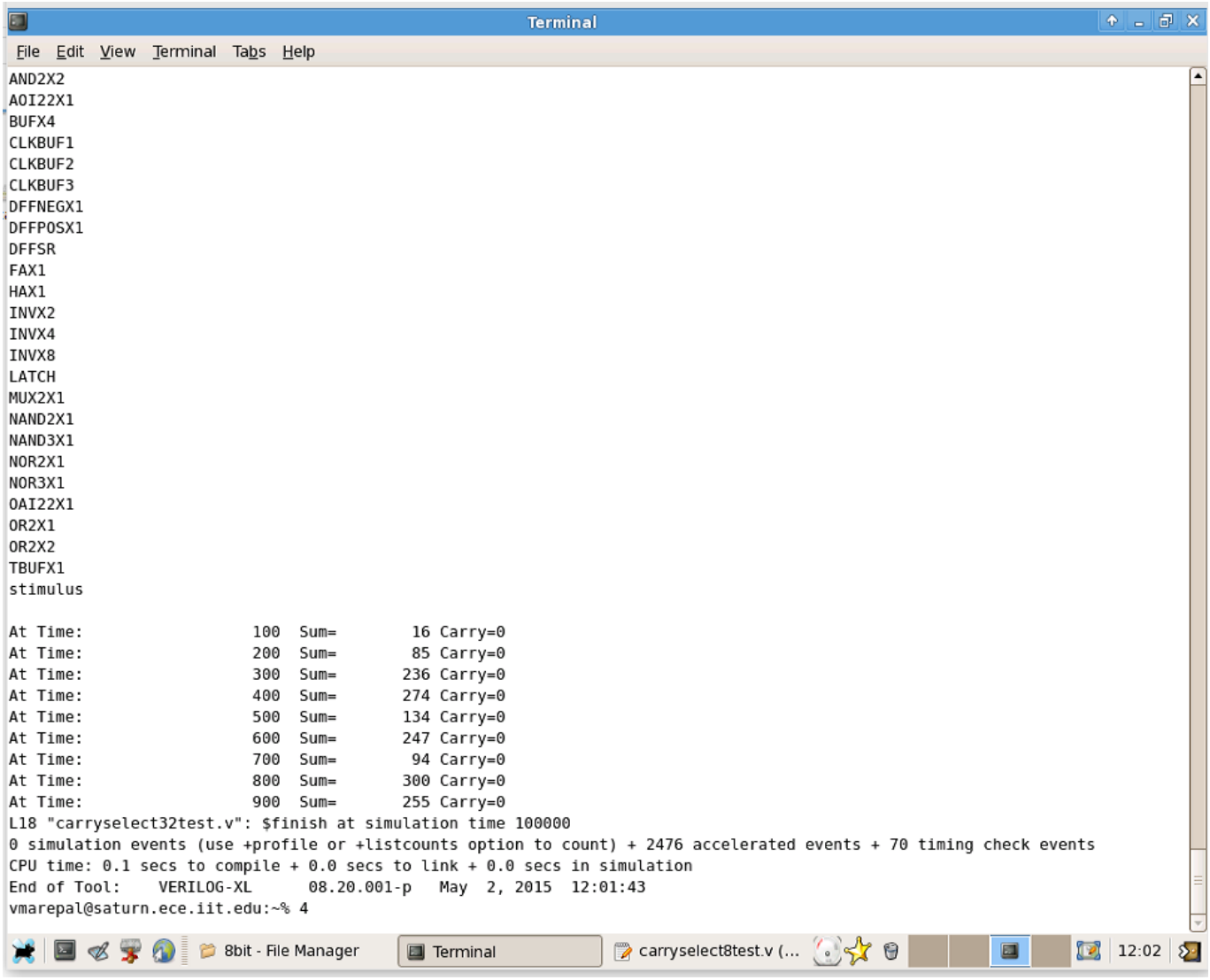
a. Verilog simulation:



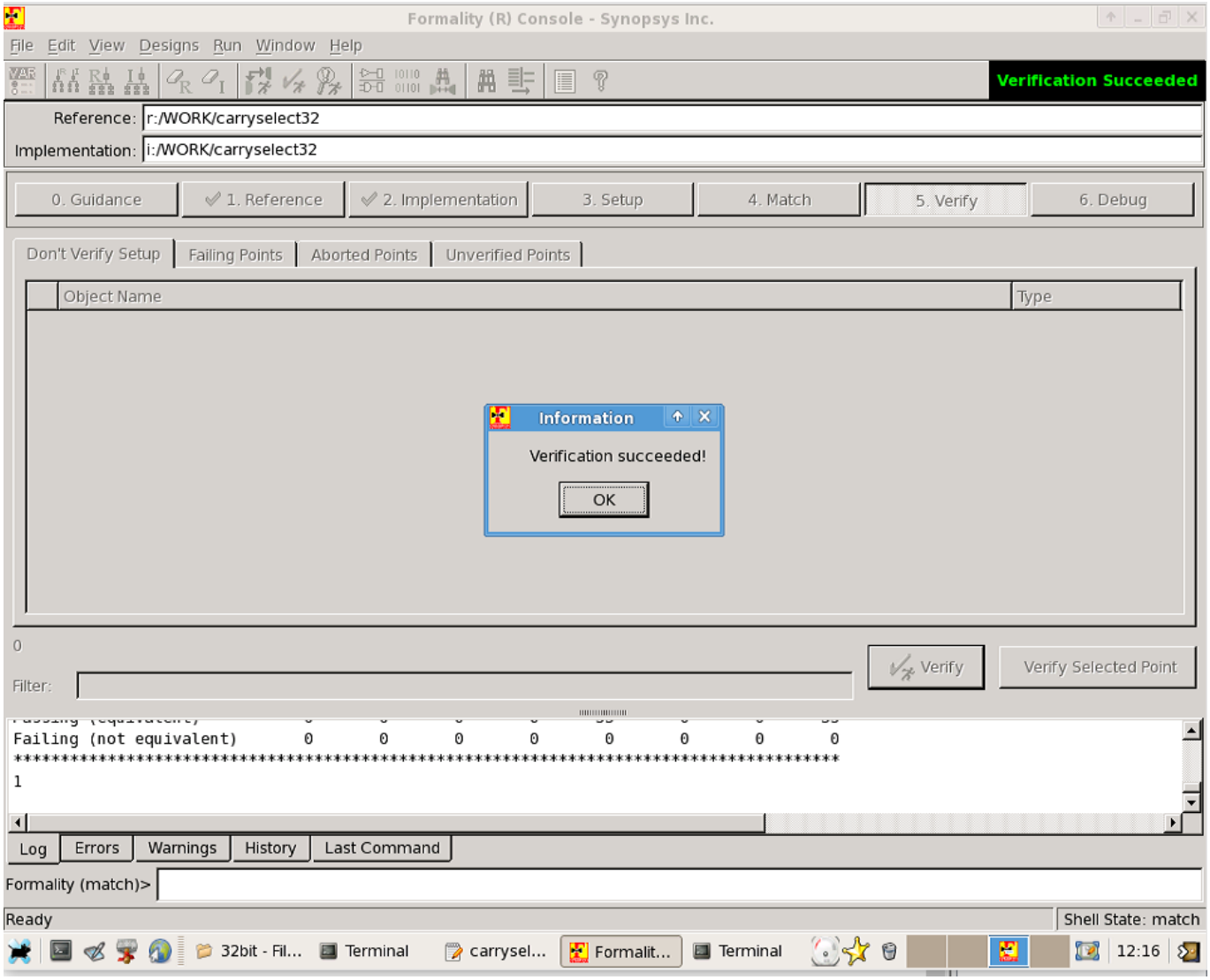
b. DC Compile:


Terminal
File Edit View Terminal Tabs Help
AND2X2
AOI22X1
BUFX4
CLKBUF1
CLKBUF2
CLKBUF3
DFFNEGX1
DFFPOSX1
DFFSR
FAX1
HAX1
INVX2
INVX4
INVX8
LATCH
MUX2X1
NAND2X1
NAND3X1
NOR2X1
NOR3X1
OAI22X1
OR2X1
OR2X2
TBUFX1
stimulus
At Time: 100 Sum= 16 Carry=0
At Time: 200 Sum= 85 Carry=0
At Time: 300 Sum= 236 Carry=0
At Time: 400 Sum= 274 Carry=0
At Time: 500 Sum= 134 Carry=0
At Time: 600 Sum= 247 Carry=0
At Time: 700 Sum= 94 Carry=0
At Time: 800 Sum= 300 Carry=0
At Time: 900 Sum= 255 Carry=0
L18 "carryselect32test.v": $finish at simulation time 100000
0 simulation events (use +profile or +listcounts option to count) + 2476 accelerated events + 70 timing check events
CPU time: 0.1 secs to compile + 0.0 secs to link + 0.0 secs in simulation
End of Tool: VERILOG-XL 08.20.001-p May 2, 2015 12:01:04
vmarepal@saturn.ece.iit.edu:~%
8bit - File Manager
Terminal
carryselect8test.v (...
12:01

c. Encounter:

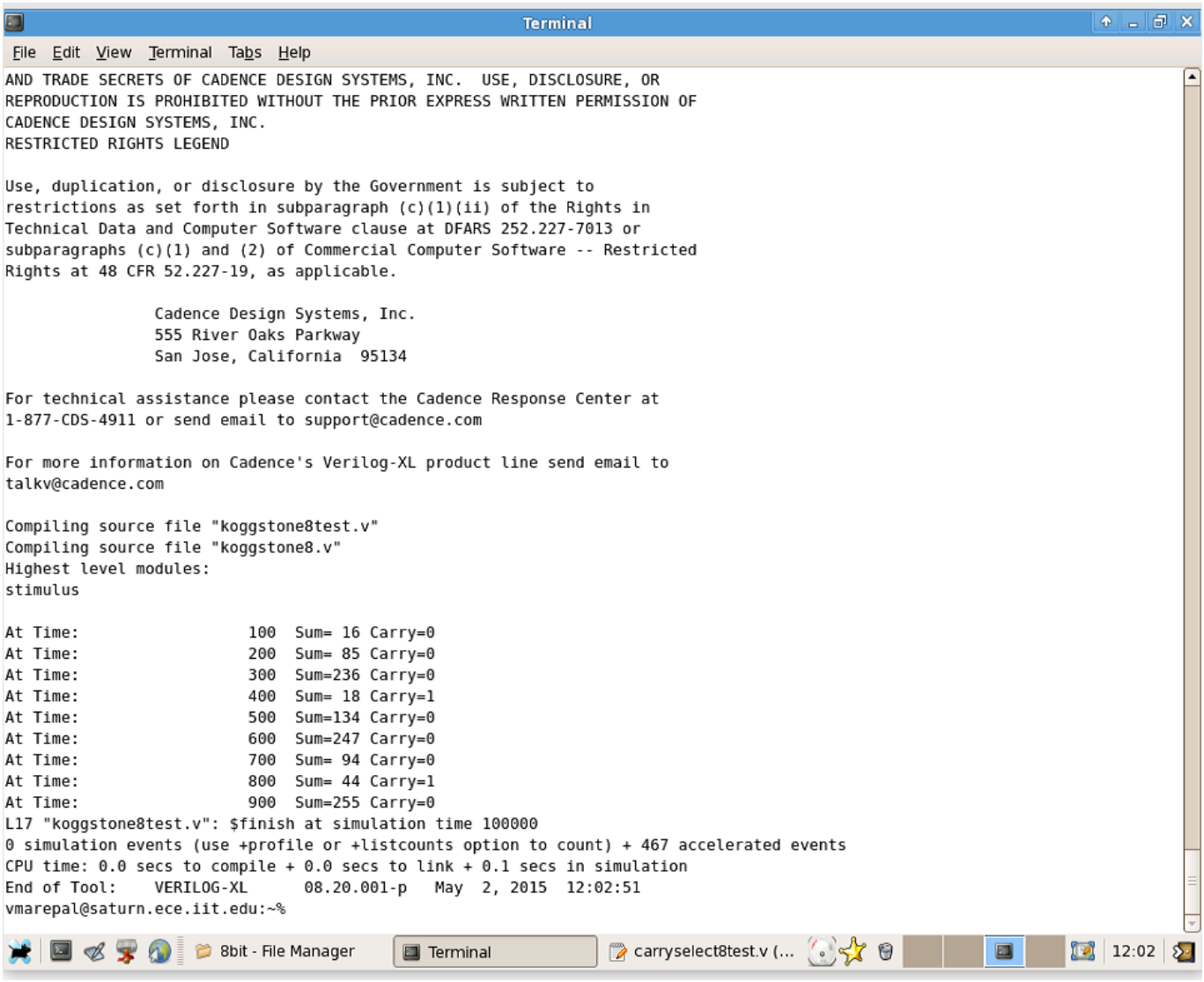

Terminal
File Edit View Terminal Tabs Help
AND2X2
AOI22X1
BUFX4
CLKBUF1
CLKBUF2
CLKBUF3
DFFNEGX1
DFFPOSX1
DFFSR
FAX1
HAX1
INVX2
INVX4
INVX8
LATCH
MUX2X1
NAND2X1
NAND3X1
NOR2X1
NOR3X1
OAI22X1
OR2X1
OR2X2
TBUFX1
stimulus
At Time: 100 Sum= 16 Carry=0
At Time: 200 Sum= 85 Carry=0
At Time: 300 Sum= 236 Carry=0
At Time: 400 Sum= 274 Carry=0
At Time: 500 Sum= 134 Carry=0
At Time: 600 Sum= 247 Carry=0
At Time: 700 Sum= 94 Carry=0
At Time: 800 Sum= 300 Carry=0
At Time: 900 Sum= 255 Carry=0
L18 "carryselect32test.v": $finish at simulation time 100000
0 simulation events (use +profile or +listcounts option to count) + 2476 accelerated events + 70 timing check events
CPU time: 0.1 secs to compile + 0.0 secs to link + 0.0 secs in simulation
End of Tool: VERILOG-XL 08.20.001-p May 2, 2015 12:01:43
vmarepal@saturn.ece.iit.edu:~% 4
8bit - File Manager
Terminal
carryselect8test.v (...
12:02

d. Formality

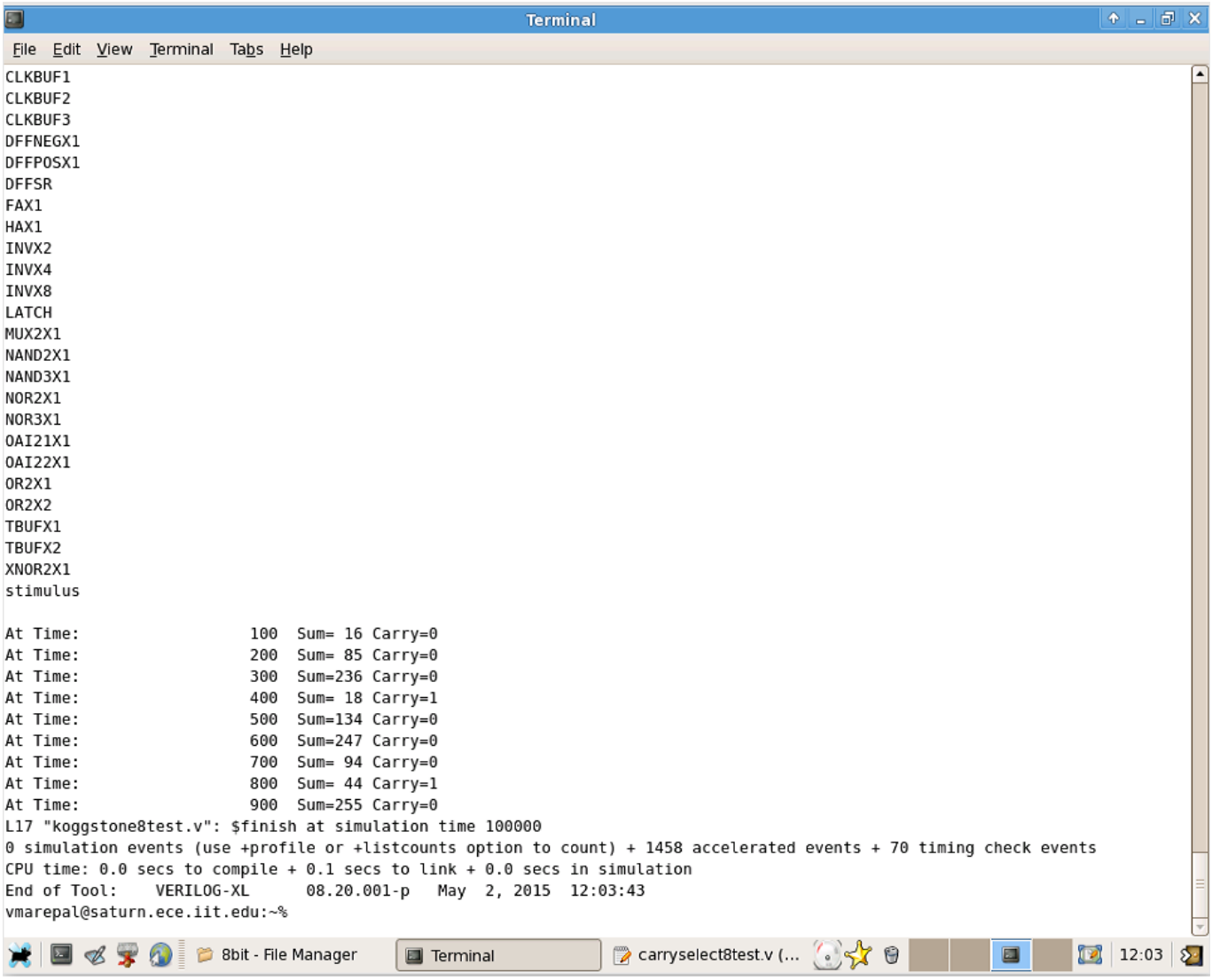

Formality (R) Console - Synopsys Inc.
File Edit View Designs Run Window Help
Verification Succeeded
Reference: r:/WORK/carryselect32
Implementation: i:/WORK/carryselect32
0. Guidance
1. Reference
2. Implementation
3. Setup
4. Match
5. Verify
6. Debug
Don't Verify Setup
Failing Points
Aborted Points
Unverified Points
Object Name
Type
Information
Verification succeeded!
OK
0
Filter:
Verify
Verify Selected Point
Failing (not equivalent) 0 0 0 0 0 0 0 0
1
Log
Errors
Warnings
History
Last Command
Formality (match)>
Ready
Shell State: match
32bit - Fil...
Terminal
carrysel...
Formalit...
Terminal
12:16

H. Functional Validation of Koggestone Adder 8-bit:

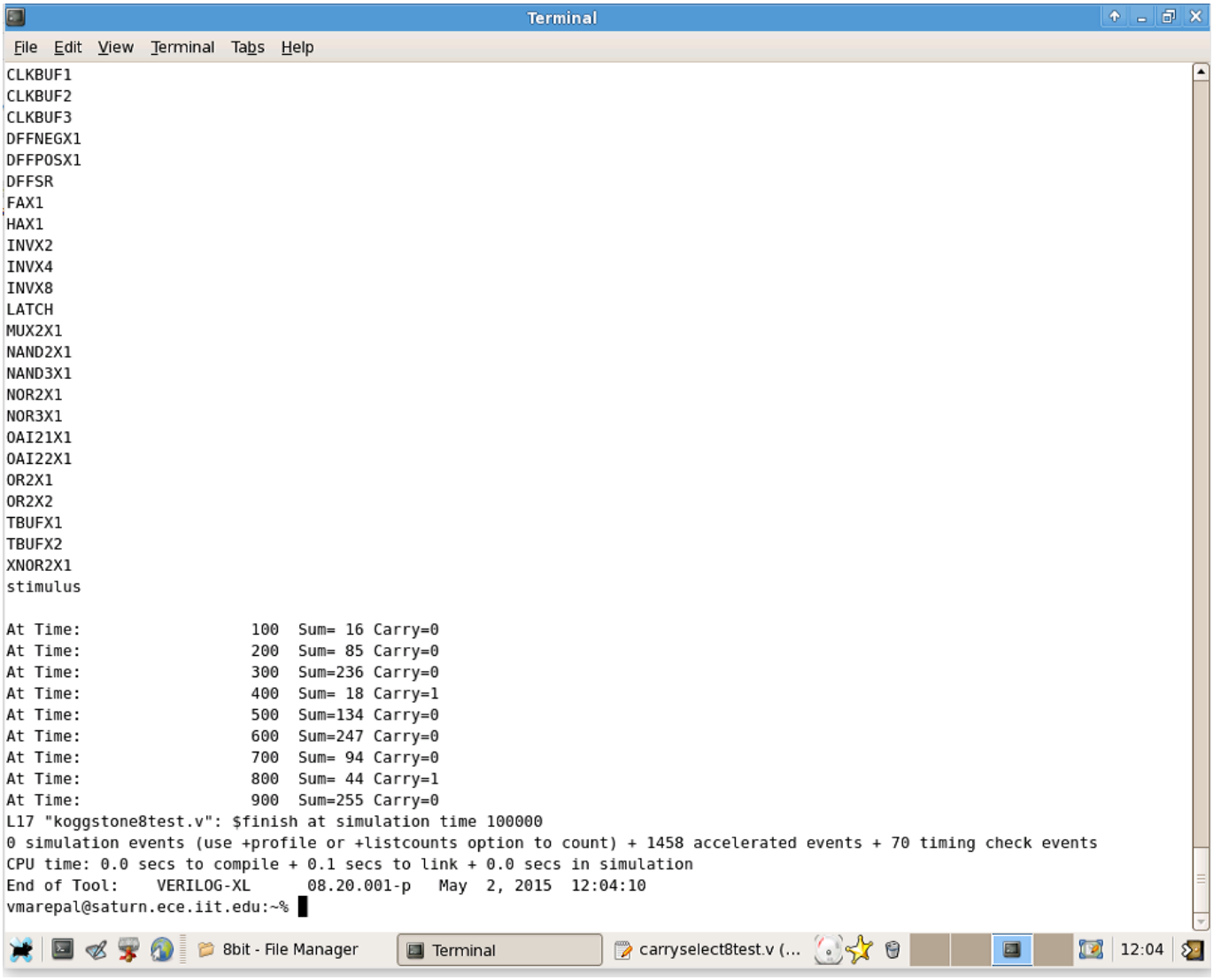
a. Verilog simulation:



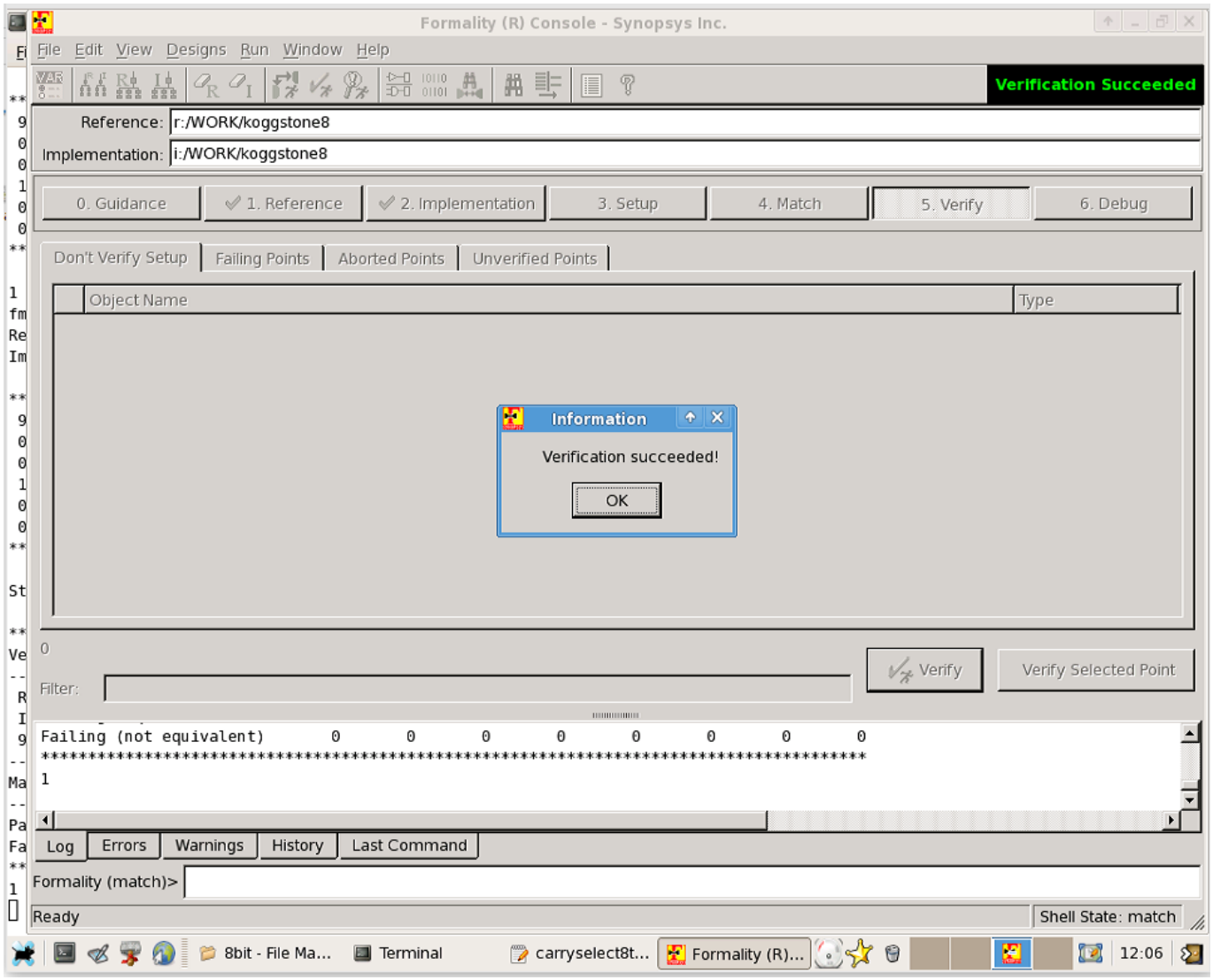
b. DC Compile:


Terminal
File Edit View Terminal Tabs Help
CLKBUF1
CLKBUF2
CLKBUF3
DFFNEGX1
DFFPOSX1
DFFSR
FAX1
HAX1
INVX2
INVX4
INVX8
LATCH
MUX2X1
NAND2X1
NAND3X1
NOR2X1
NOR3X1
OAI21X1
OAI22X1
OR2X1
OR2X2
TBUFX1
TBUFX2
XNOR2X1
stimulus
At Time: 100 Sum= 16 Carry=0
At Time: 200 Sum= 85 Carry=0
At Time: 300 Sum=236 Carry=0
At Time: 400 Sum= 18 Carry=1
At Time: 500 Sum=134 Carry=0
At Time: 600 Sum=247 Carry=0
At Time: 700 Sum= 94 Carry=0
At Time: 800 Sum= 44 Carry=1
At Time: 900 Sum=255 Carry=0
L17 "koggstone8test.v": $finish at simulation time 100000
0 simulation events (use +profile or +listcounts option to count) + 1458 accelerated events + 70 timing check events
CPU time: 0.0 secs to compile + 0.1 secs to link + 0.0 secs in simulation
End of Tool: VERILOG-XL 08.20.001-p May 2, 2015 12:03:43
vmarepal@saturn.ece.iit.edu:~%
8bit - File Manager
Terminal
carryselect8test.v (...
12:03

c. Encounter:

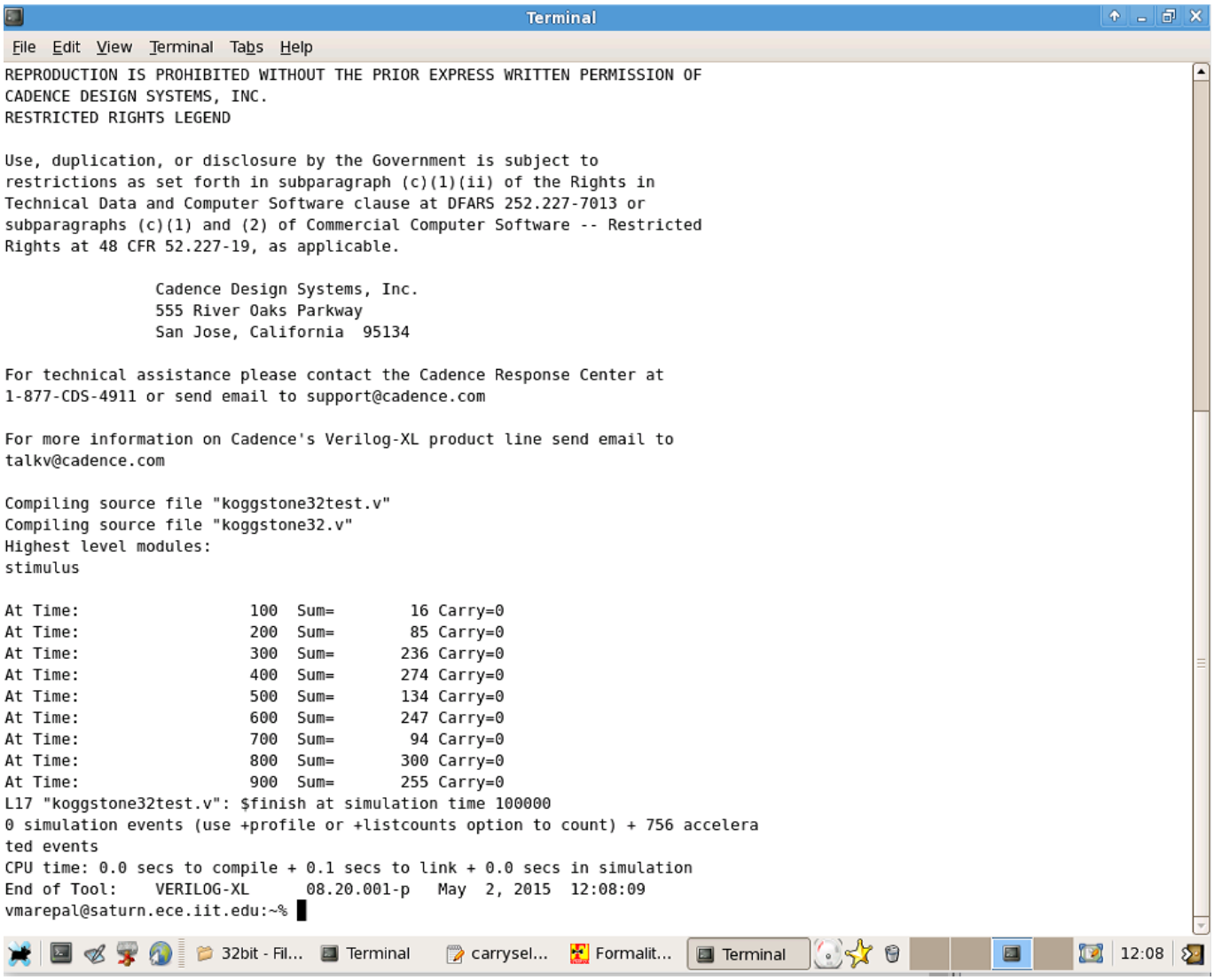

Terminal
File Edit View Terminal Tabs Help
CLKBUF1
CLKBUF2
CLKBUF3
DFFNEGX1
DFFPOSX1
DFFSR
FAX1
HAX1
INVX2
INVX4
INVX8
LATCH
MUX2X1
NAND2X1
NAND3X1
NOR2X1
NOR3X1
OAI21X1
OAI22X1
OR2X1
OR2X2
TBUFX1
TBUFX2
XNOR2X1
stimulus
At Time: 100 Sum= 16 Carry=0
At Time: 200 Sum= 85 Carry=0
At Time: 300 Sum=236 Carry=0
At Time: 400 Sum= 18 Carry=1
At Time: 500 Sum=134 Carry=0
At Time: 600 Sum=247 Carry=0
At Time: 700 Sum= 94 Carry=0
At Time: 800 Sum= 44 Carry=1
At Time: 900 Sum=255 Carry=0
L17 "koggstone8test.v": $finish at simulation time 100000
0 simulation events (use +profile or +listcounts option to count) + 1458 accelerated events + 70 timing check events
CPU time: 0.0 secs to compile + 0.1 secs to link + 0.0 secs in simulation
End of Tool: VERILOG-XL 08.20.001-p May 2, 2015 12:04:10
vmarepal@saturn.ece.iit.edu:~%
8bit - File Manager
Terminal
carryselect8test.v (...
12:04

d. Formality

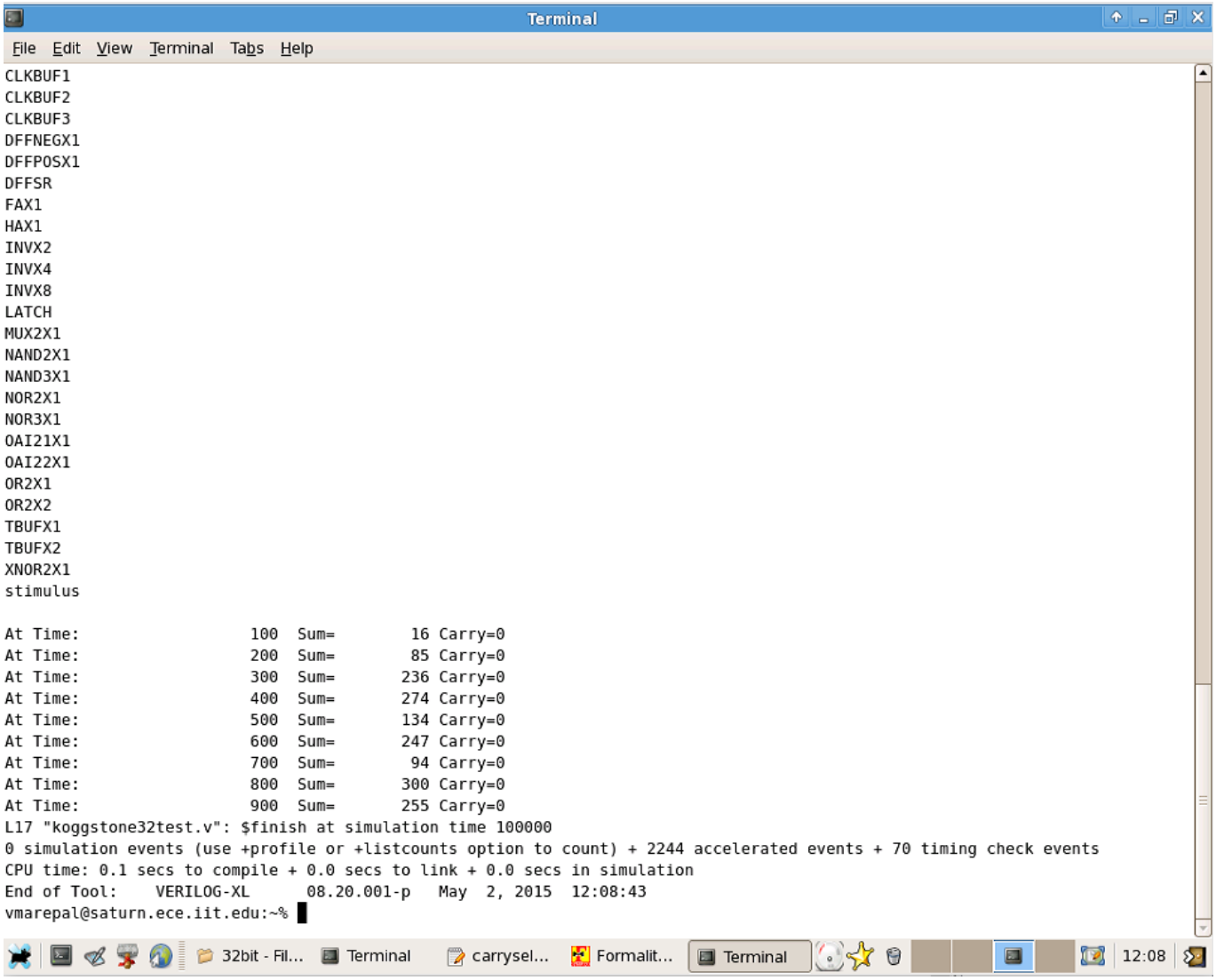

Formality (R) Console - Synopsys Inc.
File Edit View Designs Run Window Help
Verification Succeeded
Reference: r:/WORK/koggstone8
Implementation: i:/WORK/koggstone8
0. Guidance
1. Reference
2. Implementation
3. Setup
4. Match
5. Verify
6. Debug
Don't Verify Setup
Failing Points
Aborted Points
Unverified Points
Object Name
Type
Information
Verification succeeded!
OK
Filter:
Verify
Verify Selected Point
Failing (not equivalent) 0 0 0 0 0 0 0 0
Log
Errors
Warnings
History
Last Command
Formality (match)>
Ready
Shell State: match
8bit - File Ma...
Terminal
carryselect8t...
Formality (R)...
12:06

I. Functional Validation of Koggestone Adder 32-bit:

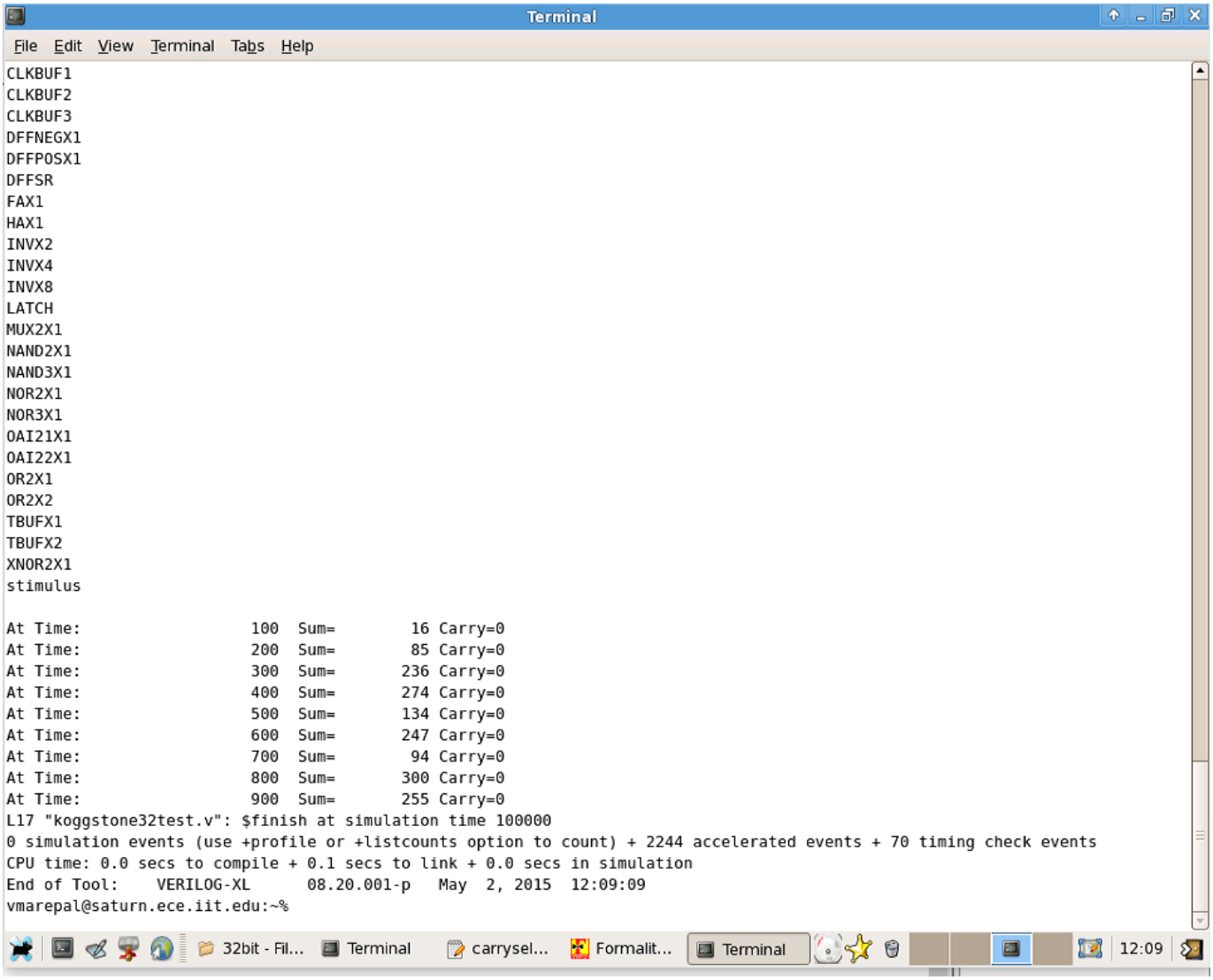
a. Verilog simulation:



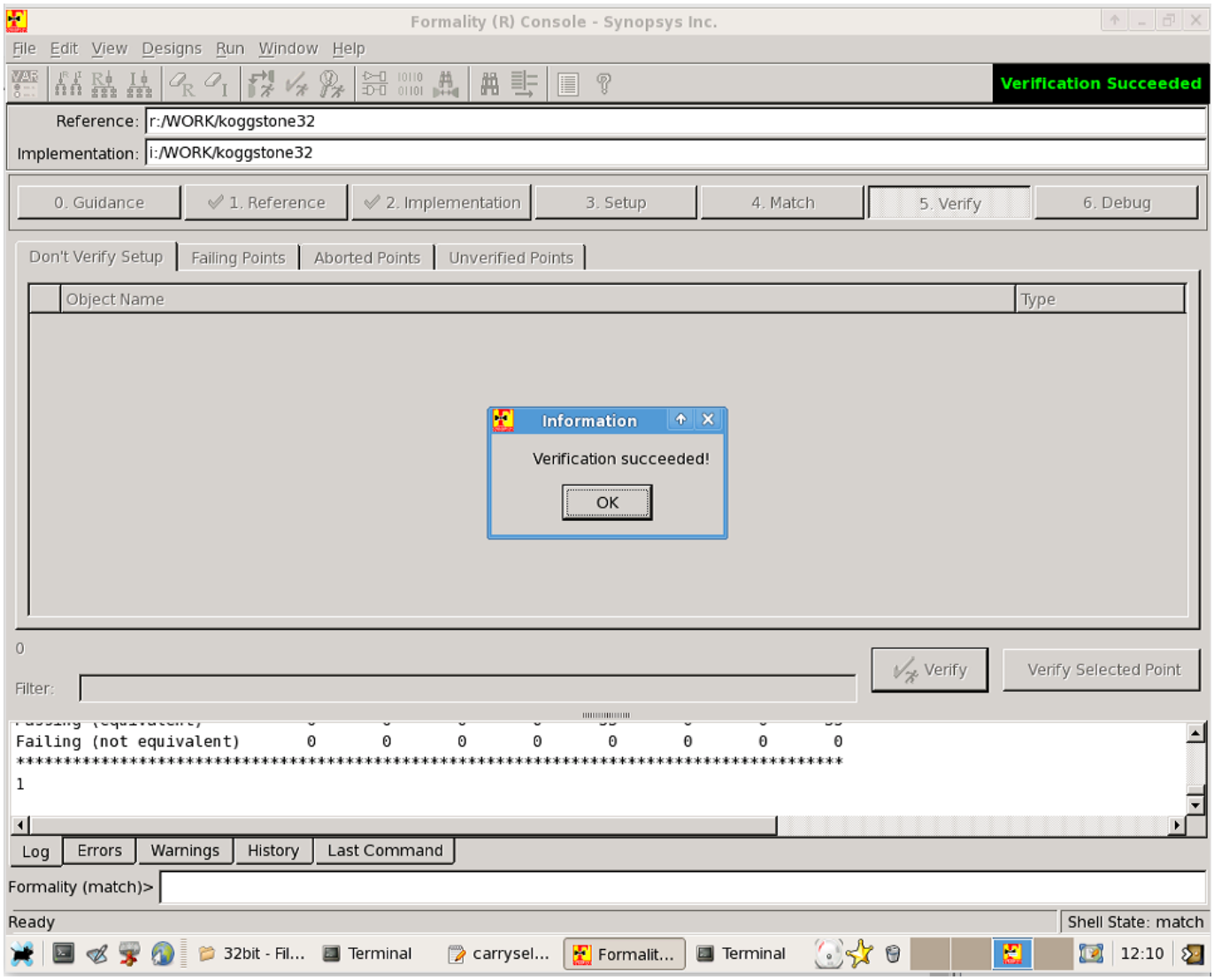
b. DC Compile:


Terminal
File Edit View Terminal Tabs Help
CLKBUF1
CLKBUF2
CLKBUF3
DFFNEGX1
DFFPOSX1
DFFSR
FAX1
HAX1
INVX2
INVX4
INVX8
LATCH
MUX2X1
NAND2X1
NAND3X1
NOR2X1
NOR3X1
OAI21X1
OAI22X1
OR2X1
OR2X2
TBUFX1
TBUFX2
XNOR2X1
stimulus
At Time: 100 Sum= 16 Carry=0
At Time: 200 Sum= 85 Carry=0
At Time: 300 Sum= 236 Carry=0
At Time: 400 Sum= 274 Carry=0
At Time: 500 Sum= 134 Carry=0
At Time: 600 Sum= 247 Carry=0
At Time: 700 Sum= 94 Carry=0
At Time: 800 Sum= 300 Carry=0
At Time: 900 Sum= 255 Carry=0
L17 "koggstone32test.v": $finish at simulation time 100000
0 simulation events (use +profile or +listcounts option to count) + 2244 accelerated events + 70 timing check events
CPU time: 0.1 secs to compile + 0.0 secs to link + 0.0 secs in simulation
End of Tool: VERILOG-XL 08.20.001-p May 2, 2015 12:08:43
vmarepal@saturn.ece.iit.edu:~%
32bit - Fil...
Terminal
carrysel...
Formalit...
Terminal
12:08

c. Encounter:

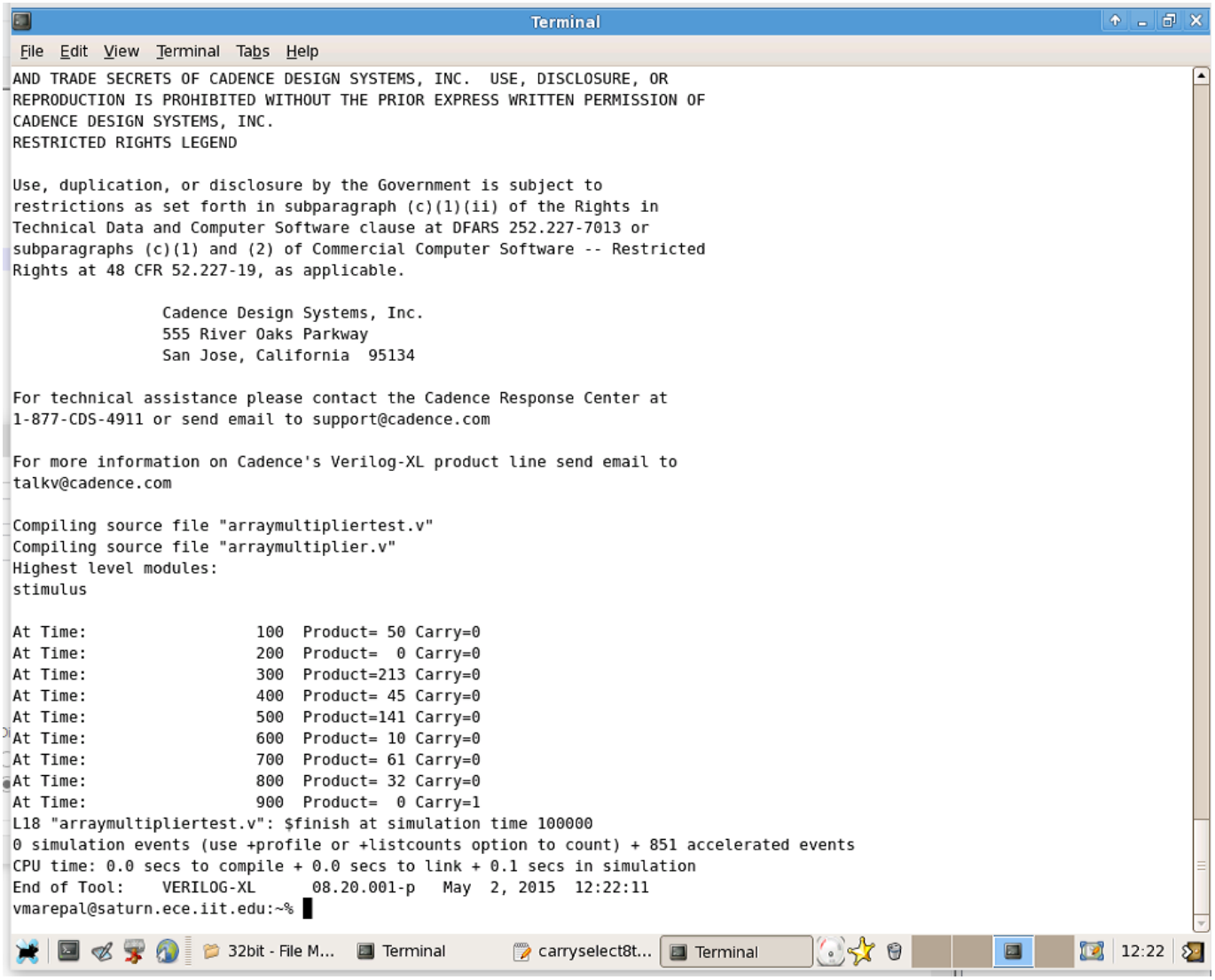

Terminal
File Edit View Terminal Tabs Help
CLKBUF1
CLKBUF2
CLKBUF3
DFFNEGX1
DFFPOSX1
DFFSR
FAX1
HAX1
INVX2
INVX4
INVX8
LATCH
MUX2X1
NAND2X1
NAND3X1
NOR2X1
NOR3X1
OAI21X1
OAI22X1
OR2X1
OR2X2
TBUFX1
TBUFX2
XNOR2X1
stimulus
At Time: 100 Sum= 16 Carry=0
At Time: 200 Sum= 85 Carry=0
At Time: 300 Sum= 236 Carry=0
At Time: 400 Sum= 274 Carry=0
At Time: 500 Sum= 134 Carry=0
At Time: 600 Sum= 247 Carry=0
At Time: 700 Sum= 94 Carry=0
At Time: 800 Sum= 300 Carry=0
At Time: 900 Sum= 255 Carry=0
L17 "koggstone32test.v": $finish at simulation time 100000
0 simulation events (use +profile or +listcounts option to count) + 2244 accelerated events + 70 timing check events
CPU time: 0.0 secs to compile + 0.1 secs to link + 0.0 secs in simulation
End of Tool: VERILOG-XL 08.20.001-p May 2, 2015 12:09:09
vmarepal@saturn.ece.iit.edu:~%
32bit - Fil...
Terminal
carrysel...
Formalit...
Terminal
12:09

d. Formality

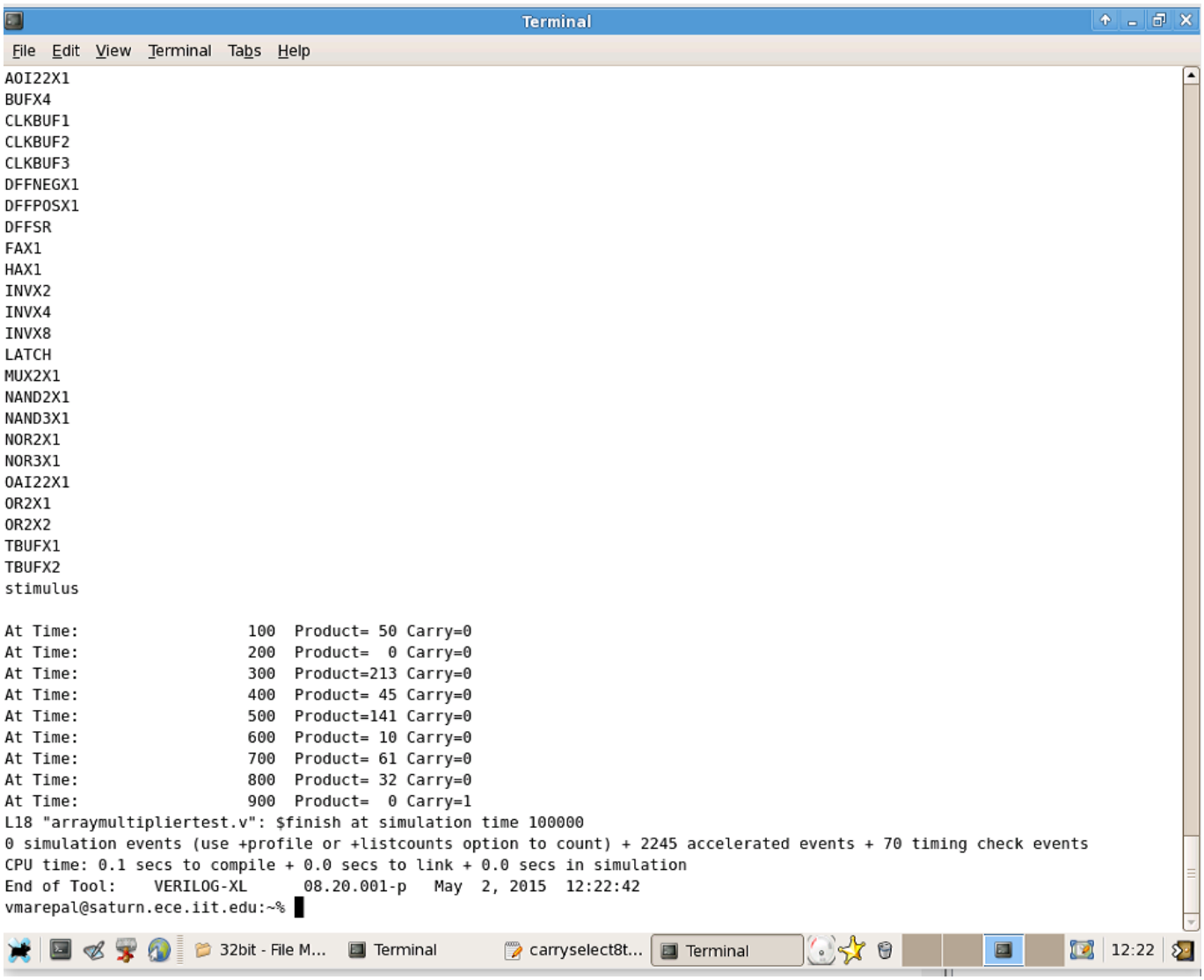

Formality (R) Console - Synopsys Inc.
File Edit View Designs Run Window Help
Verification Succeeded
Reference: r:/WORK/koggstone32
Implementation: i:/WORK/koggstone32
0. Guidance
1. Reference
2. Implementation
3. Setup
4. Match
5. Verify
6. Debug
Don't Verify Setup
Failing Points
Aborted Points
Unverified Points
Object Name
Type
Information
Verification succeeded!
OK
0
Filter:
Verify
Verify Selected Point
Failing (not equivalent) 0 0 0 0 0 0 0 0
1
Log
Errors
Warnings
History
Last Command
Formality (match)>
Ready
Shell State: match
32bit - Fil...
Terminal
carrysel...
Formalit...
Terminal
12:10

J. Functional Validation of ArrayMultiplier:

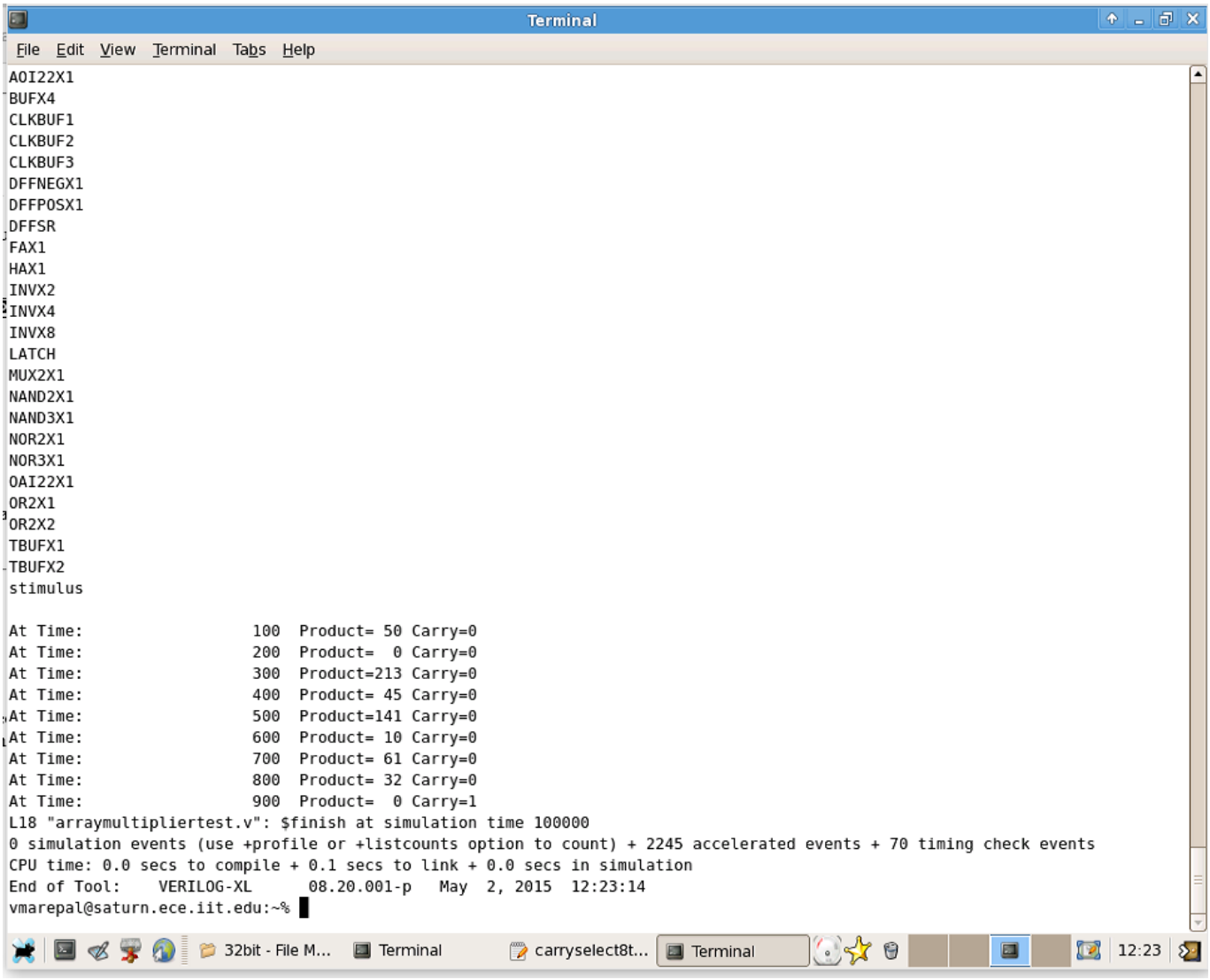
a. Verilog simulation:



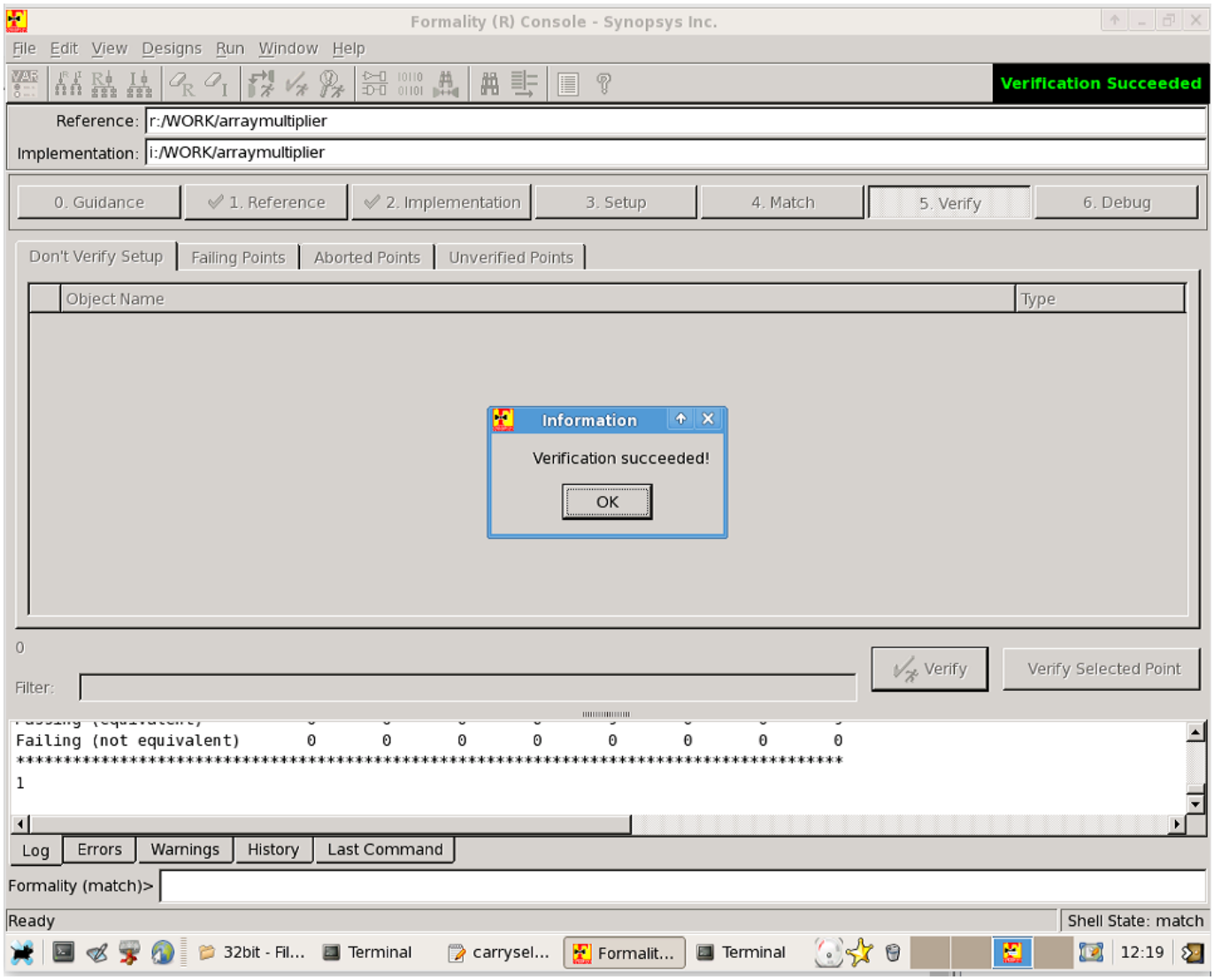
b. DC Compile:


Terminal
File Edit View Terminal Tabs Help
AOI22X1
BUFX4
CLKBUF1
CLKBUF2
CLKBUF3
DFFNEGX1
DFFPOSX1
DFFSR
FAX1
HAX1
INVX2
INVX4
INVX8
LATCH
MUX2X1
NAND2X1
NAND3X1
NOR2X1
NOR3X1
OAI22X1
OR2X1
OR2X2
TBUFX1
TBUFX2
stimulus
At Time: 100 Product= 50 Carry=0
At Time: 200 Product= 0 Carry=0
At Time: 300 Product=213 Carry=0
At Time: 400 Product= 45 Carry=0
At Time: 500 Product=141 Carry=0
At Time: 600 Product= 10 Carry=0
At Time: 700 Product= 61 Carry=0
At Time: 800 Product= 32 Carry=0
At Time: 900 Product= 0 Carry=1
L18 "arraymultipliertest.v": $finish at simulation time 100000
0 simulation events (use +profile or +listcounts option to count) + 2245 accelerated events + 70 timing check events
CPU time: 0.1 secs to compile + 0.0 secs to link + 0.0 secs in simulation
End of Tool: VERILOG-XL 08.20.001-p May 2, 2015 12:22:42
vmarepal@saturn.ece.iit.edu:~%
32bit - File M...
Terminal
carryselect8t...
Terminal
12:22

c. Encounter:


Terminal
File Edit View Terminal Tabs Help
AOI22X1
BUFX4
CLKBUF1
CLKBUF2
CLKBUF3
DFFNEGX1
DFFPOSX1
DFFSR
FAX1
HAX1
INVX2
INVX4
INVX8
LATCH
MUX2X1
NAND2X1
NAND3X1
NOR2X1
NOR3X1
OAI22X1
OR2X1
OR2X2
TBUFX1
TBUFX2
stimulus
At Time: 100 Product= 50 Carry=0
At Time: 200 Product= 0 Carry=0
At Time: 300 Product=213 Carry=0
At Time: 400 Product= 45 Carry=0
At Time: 500 Product=141 Carry=0
At Time: 600 Product= 10 Carry=0
At Time: 700 Product= 61 Carry=0
At Time: 800 Product= 32 Carry=0
At Time: 900 Product= 0 Carry=1
L18 "arraymultipliertest.v": $finish at simulation time 100000
0 simulation events (use +profile or +listcounts option to count) + 2245 accelerated events + 70 timing check events
CPU time: 0.0 secs to compile + 0.1 secs to link + 0.0 secs in simulation
End of Tool: VERILOG-XL 08.20.001-p May 2, 2015 12:23:14
vmarepal@saturn.ece.iit.edu:~%
32bit - File M...
Terminal
carryselect8t...
Terminal
12:23

d. Formality



REFERENCES

[1]. http://www.ece.msstate.edu/~reese/EE4743/vhdlcomb/sld027.htm.
[2]. Project Document -ece429-prj(2).pdf.
[3]. CMOS VLSI Design by Neil H.E.Weste, David Harris.
[4]. http://www.engr.sjsu.edu/dparent/ee224/adder_verification.pdf