8 Bit CISC Description using Verilog & Synthesis

By
K.Narayanan
Spring 2002

ECE 128 Verilog Project
VLSI Testing Taught by:
Prof. M .E. Zaghloul
ECE Department
George Washington University

Objective

- Understand the role of EDA in Rapid prototype development.
- Use Cadence tools for Description, Simulation.
- Why HDL ?
- Virtual Testing and Simulation,Fault modeling possible from HDL.

Objective (contd.)

- Synthesize the modules and understand the logic or layout automation from hardware description.
- Incorporate Design for Testing principles in the design.

Extracted Requirements

- The CPU should provide suitable interface to the provided memory unit.
- The CPU should perform, logical,arithmetic and shifting operations-conforming to the provided instruction set.
- Modules identified for the CPU - ALU, PC, IO & CU. Should pass the test-benches
- Integrated CHIP should pass CPU tests.

Implementation - ALU

```
module my_alu (result,zero,cout,inh,ALU_code,a_8,b_8,c);
input  [3:0] ALU_code;
input        inh,c;
input  [7:0] a_8,b_8;
output [7:0] result;
output       zero,cout;
// all inputs required in sensitivity list  ??????
always @(inh or ALU_code or a_8 or b_8 or c) begin
casex ({inh,ALU_code})

{`INH_0,`ADD} :              {result} <= {a_8 + b_8 + c};
{`INH_0,`SUB}  :             {result} <= {a_8 - b_8 - c };
{`INH_0,`AND} :              result <= a_8 & b_8;
{`INH_0,`OR}    :            result <= a_8 | b_8;

{`INH_0,`XOR} :              result <= a_8 ^ b_8;

{`INH_0,`B_LD}:              result <=  b_8;

{`INH_0,`A1NOP}:             result <= a_8;
{`INH_0,`A2NOP}:             result <= a_8;
{`INH_1,`RLFT}   :           {result} <= {a_8[6:0],c};
{`INH_1,`RRGT}:              {result} <= {c,a_8[7:1]};
{`INH_1,`AS_LFT}:            {result} <= {a_8[6:0],1'b0};
{`INH_1,`AS_RGT}:            {result} <= {a_8[7],a_8[7:1]};
{`INH_1,`INCR}:              {result} <= {a_8 + 1'b1};
{`INH_1,`DECR}:              {result} <= {a_8 + 8'hff}  ;
{`INH_1,`ZERO}:              result <= 0;
{`INH_1,`A3NOP}:             result <= a_8;
{`INH_1,`ONESC}:             result <= 8'hff^a_8 ;
{`INH_1,`TWOSC}:             {result} <= {(8'hff^a_8) + 1'b1} ;
{`INH_1,`A_C_0}:             result <= a_8 ;
{`INH_1,`A_C_1}:             result <= a_8 ;
endcase
end // always

always @(result or a_8 or b_8 or c) begin //zero and carry calc
  if(result == 8'h00) zero <= 1'b1;
  else  zero <= 1'b0;
casex ({inh,ALU_code})
{`INH_0,`ADD}: cout <= ((a_8[7]&b_8[7]) |
      ((1'b1^result[7])&b_8[7]) | (a_8[7]&(1'b1^result[7])));
{`INH_0,`SUB}: cout <= ((a_8[7]&b_8[7]) |
      ((1'b1^result[7])&b_8[7]) | (a_8[7]&(1'b1^result[7])));
{`INH_0,`AND}:               cout <=c;
{`INH_0,`OR}:                cout <=c;
{`INH_0,`XOR}:               cout <=c;
{`INH_0,`B_LD}:              cout <=c;
{`INH_0,`A1NOP}:             cout <=c;
{`INH_0,`A2NOP}:             cout <=c;
{`INH_1,`RLFT}:              cout <= a_8[7];
{`INH_1,`RRGT}:              cout <= a_8[0];
{`INH_1,`AS_LFT}:            cout<= a_8[7];
{`INH_1,`AS_RGT}:            cout<= a_8[0];
{`INH_1,`INCR}:              cout   <=
      ((a_8[7])&(a_8[6])&(a_8[5])&(a_8[4])&(a_8[3])&(a_8[2])
      &(a_8[1])&(a_8[0]));
{`INH_1,`DECR}:              cout   <=
      ((!a_8[7])&(!a_8[6])&(!a_8[5])&(!a_8[4])&(!a_8[3])&(!a_8
      [2])&(!a_8[1])&(!a_8[0]));
{`INH_1,`ZERO}:              cout <= 0;
{`INH_1,`A3NOP}:             cout <= c;
{`INH_1,`ONESC}:             cout <=c;
{`INH_1,`TWOSC}:             cout   <=
      ((result[7])|(result[6])|(result[5])|(result[4])|(result[3])|(result[
      2])|(result[1])|(result[0]));
{`INH_1,`A_C_0}:             cout <= 1'b0;
{`INH_1,`A_C_1}:             cout <= 1'b1;
endcase
end
endmodule
```

Two Versions of Control Unit Which is better ?

```
module cu_mod (pc_l,a_l,b_l,addr_l,data_out,addr_sel_l,irh_l,irl_l,state_l,rw,nxt_st,state,E,clk,clkby2);
input [1:0] nxt_st,state;
input rw,clk,clkby2;
output pc_l,a_l,b_l,addr_l,data_out,addr_sel_l,irh_l,irl_l,E,state_l;
wire  pc_l,a_l,b_l,addr_l,data_out,addr_sel_l,irh_l,irl_l,E,state_l;

assign state_l =    ({clk,clkby2} == 2'b00) ? 1'b1:({clk,clkby2} == 2'b01)? 1'b0:state_l;
assign irh_l   =    ({clk,clkby2} == 2'b00) ? 1'b0:(({clk,clkby2} == 2'b01)&&(state == 2'b00))? 1'b1:irh_l;
assign irl_l   =    ({clk,clkby2} == 2'b00) ? 1'b0:(({clk,clkby2} == 2'b01)&&(state == 2'b01))? 1'b1: irl_l ;
assign addr_sel_l = ({clk,clkby2} == 2'b00) ?1'b0:({clk,clkby2} == 2'b11) ? 1'b1: addr_sel_l;
assign pc_l    =    ({clk,clkby2} == 2'b00) ?1'b0:({clk,clkby2} == 2'b11) ? 1'b1: pc_l;
assign a_l     =    ({clk,clkby2} == 2'b00) ?1'b0:(({clk,clkby2} == 2'b11)&& (nxt_st == 2'b00)) ? 1'b1:a_l;
assign b_l     =    ({clk,clkby2} == 2'b00) ?1'b0:({clk,clkby2} == 2'b01) ? 1'b1:b_l;
assign addr_l  =    ({clk,clkby2} == 2'b00) ?1'b1:({clk,clkby2} == 2'b01)? 1'b0:addr_l;
assign data_out=    ({clk,clkby2} == 2'b00) ?1'b0:(({clk,clkby2} == 2'b10) && (rw == 1'b0))? 1'b1: data_out;
assign E       =    ({clk,clkby2} == 2'b00) ?1'b0:(({clk,clkby2} == 2'b10) && (rw == 1'b1))? 1'b1:(({clk,clkby2} == 2'b01) && (rw
       == 1'b1)) ? 1'b1: E;
endmodule
```

Version 2 Control Unit

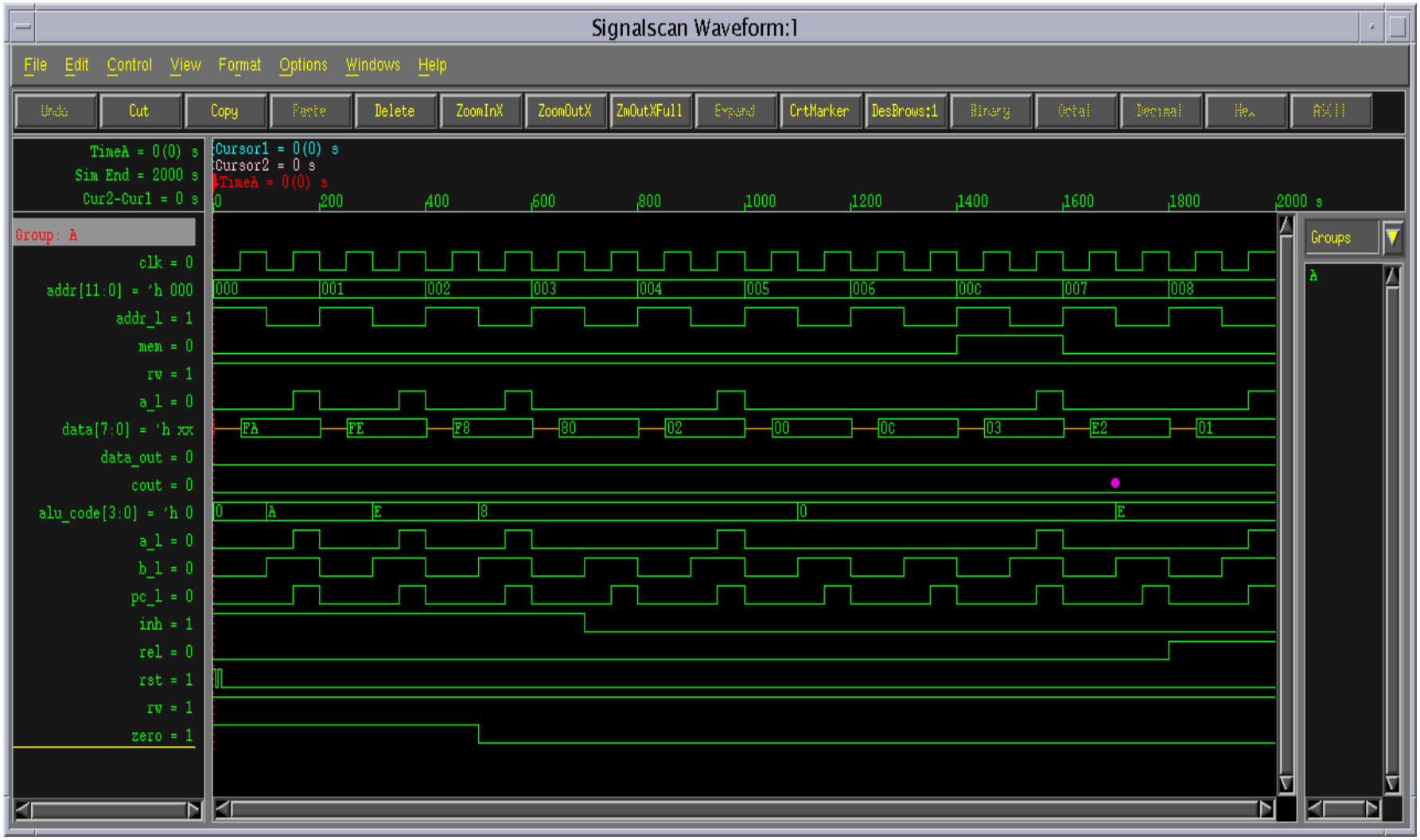
```
module cu_mod
        (pc_l,a_l,b_l,addr_l,data_out,addr_sel_l,irh_l,irl_l,E,state_l,r
        w,nxt_st,state,clk,clkby2,rst);
input [1:0] nxt_st,state;
input       rw,clk,clkby2,rst;
output    pc_l, a_l,b_l,addr_l,data_out,addr_sel_l,
                                    irh_l,irl_l,E,state_l;
reg   [1:0]   phase;

always @ (posedge rst)
 phase <= 2'b11;


always @( clk or  clkby2  ) begin

case ({clk,clkby2})

2'b00: begin //$display("phase 00");
// generation of latches in {clk,clkby2} 00 state

if(phase == 2'b11) begin
state_l         <= !clkby2;
irh_l           <=  clkby2;
irl_l           <=  clkby2;
addr_sel_l      <=  clk;
pc_l            <=  clk;
a_l             <=  clk;
b_l             <=  clkby2;
addr_l          <= !clkby2;
data_out        <=  clk;
E               <=  clk;
phase           <=  2'b10;
end
end

2'b10: begin //$display("phase 10");
// generation of latches in {clk,clkby2} 10 state

if (phase == 2'b10); begin
data_out        <=  (!rw)?clk:data_out;
E               <=  (rw)?clk:E;
phase           <= 2'b01;

end
end

2'b01: begin //$display("phase 01");
// generation of latches in {clk,clkby2} 01 state
if(phase == 2'b01) begin
state_l         <= !clkby2;
irh_l           <=  (state == 2'b00)?clkby2:!clkby2;
irl_l           <=  (state == 2'b01)?clkby2:!clkby2;
b_l             <=  clkby2;
addr_l          <= !clkby2;
E               <=  (rw)?!clk:E;
phase           <= 2'b00;
end
end

2'b11: begin
// generation of latches in {clk,clkby2} 11 state
if(phase == 2'b00) begin
addr_sel_l      <=  clk;
pc_l            <=  clk;
a_l             <=  (nxt_st==2'b00)?clk:!clk;

phase           <=  2'b11;
end
end
endcase
end                                                     // always
endmodule
```

Sample Simulation CPU_test

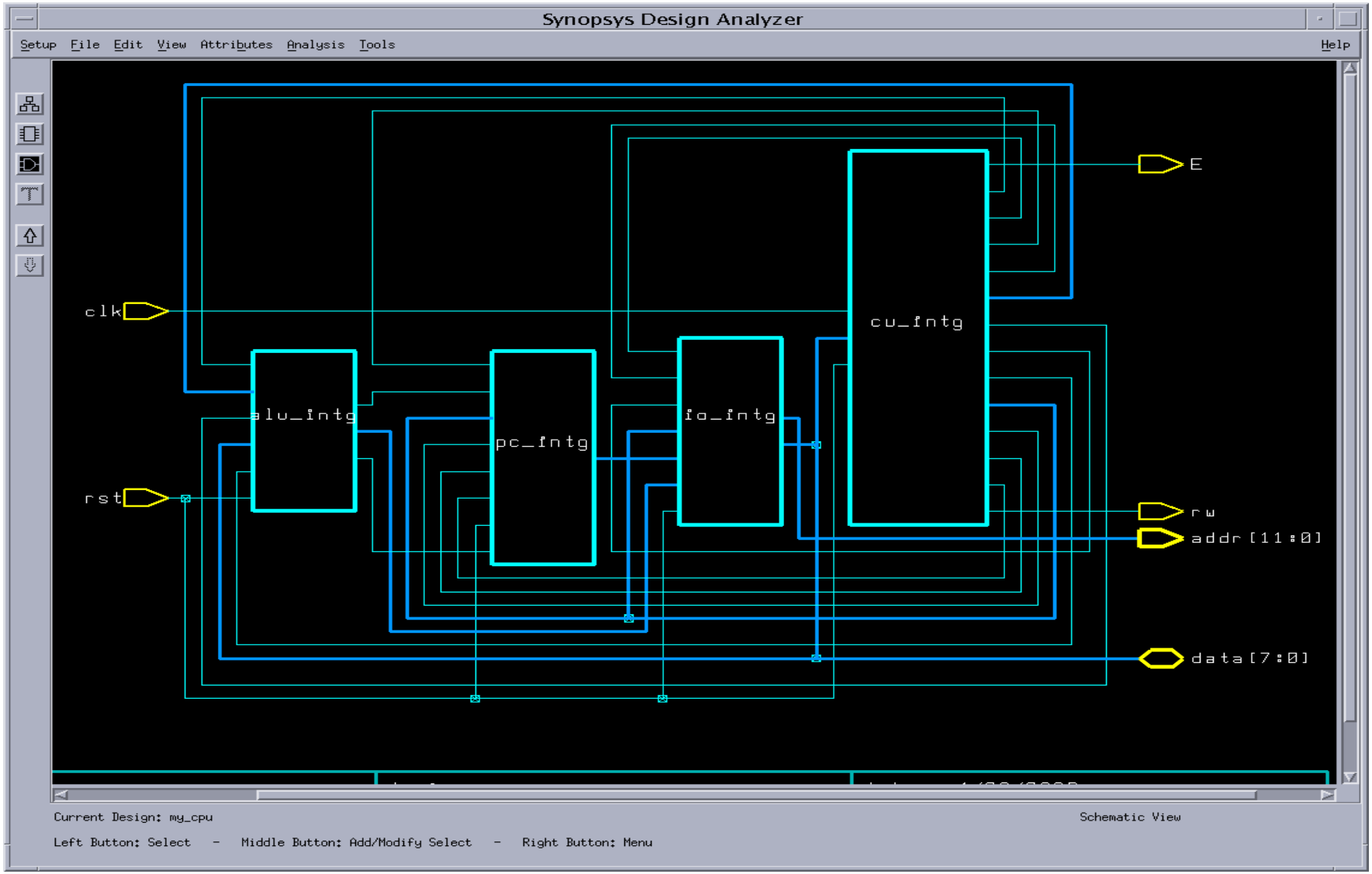
```
Compiling source file "myalu.h"
Compiling source file "a_reg.v"
Compiling source file "b_reg.v"
Compiling source file "c_reg.v"
Compiling source file "my_alu.v"
Compiling included source file "myalu.h"
Continuing compilation of source file "my_alu.v"
Compiling source file "alu_intg.v"
Compiling source file "bra.h"
Compiling source file "bra_gen.v"
Compiling included source file "bra.h"
Continuing compilation of source file "bra_gen.v"
Compiling source file "next_pc_gen.v"
Compiling source file "pc_reg.v"
Compiling source file "pc_intg.v"
Compiling source file "io_areg.v"
Compiling source file "io_asel.v"
Compiling source file "io_tri.v"
Compiling source file "io_intg.v"
Compiling source file "a_sel_reg.v"
Compiling source file "clkby2.v"
Compiling source file "cu_mod.v"
Compiling source file "decode.v"
Compiling source file "invert_rst.v"
Compiling source file "irh_reg.v"
Compiling source file "irl_reg.v"
Compiling source file "ir_reg.v"
Compiling source file "next_state.v"
Compiling source file "state_reg.v"
Compiling source file "cu_intg.v"
Compiling source file "my_cpu.v"
Compiling source file "clock_gen.v"
Compiling source file "memory.v"
Compiling source file "cpu_test.v"
Highest level modules:
```

```
cpu_test

MEMORY: LOADING FILE cpu_test.hex
Cycle:                1  Addr:000  Data:fa  RW:1
Cycle:                3  Addr:001  Data:fe  RW:1
Cycle:                5  Addr:002  Data:f8  RW:1
Cycle:                7  Addr:003  Data:80  RW:1
Cycle:                9  Addr:004  Data:02  RW:1
Cycle:               11  Addr:005  Data:00  RW:1
Cycle:               13  Addr:006  Data:0c  RW:1
Cycle:               15  Addr:00c  Data:03  RW:1
Cycle:               17  Addr:007  Data:e2  RW:1
Cycle:               19  Addr:008  Data:01  RW:1
L35 "cpu_test.v": $finish at simulation time 2000
```

CPU SignalScan



CPU Synthesized



AREA of Chip and Modules

```
****************************************
Report : area
Design : my_cpu
Version: 2000.11
Date   : Sun Apr 28 16:05:14 2002
****************************************
 Library(s) Used:

    lsi_10k (File: /apps/synopsys/2000.11/libraries/syn/lsi_10k.db)

Number of ports:              24
Number of nets:               72
Number of cells:               4
Number of references:          4

Combinational area:       755.000000
Noncombinational area:    690.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:         1445.000000

****************************************
Report : area
Design : alu_intg
Version: 2000.11
Date   : Sun Apr 28 15:06:05 2002
****************************************
Library(s) Used:

    lsi_10k (File: /apps/synopsys/2000.11/libraries/syn/lsi_10k.db)

Number of ports:              26
Number of nets:               43
Number of cells:               4
Number of references:          4

Combinational area:       422.000000
Noncombinational area:    198.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:         620.000000
```

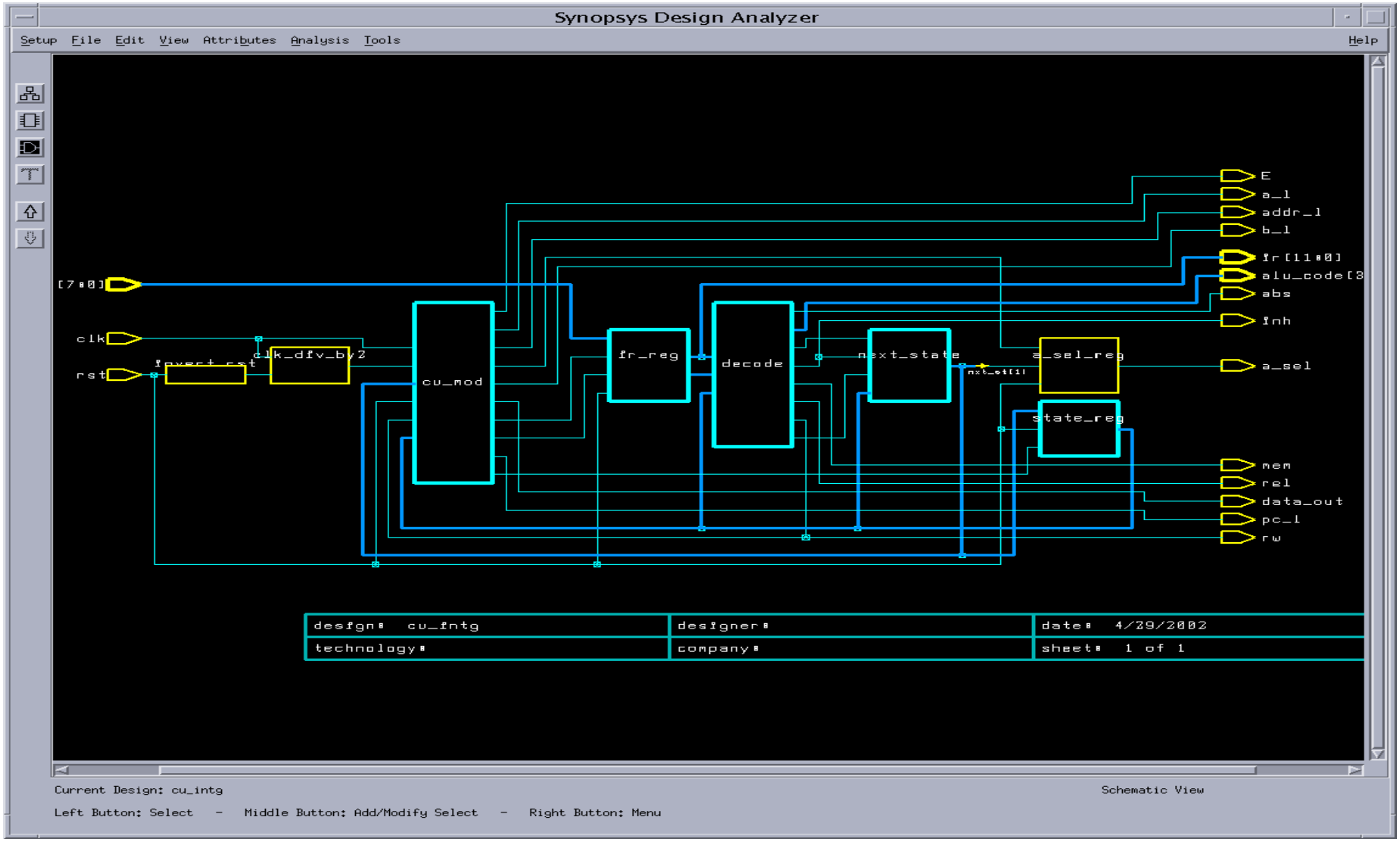
```
****************************************
Report : area
Design : pc_intg
Version: 2000.11
Date   : Sun Apr 28 15:09:15 2002
****************************************
 Library(s) Used:

    lsi_10k (File: /apps/synopsys/2000.11/libraries/syn/lsi_10k.db)

Number of ports:              31
Number of nets:               44
Number of cells:               3
Number of references:          3

Combinational area:       187.000000
Noncombinational area:    108.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:         295.000000

****************************************
Report : area
Design : cu_intg
Version: 2000.11
Date   : Sun Apr 28 15:54:02 2002
****************************************
Library(s) Used:

    lsi_10k (File: /apps/synopsys/2000.11/libraries/syn/lsi_10k.db)

Number of ports:              38
Number of nets:               54
Number of cells:               8
Number of references:          8

Combinational area:        88.000000
Noncombinational area:    252.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:         340.000000
```

Area (contd.) & Timing Report

```
****************************************
Report : area
Design : io_intg
Version: 2000.11
Date   : Sun Apr 28 15:16:49 2002
****************************************
 Library(s) Used:

    lsi_10k (File:
       /apps/synopsys/2000.11/libraries/syn/lsi_10k.db)

Number of ports:            57
Number of nets:             69
Number of cells:             3
Number of references:        3

Combinational area:      49.000000
Noncombinational area:   188.000000
Net Interconnect area:   undefined
                         (No wire load spec)

Total cell area:         237.000000
```

```
************************************
Report : timing
         -path full
         -delay max
         -max_paths 1
Design : io_intg
Version: 2000.11
Date   : Sun Apr 28 15:16:49 2002
************************************
Operating Conditions:
Wire Load Model Mode: top
  Startpoint: result[0] (input port clocked by clk)
  Endpoint: data[0] (output port clocked by clk)
  Path Group: clk
  Path Type: max

  Point                                    Incr       Path
  -----------------------------------------------------------
  clock clk (rise edge)                    0.00       0.00
  clock network delay (ideal)              0.00       0.00
  input external delay                     5.00       5.00 f
  result[0] (in)                           0.00       5.00 f
  tri_1/result[0] (io_tri)                 0.00       5.00 f
  tri_1/data_tri[0]/Z (BTS4P)              0.74       5.74 f
  tri_1/data[0] (io_tri)                   0.00       5.74 f
  data[0] (out)                            0.00       5.74 f
  data arrival time                                   5.74

  clock clk (rise edge)                   10.00      10.00
  clock network delay (ideal)              0.00      10.00
  clock uncertainty                       -0.33       9.67
  output external delay                   -5.00       4.67
  data required time                                  4.67
  -----------------------------------------------------------
  data required time                                  4.67
  data arrival time                                  -5.74
  -----------------------------------------------------------
  slack (VIOLATED)                                   -1.07
```

Pretty Pictures1 ;)



This is the Carry Register! That's it Hope you Enjoyed the Presentation ;)