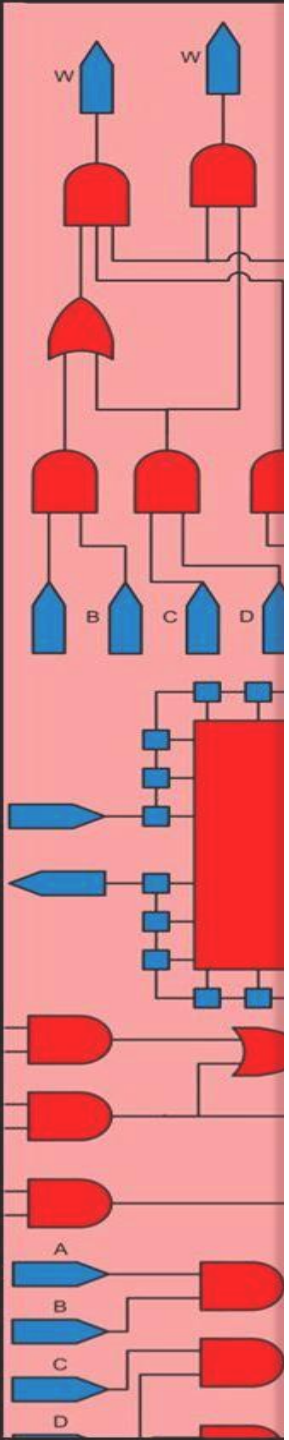
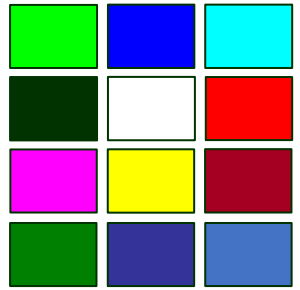
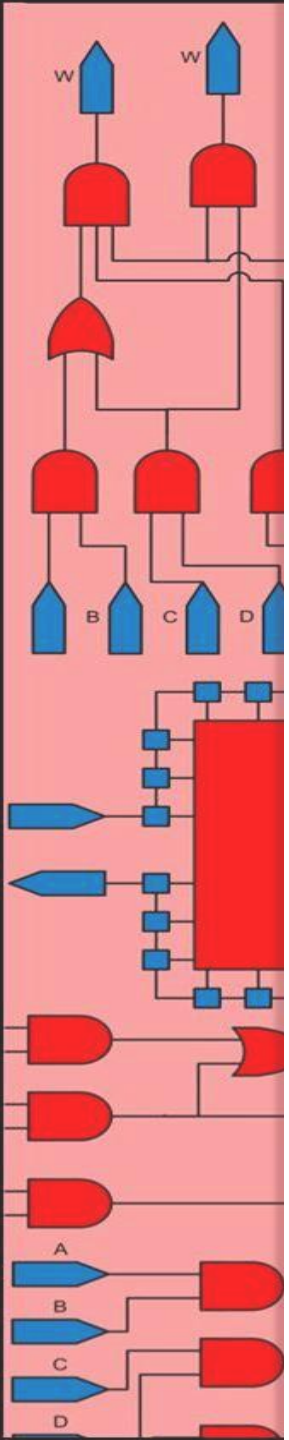


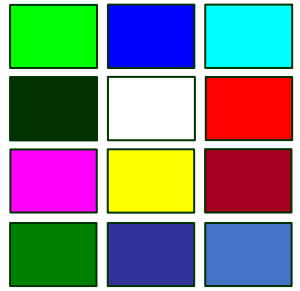


Yosys Synthesis

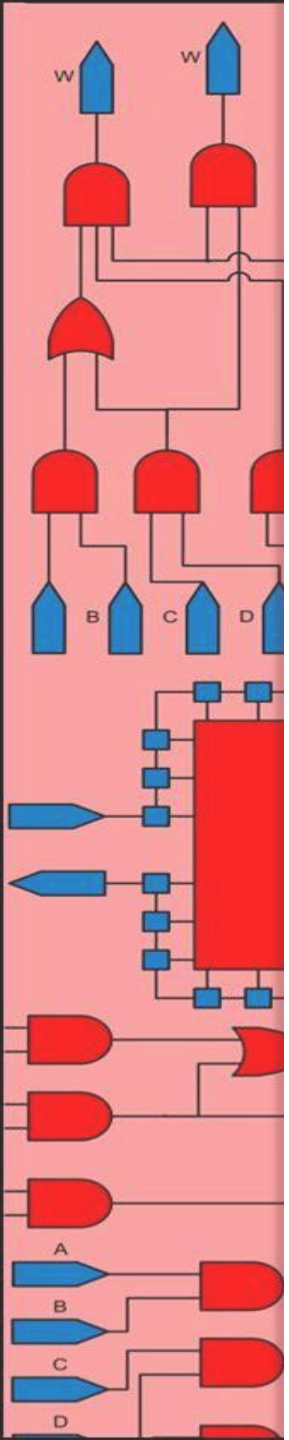
Appendix A



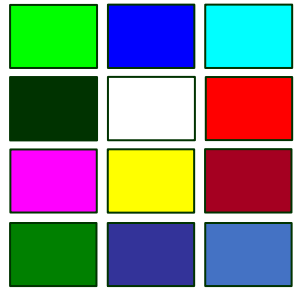
Outline



A-1 Basics of Yosys
A-2 Synthesize to gates
A-3 Combinational RTL
A-4 Sequential RTL
A-5 FSM
A-6 Complete RTL synthesis

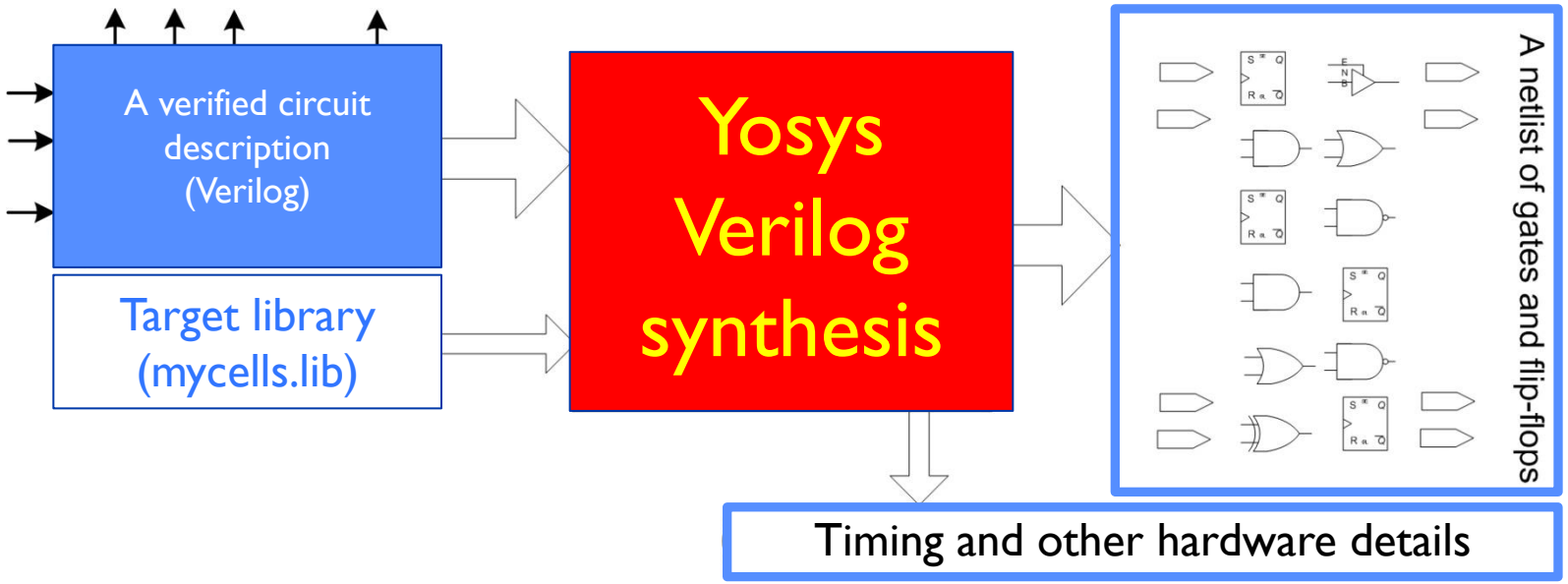


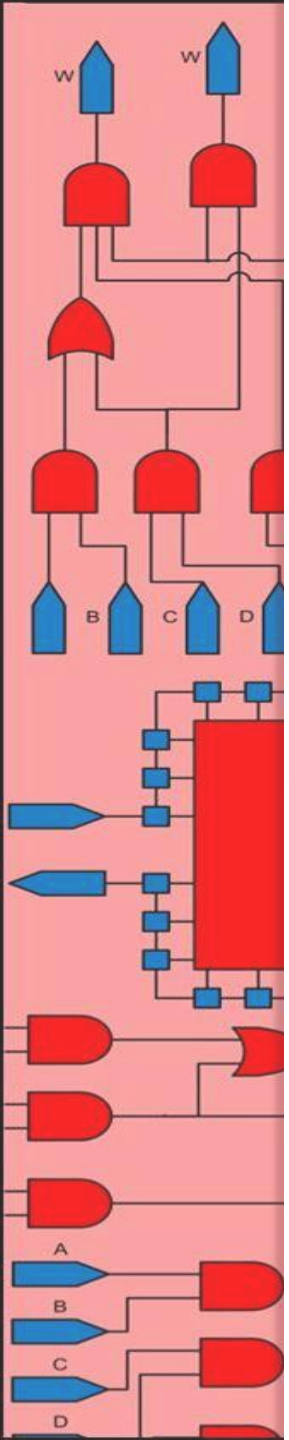
Outline



Synthesis with Yosys

Yosys – Yosys Open SYnthesis Suite





Yosys components

- Aspects of Yosys

Name	Date modified	Type	Size
example.v	4/10/2021 1:56 AM	V File	1 KB
mycells.lib	1/3/2021 9:43 PM	C compiler library ...	1 KB
mycells.v	1/3/2021 9:43 PM	V File	1 KB
pthreadVC2.dll	8/26/2019 1:58 PM	Application exten...	85 KB
simple_synth.v	4/10/2021 1:56 AM	V File	1 KB
yosys.exe	11/19/2020 11:45 AM	Application	5,325 KB
yosys-abc.exe	11/19/2020 11:45 AM	Application	12,161 KB

mycells.lib

```
library(demo) {
  cell(BUF) {
    area: 6;
    pin(A) { direction: input; }
    pin(Y) { direction: output;
             function: "A"; }
  }
  cell(NOT) {
    area: 3;
    pin(A) { direction: input; }
    pin(Y) { direction: output;
             function: "A'"; }
  }
  cell(NAND) {
    area: 4;
    pin(A) { direction: input; }
    pin(B) { direction: input; }
    pin(Y) { direction: output;
             function: "(A*B)'"; }
  }
  cell(NOR) {
    area: 4;
    pin(A) { direction: input; }
    pin(B) { direction: input; }
    pin(Y) { direction: output;
             function: "(A+B)'"; }
  }

}
```

Yosys commands

```
yosys> read_verilog myLittleCircuit.v
yosys> synth -auto-top
yosys> dfflibmap -liberty mycells.lib
yosys> abc -liberty mycells.lib
yosys> write_verilog -noattr myLittleCircuit_Synth.v
yosys> exit
```

mycells.v

```
module NOT(A, Y);
input A;
output Y;
assign Y = ~A;
endmodule

module NAND(A, B, Y);
input A, B;
output Y;
assign Y = ~(A & B);
endmodule

module NOR(A, B, Y);
input A, B;
output Y;
assign Y = ~(A | B);
endmodule

endmodule
```

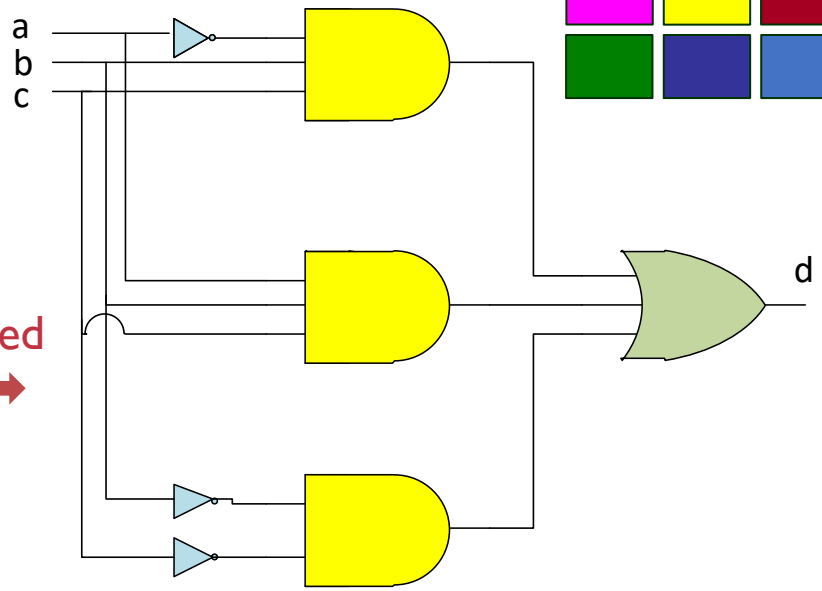
First run

Example

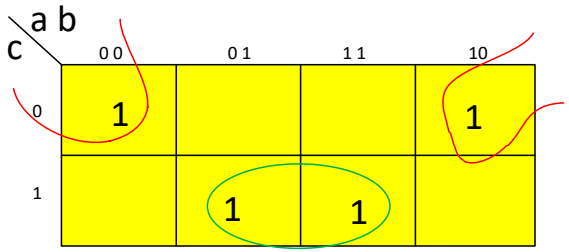
- Let's start a simple example

```
1  module simple (a,b,c,d);
2      input a,b,c;
3      output d;
4      assign d = (~a & b & c) | (a & b & c) | (~b & ~c);
5  endmodule
```

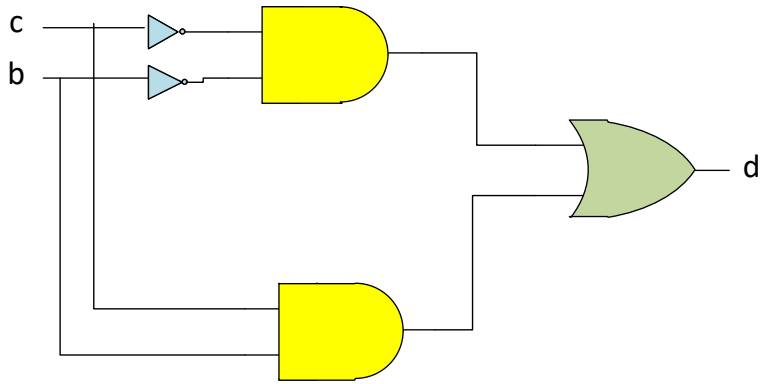
Not minimized



Minimize



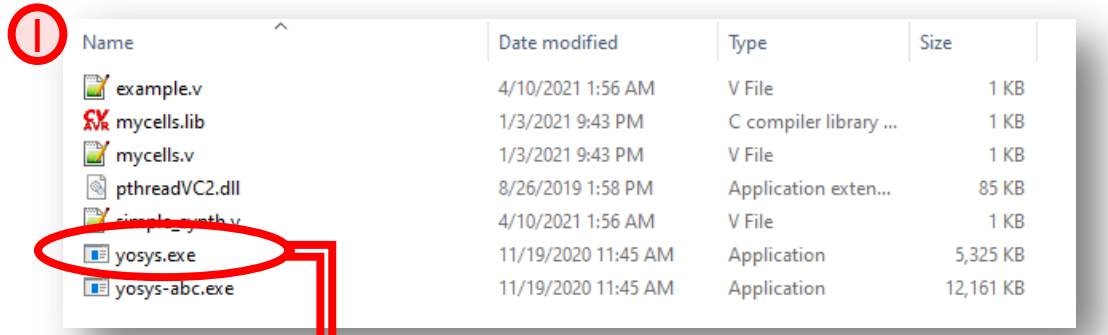
$d = bc + \bar{b}\bar{c}$

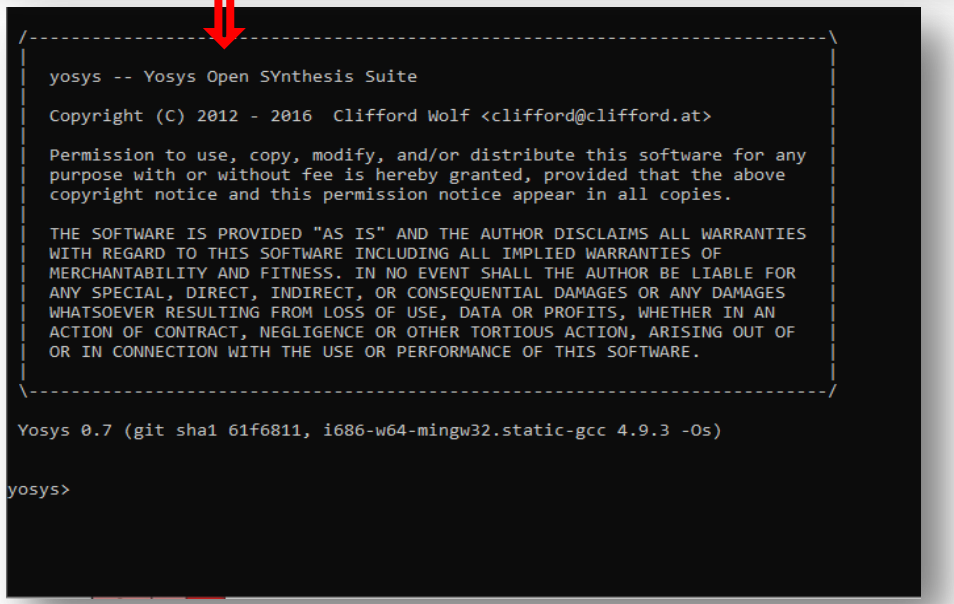


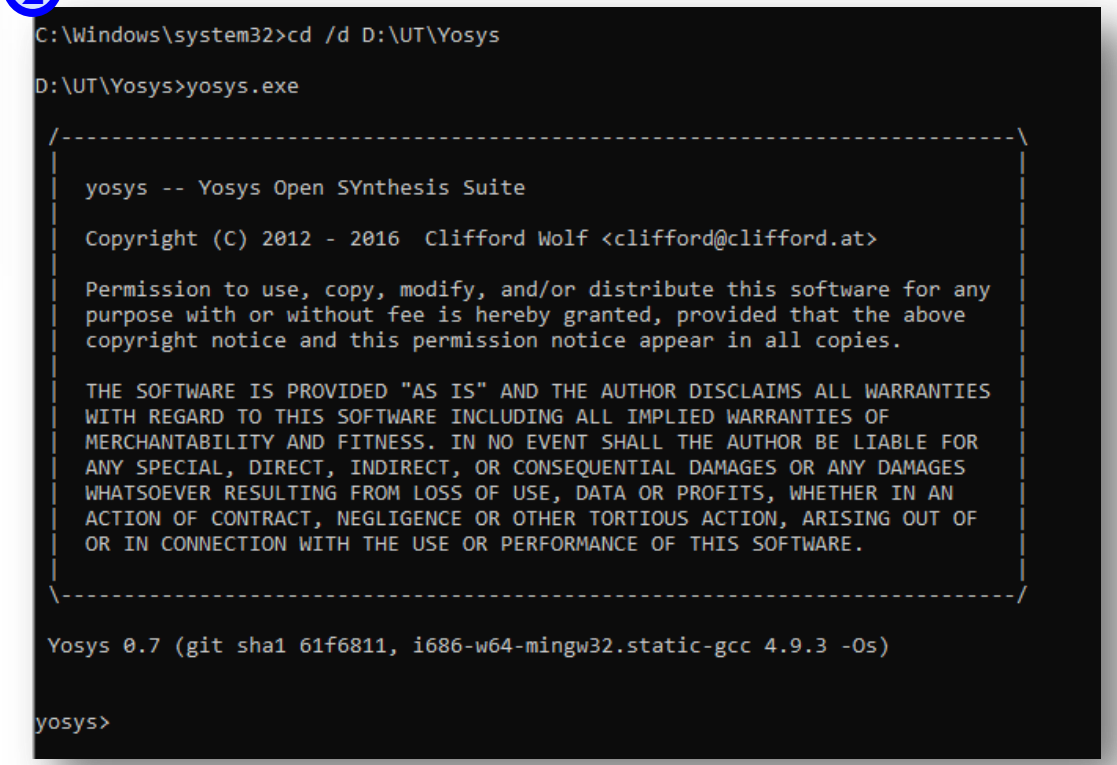
First run

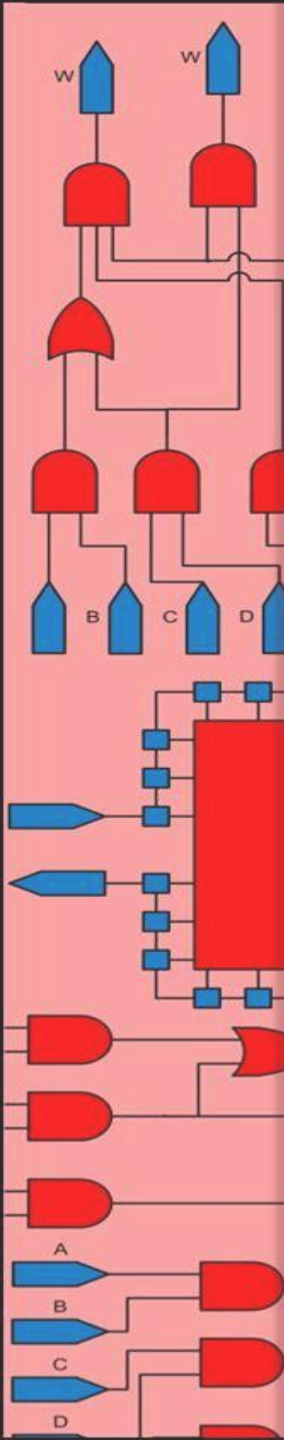
Step by step with Yosys

1- Load Yosys







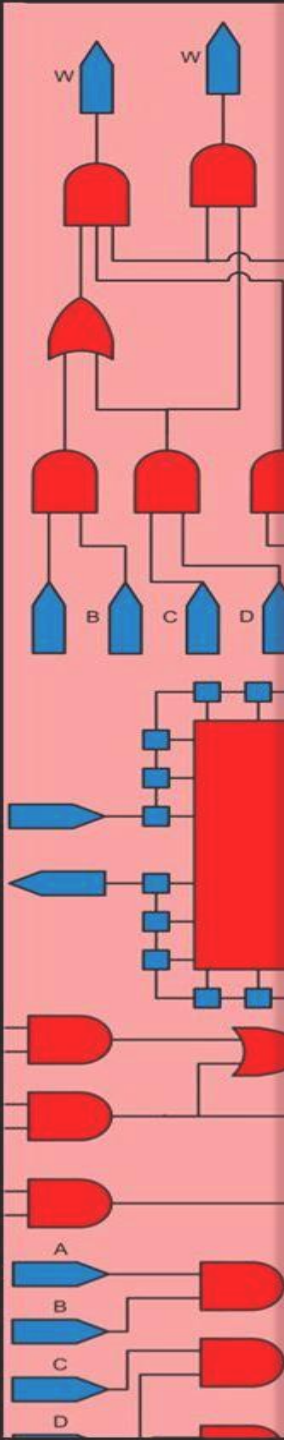


First run

Step by step with Yosys

1- Reading design file

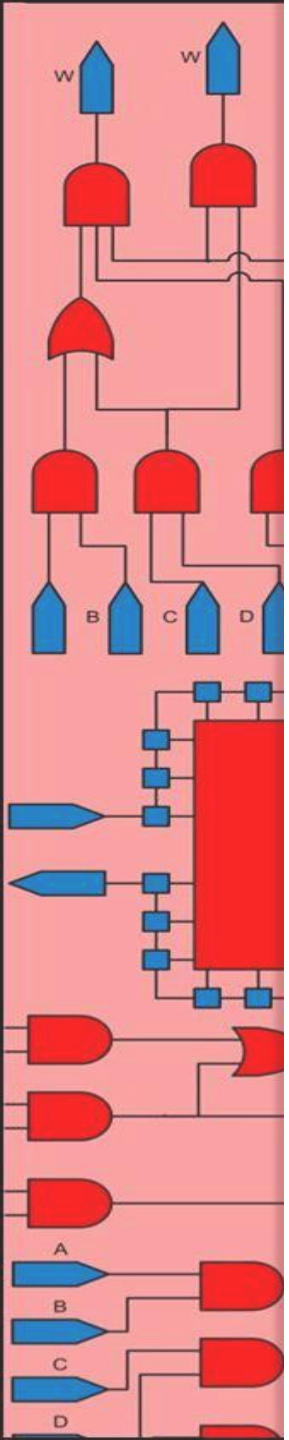
```
/----------------------------------------------------------------------------\
|                                                                            |
|  yosys -- Yosys Open SYnthesis Suite                                       |
|                                                                            |
|  Copyright (C) 2012 - 2016  Clifford Wolf <clifford@clifford.at>           |
|                                                                            |
|  Permission to use, copy, modify, and/or distribute this software for any  |
|  purpose with or without fee is hereby granted, provided that the above    |
|  copyright notice and this permission notice appear in all copies.         |
|                                                                            |
|  THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES  |
|  WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF          |
|  MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR   |
|  ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES    |
|  WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN     |
|  ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF   |
|  OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.            |
|                                                                            |
\----------------------------------------------------------------------------/

Yosys 0.7 (git sha1 61f6811, i686-w64-mingw32.static-gcc 4.9.3 -Os)

yosys> read_verilog simple.v
```



First run

2- Complete reading file

```
 /-----------------------------------------------------------------------------\
 |                                                                             |
 |  yosys -- Yosys Open SYnthesis Suite                                        |
 |                                                                             |
 |  Copyright (C) 2012 - 2016  Clifford Wolf <clifford@clifford.at>            |
 |                                                                             |
 |  Permission to use, copy, modify, and/or distribute this software for any   |
 |  purpose with or without fee is hereby granted, provided that the above     |
 |  copyright notice and this permission notice appear in all copies.          |
 |                                                                             |
 |  THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES   |
 |  WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF           |
 |  MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR    |
 |  ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES     |
 |  WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN      |
 |  ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF    |
 |  OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.             |
 |                                                                             |
 \-----------------------------------------------------------------------------/

 Yosys 0.7 (git sha1 61f6811, i686-w64-mingw32.static-gcc 4.9.3 -Os)


yosys> read_verilog simple.v
1. Executing Verilog-2005 frontend.
Parsing Verilog input from `simple.v' to AST representation.
Generating RTLIL representation for module `\simple'.
Successfully finished Verilog frontend.

yosys>
```



First run

3- Synthesis design file.

Synthesizing top module by the name

```
yosys> synth -top simple
```

or

```
yosys> synth -auto-top
```

Synthesizing top module automatically

4- Results of synthesis in Yosys without our library.

```
=== simple ===

   Number of wires:                 9
   Number of wire bits:             9
   Number of public wires:          4
   Number of public wire bits:      4
   Number of memories:              0
   Number of memory bits:           0
   Number of processes:             0
   Number of cells:                 6
     $_MUX_                         1
     $_NAND_                        2
     $_NOT_                         2
     $_OAI3_                        1
```

5- Needed command for recognizing our library, wrapper, … for mapping.

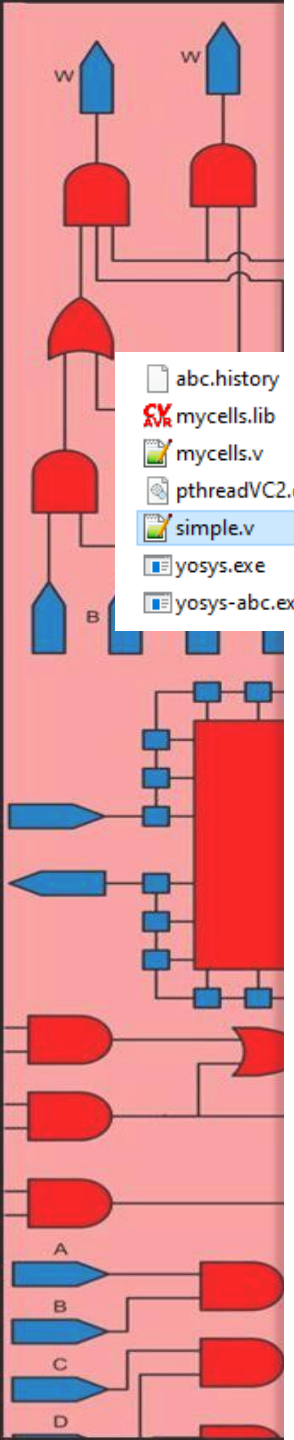
```
yosys> dfflibmap -liberty mycells.lib
```

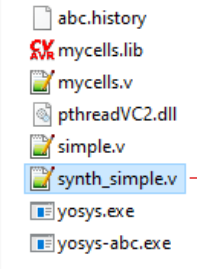
```
yosys> abc -liberty mycells.lib
```

6- Dumping results base on our library.

```
yosys> write_verilog simple-synth.v
```

Simple example

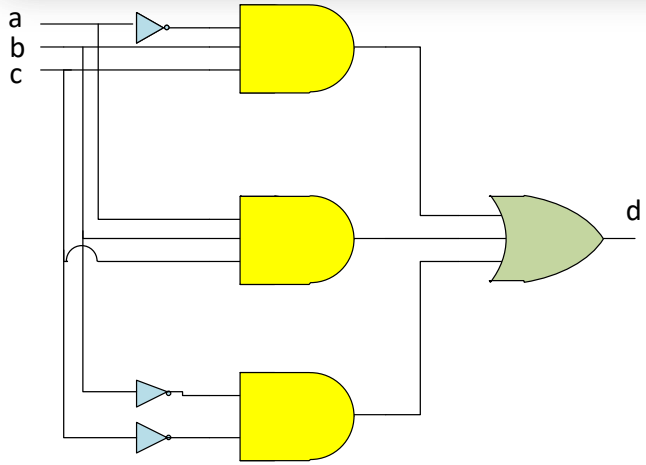




Input Verilog file and output result of Yosys

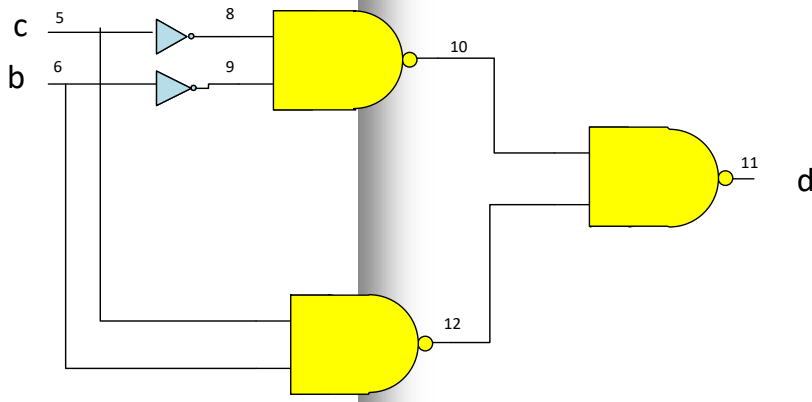
Yosys input

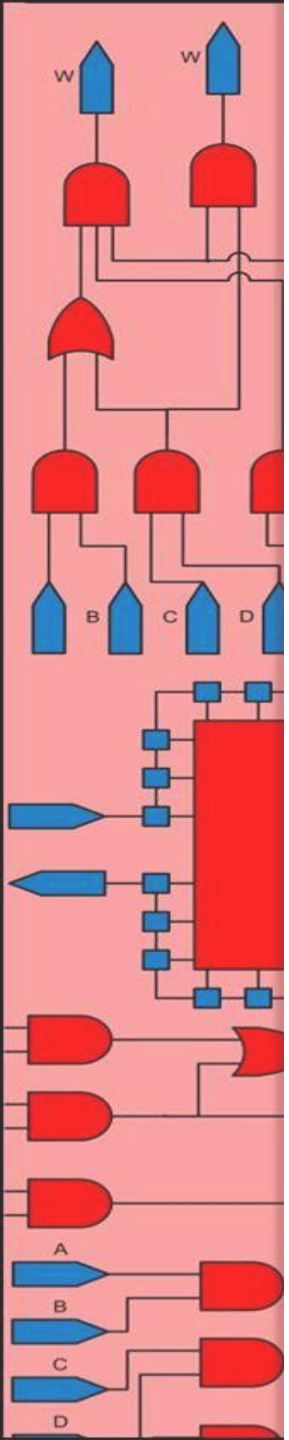
```
1  module simple (a,b,c,d);
2      input a,b,c;
3      output d;
4      assign d = (~a & b & c) | (a & b & c) | (~b & ~c);
5  endmodule
```



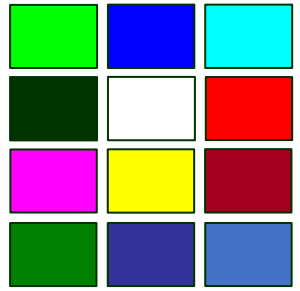
Yosys output

```
1   /* Generated by Yosys 0.7 (git sha1 61f6811, i686-w6
2
3   (* top =  1  *)
4   (* src = "simple.v:1" *)
5   module simple(a, b, c, d);
6     wire _00_;
7     wire _01_;
8     wire _02_;
9     wire _03_;
10    wire _04_;
11    wire _05_;
12    wire _06_;
13    wire _07_;
14    wire _08_;
15    wire _09_;
16    wire _10_;
17    wire _11_;
18    wire _12_;
19    (* src = "simple.v:2" *)
20    input a;
21    (* src = "simple.v:2" *)
22    input b;
23    (* src = "simple.v:2" *)
24    input c;
25    (* src = "simple.v:3" *)
26    output d;
27    NOT _13_ (
28      .A(_05_),
29      .Y(_08_)
30    );
31    NOT _14_ (
32      .A(_06_),
33      .Y(_09_)
34    );
35    NAND _15_ (
36      .A(_09_),
37      .B(_08_),
38      .Y(_10_)
39    );
40    NAND _16_ (
41      .A(_06_),
42      .B(_05_),
43      .Y(_12_)
44    );
45    NAND _17_ (
46      .A(_12_),
47      .B(_10_),
48      .Y(_11_)
49    );
50    assign _05_ = c;
51    assign _06_ = b;
52    assign _07_ = a;
53    assign d = _11_;
54  endmodule
55
```





Outline

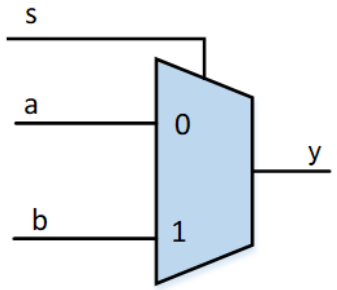


Multiplexer example

- Simple example synthesis for 2 to 1 MUX (1 bit)

Yosys input

```
1  module mux (a,b,s,y);
2      input a,b,s;
3      output y;
4      assign y = ~s ? a : b;
5  endmodule
```

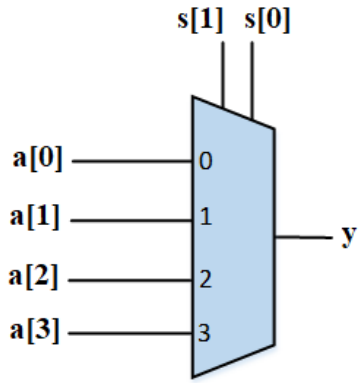


Yosys output

```
3  module mux(a, b, s, y);
4    wire _00_;
5    wire _01_;
6    wire _02_;
7    wire _03_;
8    wire _04_;
9    wire _05_;
10   wire _06_;
11   input a;
12   input b;
13   input s;
14   output y;
15
16   NOR _07_ (.A(_02_),.B(_00_),.Y(_04_));
17   NOT _08_ (.A(_02_),.Y(_05_));
18   NOR _09_ (.A(_05_),.B(_01_),.Y(_06_));
19   NOR _10_ (.A(_06_),.B(_04_),.Y(_03_));
20
21   assign _00_ = a;
22   assign _01_ = b;
23   assign _02_ = s;
24   assign y = _03_;
25 endmodule
26
```

Multiplexer example

- Simple example synthesis for 4 to 1 MUX (1 bit)



Yosys input

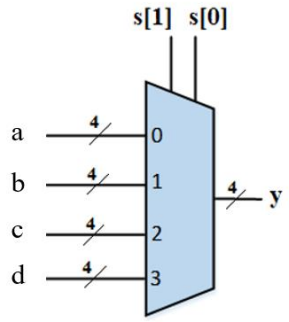
```
1   module mux4to1 (a,s,y);
2       input [3:0]a;
3       input [1:0]s;
4       output y;
5       assign y = (s == 2'b00) ? a[0] :
6                  (s == 2'b01) ? a[1] :
7                  (s == 2'b10) ? a[2] :
8                  (s == 2'b11) ? a[3] :
9                  1'bx;
10  endmodule
```

Part of Yosys output

```
NOT _23_ (.A(_06_),.Y(_20_));
NOR _24_ (.A(_21_),.B(_07_),.Y(_22_));
NOT _25_ (.A(_07_),.Y(_08_));
NOR _26_ (.A(_19_),.B(_08_),.Y(_10_));
NOR _27_ (.A(_10_),.B(_22_),.Y(_12_));
NOR _28_ (.A(_12_),.B(_20_),.Y(_13_));
NOR _29_ (.A(_09_),.B(_07_),.Y(_14_));
NOR _30_ (.A(_18_),.B(_08_),.Y(_15_));
NOR _31_ (.A(_15_),.B(_14_),.Y(_16_));
NOR _32_ (.A(_16_),.B(_06_),.Y(_17_));
NOR _33_ (.A(_17_),.B(_13_),.Y(_11_));
```

Multiplexer example

- Simple example synthesis for quad 4 to 1 MUX (4 bits)



Yosys input

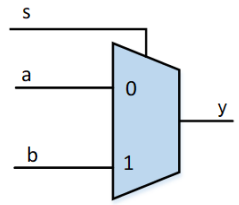
```
1  module mux4to1_4bit (a,b,c,d,s,y);
2      input [3:0]a,b,c,d;
3      input [1:0]s;
4      output [3:0]y;
5
6      assign y = (s == 2'b00) ? a :
7                 (s == 2'b01) ? b :
8                 (s == 2'b10) ? c :
9                 (s == 2'b11) ? d :
10                4'bxxx;
11 endmodule
```

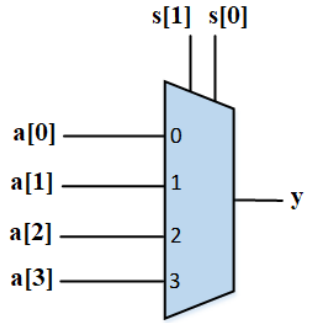
Part of Yosys output

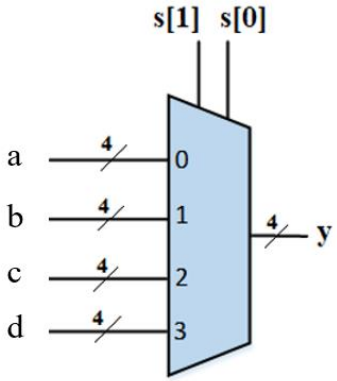
```
88   NOT _077_ (.A(_004_),.Y(_010_));
89   NOT _078_ (.A(_005_),.Y(_012_));
90   NAND _079_ (.A(_012_),.B(_010_),.Y(_014_));
91   NOR _080_ (.A(_014_),.B(_034_),.Y(_016_));
92   NOT _081_ (.A(_027_),.Y(_017_));
93   NAND _082_ (.A(_012_),.B(_004_),.Y(_019_));
94   NAND _083_ (.A(_005_),.B(_010_),.Y(_020_));
95   NAND _084_ (.A(_020_),.B(_019_),.Y(_022_));
96   NOR _085_ (.A(_022_),.B(_017_),.Y(_024_));
97   NOR _086_ (.A(_012_),.B(_004_),.Y(_026_));
98   NAND _087_ (.A(_026_),.B(_025_),.Y(_028_));
99   NOT _088_ (.A(_000_),.Y(_029_));
100  NAND _089_ (.A(_004_),.B(_029_),.Y(_030_));
101  NAND _090_ (.A(_030_),.B(_012_),.Y(_031_));
102  NAND _091_ (.A(_031_),.B(_028_),.Y(_032_));
103  NOR _092_ (.A(_032_),.B(_024_),.Y(_033_));
104  NOR _093_ (.A(_033_),.B(_016_),.Y(_053_));
105  NOR _094_ (.A(_014_),.B(_002_),.Y(_035_));
106  NOT _095_ (.A(_064_),.Y(_036_));
107  NOR _096_ (.A(_022_),.B(_036_),.Y(_037_));
108  NAND _097_ (.A(_026_),.B(_063_),.Y(_038_));
109  NOT _098_ (.A(_001_),.Y(_039_));
110  NAND _099_ (.A(_004_),.B(_039_),.Y(_040_));
111  NAND _100_ (.A(_040_),.B(_012_),.Y(_041_));
112  NAND _101_ (.A(_041_),.B(_038_),.Y(_042_));
113  NOR _102_ (.A(_042_),.B(_037_),.Y(_043_));
114  NOR _103_ (.A(_043_),.B(_035_),.Y(_003_));
115  NOR _104_ (.A(_014_),.B(_009_),.Y(_044_));
116  NOT _105_ (.A(_007_),.Y(_045_));
117  NOR _106_ (.A(_022_),.B(_045_),.Y(_046_));
118  NAND _107_ (.A(_026_),.B(_006_),.Y(_047_));
119  NOT _108_ (.A(_008_),.Y(_048_));
120  NAND _109_ (.A(_048_),.B(_004_),.Y(_049_));
121  NAND _110_ (.A(_049_),.B(_012_),.Y(_050_));
122  NAND _111_ (.A(_050_),.B(_047_),.Y(_051_));
123  NOR _112_ (.A(_051_),.B(_046_),.Y(_052_));
124  NOR _113_ (.A(_052_),.B(_044_),.Y(_011_));
125  NOR _114_ (.A(_014_),.B(_021_),.Y(_054_));
```

Multiplexer example

- Synthesis of various multiplexers and their number of cells








```
=== mux ===

   Number of wires:                 4
   Number of wire bits:             4
   Number of public wires:          4
   Number of public wire bits:      4
   Number of memories:              0
   Number of memory bits:           0
   Number of processes:             0
   Number of cells:                 1
     $_MUX_                         1
```

```
=== mux4to1 ===

   Number of wires:                 9
   Number of wire bits:            13
   Number of public wires:          3
   Number of public wire bits:      7
   Number of memories:              0
   Number of memory bits:           0
   Number of processes:             0
   Number of cells:                 7
     $_MUX_                         3
     $_NAND_                        1
     $_NOT_                         1
     $_OR_                          2
```

```
=== mux4to1_4bit ===

   Number of wires:                18
   Number of wire bits:            34
   Number of public wires:          6
   Number of public wire bits:     22
   Number of memories:              0
   Number of memory bits:           0
   Number of processes:             0
   Number of cells:                16
     $_MUX_                        12
     $_NAND_                        1
     $_NOT_                         1
     $_OR_                          2
```

Timing parameters

Synthesis without delay

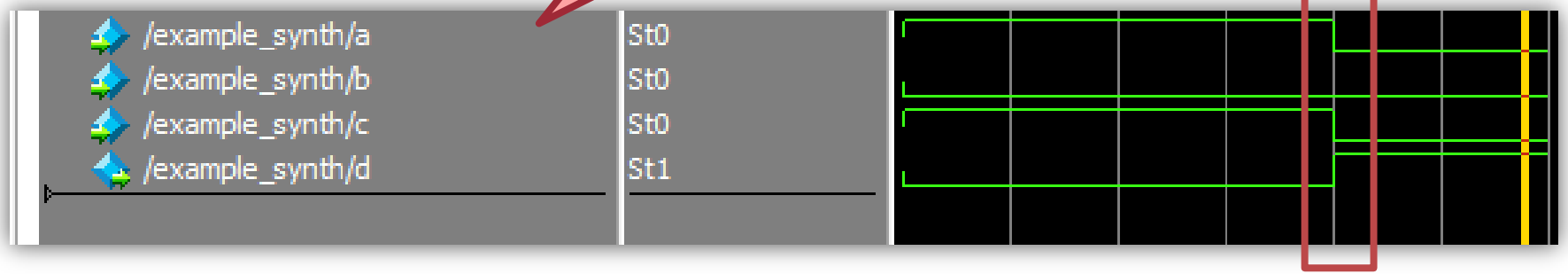
- Delay was not added

```
1  module example (a,b,c,d);
2      input a,b,c;
3      output d;
4      assign d = (~a & b & c) | (a & b & c) | (~b & ~c);
5  endmodule
```

```
1
2   module NOT(A, Y);
3   input A;
4   output Y;
5   assign Y = ~A;
6   endmodule
7
8   module NAND(A, B, Y);
9   input A, B;
10  output Y;
11  assign Y = ~(A & B);
12  endmodule
13
14  module NOR(A, B, Y);
15  input A, B;
16  output Y;
17  assign Y = ~(A | B);
18  endmodule
19
20  endmodule
```

Content of mycells.v library

Simulation output without delay



Timing parameters

Synthesis with delay

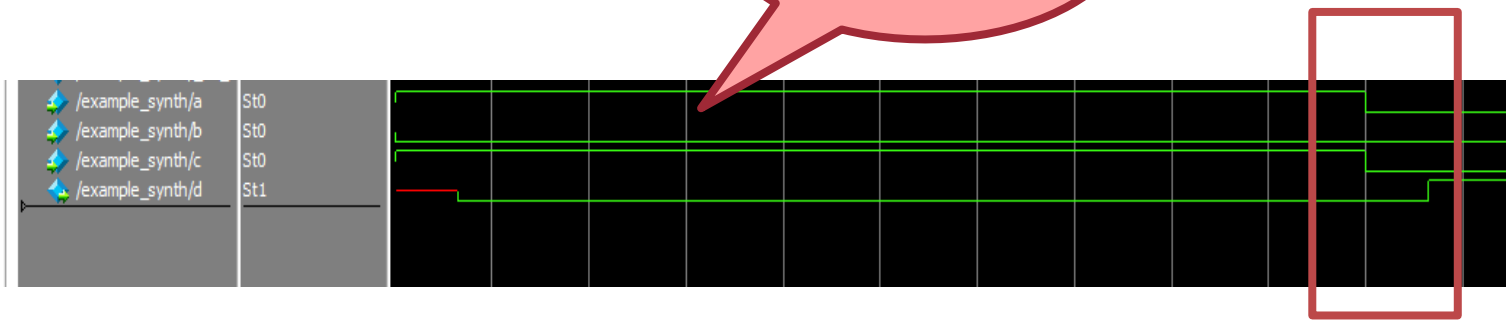
- Delay was added to the library (mycells.v)

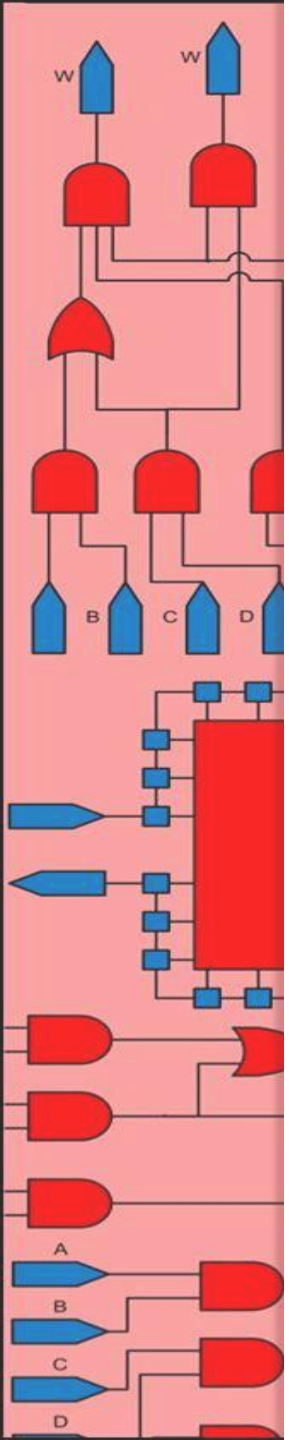
```
1  module example (a,b,c,d);
2      input a,b,c;
3      output d;
4      assign d = (~a & b & c) | (a & b & c) | (~b & ~c);
5  endmodule
```

```
2  module NOT(A, Y);
3  input A;
4  output Y;
5  assign #3 Y = ~A;
6  endmodule
7
8  module NAND(A, B, Y);
9  input A, B;
10 output Y;
11 assign #5 Y = ~(A & B);
12 endmodule
13
14 module NOR(A, B, Y);
15 input A, B;
16 output Y;
17 assign #5 Y = ~(A | B);
18 endmodule
19
```

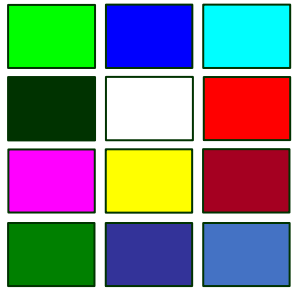
Content of mycells.v library

Simulation output with delay





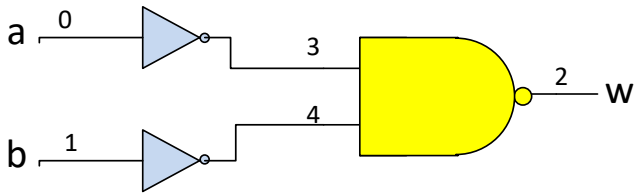
More examples



- A simple comparator (relational operator)

```
module comp (a,b,w);
    input a,b;
    output w;
    assign w = (a>=b) ? a : b;
endmodule
```

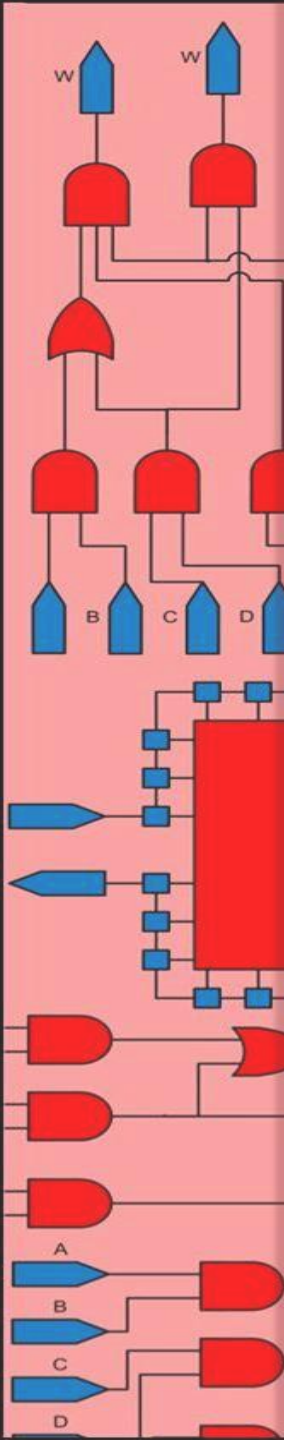
```
=== comp ===

   Number of wires:                 3
   Number of wire bits:             3
   Number of public wires:          3
   Number of public wire bits:      3
   Number of memories:              0
   Number of memory bits:           0
   Number of processes:             0
   Number of cells:                 1
     $_OR_                          1
```

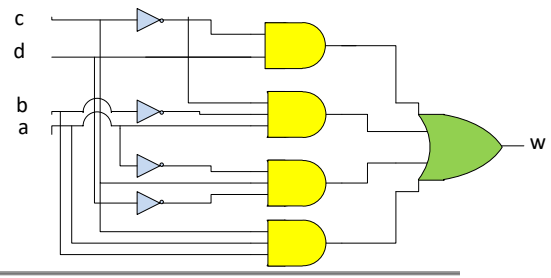



```
(* top =  1  *)
(* src = "comp.v:1" *)
module comp(a, b, w);
  wire _0_;
  wire _1_;
  wire _2_;
  wire _3_;
  wire _4_;
  (* src = "comp.v:2" *)
  input a;
  (* src = "comp.v:2" *)
  input b;
  (* src = "comp.v:3" *)
  output w;
  NOT _5_ (
    .A(_0_),
    .Y(_3_)
  );
  NOT _6_ (
    .A(_1_),
    .Y(_4_)
  );
  NAND _7_ (
    .A(_4_),
    .B(_3_),
    .Y(_2_)
  );
  assign _0_ = a;
  assign _1_ = b;
  assign w = _2_;
endmodule
```

More examples

- A Boolean expression example



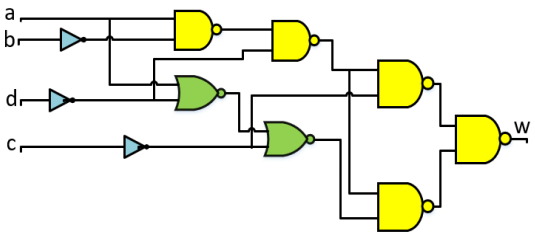



```
module hw6 (a,b,c,d,w);
    input a, b, c, d;
    output w;
    assign w = ( d & ~c) |(~a & ~d & c) | (a & ~b & ~c) | (b & a & c);
endmodule
```

Library with NOR gate

```
36   NOT _20_ (
37     .A(_17_),
38     .Y(_15_)
39   );
40   NAND _21_ (
41     .A(_15_),
42     .B(_13_),
43     .Y(_16_)
44   );
45   NOT _22_ (
46     .A(_06_),
47     .Y(_18_)
48   );
49   NOT _23_ (
50     .A(_14_),
51     .Y(_19_)
52   );
53   NOR _24_ (
54     .A(_19_),
55     .B(_13_),
56     .Y(_08_)
57   );
58   NOR _25_ (
59     .A(_08_),
60     .B(_18_),
61     .Y(_09_)
62   );
63   NAND _26_ (
64     .A(_09_),
65     .B(_16_),
66     .Y(_10_)
67   );
68   NAND _27_ (
69     .A(_16_),
70     .B(_19_),
71     .Y(_11_)
72   );
73   NAND _28_ (
74     .A(_11_),
75     .B(_18_),
76     .Y(_12_)
77   );
78   NAND _29_ (
79     .A(_12_),
80     .B(_10_),
81     .Y(_07_)
82   );
```

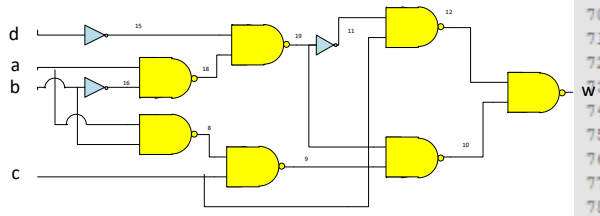
```
=== hw6 ===

   Number of wires:                 11
   Number of wire bits:             11
   Number of public wires:           5
   Number of public wire bits:       5
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                  7
     $_MUX_                          2
     $_NAND_                         3
     $_NOT_                          1
     $_OR_                           1
```



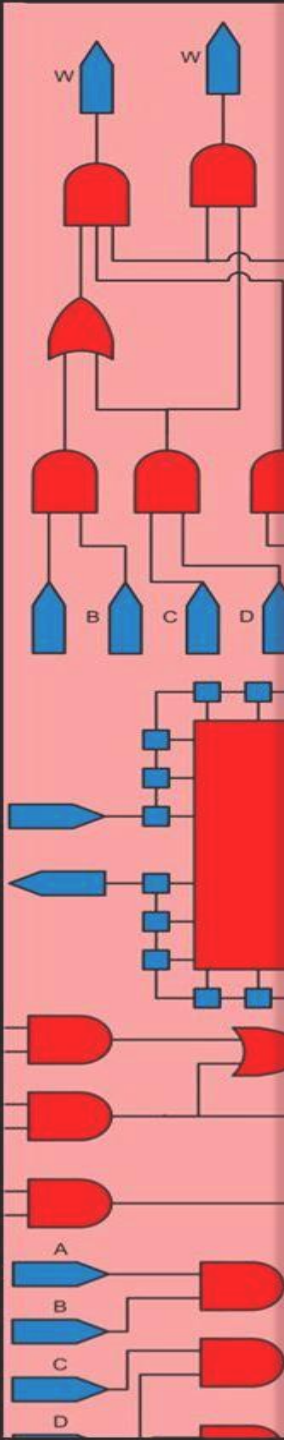

```
83     assign _06_ = c;
84     assign _13_ = a;
85     assign _14_ = d;
86     assign _17_ = b;
87     assign w = _07_;
88   endmodule
```

Library without NOR gate

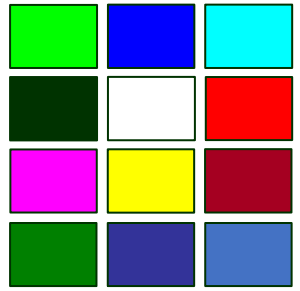
```
36   NOT _20_ (
37     .A(_14_),
38     .Y(_15_)
39   );
40   NOT _21_ (
41     .A(_17_),
42     .Y(_16_)
43   );
44   NAND _22_ (
45     .A(_16_),
46     .B(_13_),
47     .Y(_18_)
48   );
49   NAND _23_ (
50     .A(_18_),
51     .B(_15_),
52     .Y(_19_)
53   );
54   NAND _24_ (
55     .A(_17_),
56     .B(_13_),
57     .Y(_08_)
58   );
59   NAND _25_ (
60     .A(_08_),
61     .B(_06_),
62     .Y(_09_)
63   );
64   NAND _26_ (
65     .A(_09_),
66     .B(_19_),
67     .Y(_10_)
68   );
69   NOT _27_ (
70     .A(_19_),
71     .Y(_11_)
72   );
73   NAND _28_ (
74     .A(_11_),
75     .B(_06_),
76     .Y(_12_)
77   );
78   NAND _29_ (
79     .A(_12_),
80     .B(_10_),
81     .Y(_07_)
82   );
```




```
83     assign _06_ = c;
84     assign _13_ = a;
85     assign _14_ = d;
86     assign _17_ = b;
87     assign w = _07_;
88   endmodule
```



Outline



- A-1 Basics of Yosys
- A-2 Synthesize to gates
- A-3 Combinational RTL
  - A-3-1 OddParity block
  - A-3-2 comparator block
  - A-3-3 Full Adder
  - A-3-4 ALU block
  - A-3-5 Multiplication
- A-4 Sequential RTL
- A-5 FSM
- A-6 Complete RTL synthesis

Combinational synthesis

OddParity block

- OddParity statement in 3 versions (# of inputs: odd)

```
module oddParGS #(parameter S=6) (input [0:S] I, output odd);
    wire [0:S] J;
    assign J[0] = I[0];
    assign odd = J[S];

    genvar k;
    generate
        for (k=0; k<S; k=k+1) begin: XORgates
            xor XX (J[k+1], J[k], I[k+1]);
        end
    endgenerate
endmodule
```

Generate statement

```
=== oddParGS ===

   Number of wires:                  8
   Number of wire bits:             20
   Number of public wires:           3
   Number of public wire bits:      15
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                  6
     $_XNOR_                         6
```

```
module oddparity_ins (I,odd);
    input [0:6]I;
    output odd;
    wire J1, J2, J3, J4, J5, J6;
    xor x1 (J1, I[0], I[1]);
    xor x2 (J2, J1, I[2]);
    xor x3 (J3, J2, I[3]);
    xor x4 (J4, J3, I[4]);
    xor x5 (J5, J4, I[5]);
    xor x6 (odd, J5, I[6]);
endmodule
```

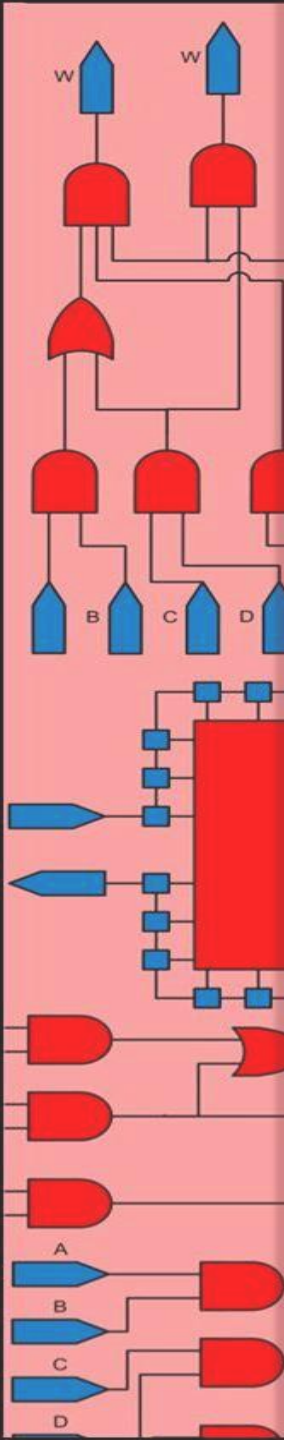
Primitive with 2 inputs

```
=== oddparity_ins ===

   Number of wires:                  7
   Number of wire bits:             13
   Number of public wires:           2
   Number of public wire bits:       8
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                  6
     $_XNOR_                         6
```

```
module oddparity_xor(I,odd);
    input [0:6]I;
    output odd;
    xor x ( odd,I[0], I[1], I[2], I[3], I[4], I[5], I[6]);
endmodule
```

Primitive with 7 inputs

```
=== oddparity_xor ===

   Number of wires:                  7
   Number of wire bits:             13
   Number of public wires:           2
   Number of public wire bits:       8
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                  6
     $_XNOR_                         6
```



Combinational synthesis

OddParity block

- OddParity statement in 3 versions
- (# of inputs: even)

```
module oddparity_Gs #(parameter S=7)(input [0:S]I, output odd);
    wire [0:S]J;
    assign J[0] = I[0];
    assign odd = J[S];

    genvar k;
    generate
        for (k=0; k<S; k=k+1)begin: XORgates
            xor XX (J[k+1], J[k], I[k+1]);
        end
    endgenerate
endmodule
```

Generate statement

```
=== oddparity_Gs ===

   Number of wires:                  9
   Number of wire bits:             23
   Number of public wires:           3
   Number of public wire bits:      17
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                  7
     $_XNOR_                         6
     $_XOR_                          1
```

```
module oddparity_ins (I,odd);
    input [0:7]I;
    output odd;
    wire J1, J2, J3, J4, J5, J6;
    xor x1 (J1, I[0], I[1]);
    xor x2 (J2, J1, I[2]);
    xor x3 (J3, J2, I[3]);
    xor x4 (J4, J3, I[4]);
    xor x5 (J5, J4, I[5]);
    xor x6 (J6, J5, I[6]);
    xor x7 (odd, J6, I[7]);
endmodule
```

Primitive with 2 inputs

```
=== oddparity_ins ===

   Number of wires:                  8
   Number of wire bits:             15
   Number of public wires:           2
   Number of public wire bits:       9
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                  7
     $_XNOR_                         6
     $_XOR_                          1
```

```
module oddparity_xor(I,odd);
    input [0:7]I;
    output odd;
    xor x ( odd,I[0], I[1], I[2], I[3], I[4], I[5], I[6], I[7]);
endmodule
```

Primitive with 8 inputs

```
=== oddparity_xor ===

   Number of wires:                  8
   Number of wire bits:             15
   Number of public wires:           2
   Number of public wire bits:       9
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                  7
     $_XNOR_                         6
     $_XOR_                          1
```

Combinational synthesis

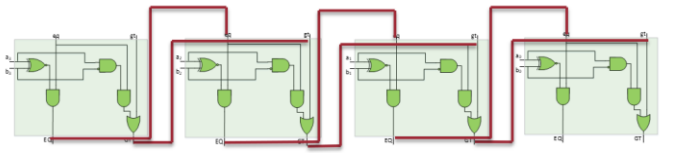
Comparator block

- Comparator in 2 versions (assign vs instantiation)

Instantiation by simple slice

```
1   module comprator_slice (a, b, eq, gt, o_EQ, o_GT);
2       input a, b, eq, gt;
3       output o_EQ, o_GT;
4       assign o_EQ = (~(a ^ b))& eq;
5       assign o_GT = ((a & ~b)& eq)|gt;
6   endmodule
7
8   module comprator_4bits (a, b, o_EQ, o_GT);
9       input [3:0]a, b;
10      output o_EQ, o_GT;
11      wire eq3, gt3, eq2, gt2, eq1, gt1;
12      comprator_slice ins3 (a[3], b[3], 1'b1, 1'b0, eq3, gt3);
13      comprator_slice ins2 (a[2], b[2], eq3, gt3, eq2, gt2);
14      comprator_slice ins1 (a[1], b[1], eq2, gt2, eq1, gt1);
15      comprator_slice ins0 (a[0], b[0], eq1, gt1, o_EQ, o_GT);
16  endmodule
```

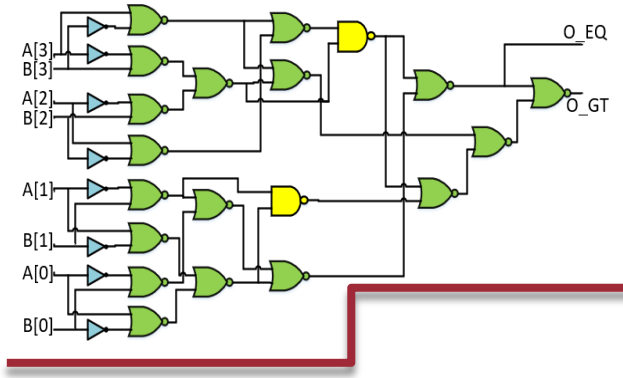
```
=== comprator_slice ===

   Number of wires:                 11
   Number of wire bits:             11
   Number of public wires:           6
   Number of public wire bits:       6
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                  7
     $_AND_                          1
     $_AOI3_                         1
     $_NOT_                          3
     $_OAI3_                         1
     $_OR_                           1
```



```
=== design hierarchy ===

   comprator_4bits                   1
     comprator_slice                 4

   Number of wires:                 56
   Number of wire bits:             62
   Number of public wires:          36
   Number of public wire bits:      42
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 28
     $_AND_                          4
     $_AOI3_                         4
     $_NOT_                         12
     $_OAI3_                         4
     $_OR_                           4
```

Assign statement

```
1   module comprator_4bits_assign (a, b, o_EQ, o_GT);
2       input [3:0]a, b;
3       output o_EQ, o_GT;
4       assign o_EQ = a==b;
5       assign o_GT = a>b;
6   endmodule
```




```
=== comprator_4bits_assign ===

   Number of wires:                 20
   Number of wire bits:             26
   Number of public wires:           4
   Number of public wire bits:      10
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 18
     $_AND_                          2
     $_AOI3_                         3
     $_NOR_                          4
     $_NOT_                          4
     $_OAI3_                         1
     $_XNOR_                         2
     $_XOR_                          2
```

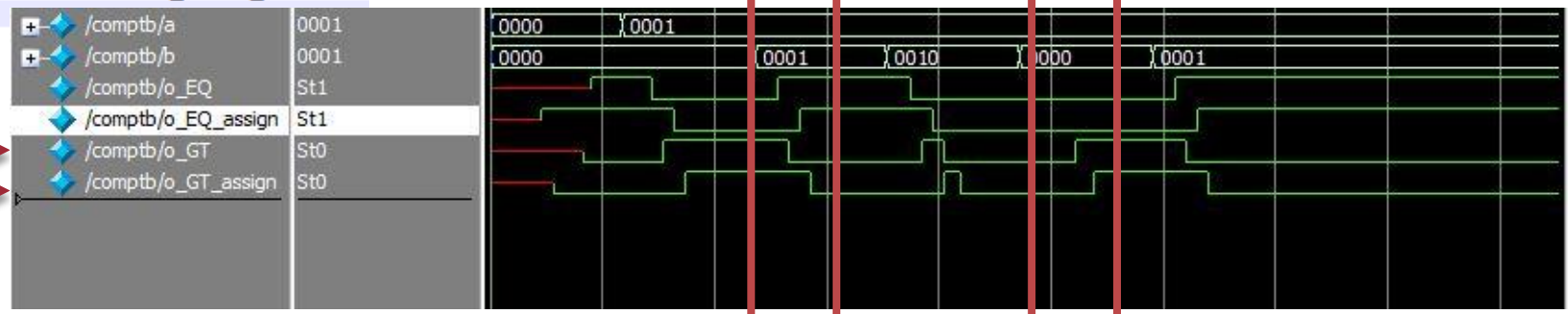
Combinational synthesis

Comparator block

- Comparator in 2 versions (assign vs instantiation)

Instantiation by simple slice

```
1  module comprator_slice (a, b, eq, gt, o_EQ, o_GT);
2      input a, b, eq, gt;
3      output o_EQ, o_GT;
4      assign o_EQ = (~(a ^ b))& eq;
5      assign o_GT = ((a & ~b)& eq)|gt;
6  endmodule
7
8  module comprator_4bits (a, b, o_EQ, o_GT);
9      input [3:0]a, b;
10     output o_EQ, o_GT;
11     wire eq3, gt3, eq2, gt2, eql, gtl;
12     comprator_slice ins3 (a[3], b[3], 1'bl, 1'b0, eq3, gt3);
13     comprator_slice ins2 (a[2], b[2], eq3, gt3, eq2, gt2);
14     comprator_slice insl (a[1], b[1], eq2, gt2, eql, gtl);
15     comprator_slice ins0 (a[0], b[0], eql, gtl, o_EQ, o_GT);
16 endmodule
```



Assign statement

```
1  module comprator_4bits_assign (a, b, o_EQ, o_GT);
2      input [3:0]a, b;
3      output o_EQ, o_GT;
4      assign o_EQ = a==b;
5      assign o_GT = a>b;
6  endmodule
```

Combinational synthesis

Comparator block

- Comparator in 2 versions and different size by parameter s

generate statement

```
module comprator_slice (a, b, eq, gt, o_EQ, o_GT);
    input a, b, eq, gt;
    output o_EQ, o_GT;
    assign o_EQ = (~(a ^ b))& eq;
    assign o_GT = ((a & ~b)& eq)|gt;
endmodule

module comprator_2048bits_gen #(parameter s=2048) (a, b, o_EQ, o_GT);
    input [s-1:0]a, b;
    output o_EQ, o_GT;
    wire [s:0] eq,gt;
    genvar k;
    generate
        for (k=s-1; k >=0 ; k=k-1)begin
            comprator_slice ins (a[k], b[k], eq[k+1], gt[k+1], eq[k], gt[k]);
        end
    endgenerate
    assign eq[s]=1'b1;
    assign gt[s]=1'b0;
    assign o_EQ= eq[0];
    assign o_GT= gt[0];
endmodule
```

S= 4

```
=== design hierarchy ===

  comprator_4bits_gen              1
    comprator_slice                4

  Number of wires:                50
  Number of wire bits:            62
  Number of public wires:         30
  Number of public wire bits:     42
  Number of memories:              0
  Number of memory bits:           0
  Number of processes:             0
  Number of cells:                28
    $_AND_                         4
    $_AOI3_                        4
    $_NOT_                        12
    $_OAI3_                        4
    $_OR_                          4
```

```
=== comprator_4bits_assign ===

  Number of wires:                20
  Number of wire bits:            26
  Number of public wires:          4
  Number of public wire bits:     10
  Number of memories:              0
  Number of memory bits:           0
  Number of processes:             0
  Number of cells:                18
    $_AND_                         2
    $_AOI3_                        3
    $_NOR_                         4
    $_NOT_                         4
    $_OAI3_                        1
    $_XNOR_                        2
    $_XOR_                         2
```

S= 4

```
module comprator_2048bits_assign #(parameter s=4) (a, b, o_EQ, o_GT);
    input [s-1:0]a, b;
    output o_EQ, o_GT;
    assign o_EQ = a==b;
    assign o_GT = a>b;
endmodule
```

Assign statement

S= 2048

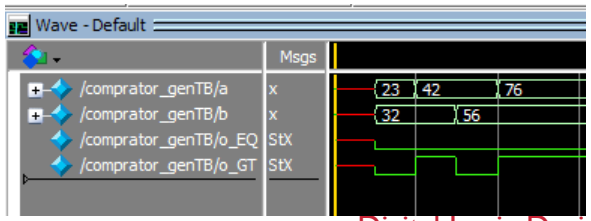
```
=== design hierarchy ===

  comprator_2048bits_gen           1
    comprator_slice             2048

  Number of wires:                22534
  Number of wire bits:            30722
  Number of public wires:         12294
  Number of public wire bits:     20482
  Number of memories:                 0
  Number of memory bits:              0
  Number of processes:                0
  Number of cells:                14336
    $_AND_                        2048
    $_AOI3_                       2048
    $_NOT_                        6144
    $_OAI3_                       2048
    $_OR_                         2048
```

```
=== comprator_2048bits_assign ===

  Number of wires:                12321
  Number of wire bits:            16417
  Number of public wires:             4
  Number of public wire bits:      4100
  Number of memories:                 0
  Number of memory bits:              0
  Number of processes:                0
  Number of cells:                12319
    $_AND_                          12
    $_AOI3_                       1366
    $_NAND_                       2337
    $_NOR_                        2716
    $_NOT_                        2056
    $_OAI3_                        683
    $_OR_                         1083
    $_XNOR_                       2032
    $_XOR_                          34
```

S= 2048

Testbench to check the parameter example

```
`timescale 1ns/1ns
module comprator_genTB();
 reg [4095:0] a, b;
   wire o_EQ, o_GT;
   comprator_4096bits_gen #(4096) ins (a, b, o_EQ, o_GT);
   initial begin
   #10 a=23;
   b=32;
   #10 a=42;
   #10 b=56;
   #10 a=76;
    #100 $stop;
   end
endmodule
```



It runs and gives the results immediately

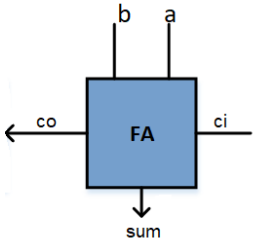
At large sizes the difference in cell number decreases and the delay increases

It takes approximately 60 seconds (wall clock time) to run and give results

Combinational synthesis

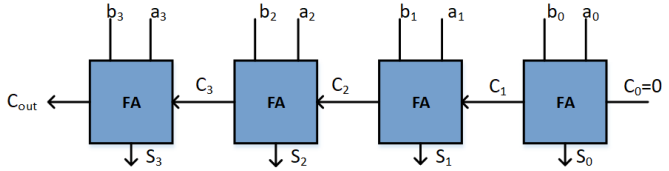
Full adder

- Full adder in different size of inputs




```
1  module example (a,b,ci,sum,co);
2      input a,b,ci;
3      output sum,co;
4      assign {co,sum} = a+b+ci;
5  endmodule
```

```
=== adder ===

   Number of wires:                 8
   Number of wire bits:             8
   Number of public wires:          5
   Number of public wire bits:      5
   Number of memories:              0
   Number of memory bits:           0
   Number of processes:             0
   Number of cells:                 5
     $_NAND_                        1
     $_NOT_                         1
     $_OAI3_                        1
     $_XNOR_                        2
```




```
1  module example (a,b,ci,sum,co);
2      input [3:0]a,b;
3      input ci;
4      output [3:0]sum;
5      output co;
6      assign {co,sum} = a+b+ci;
7  endmodule
```

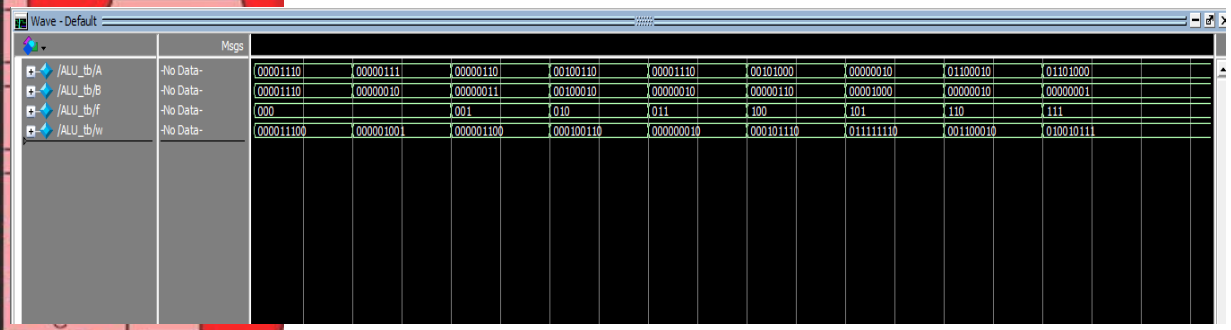
```
=== adder4bit ===

   Number of wires:                 20
   Number of wire bits:             29
   Number of public wires:           5
   Number of public wire bits:      14
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 20
     $_AND_                          4
     $_AOI3_                         3
     $_NAND_                         2
     $_OAI3_                         2
     $_OR_                           1
     $_XNOR_                         2
     $_XOR_                          6
```

Combinational synthesis

ALU block

- ALU with always statement

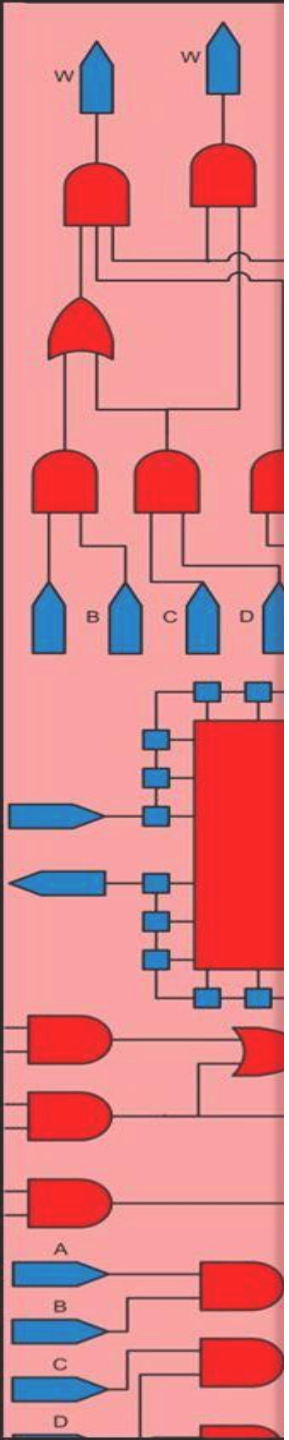
```
module ALU (input [7:0]A,B, input [2:0]f, output reg[7:0]w, output reg c);

    always @ (A, B, f) begin
    w = 8'b00000000;
    c = 1'b0;
    case (f)
        3'b000: {c,w} = A+B;
        3'b001: {c,w} ={A, 1'b0};
        3'b010: w = A>B ? A: B;
        3'b011: w = A&B;
        3'b100: w = A|B;
        3'b101: w =~A + 1;
        3'b110: w =A;
        3'b111: w =~A;
        default: w =8'b0;
      endcase
    end
endmodule
```




```
/* Generated by Yosys 0.7 (git sha1 61
module ALU(A, B, f, w, c);
  wire _000_;
  wire _001_;
  wire _002_;
  wire _003_;
  wire _004_;
  wire _005_;
  wire _006_;
  wire _007_;
  wire _008_;
  wire _009_;
  wire _010_;
  wire _011_;
  wire _012_;
  wire _013_;
  wire _014_;
  wire _015_;
  wire _016_;
  wire _017_;
  wire _018_;
  wire _019_;
  wire _020_;
  wire _021_;
  wire _022_;
  wire _023_;
  wire _024_;
  wire _025_;
  wire _026_;
  wire _027_;
  wire _028_;
  wire _029_;
  wire _030_;
  wire _031_;
```

```
  );
  NOT _803_ (
    .A(_427_),
    .Y(_439_)
  );
  NOT _804_ (
    .A(_417_),
    .Y(_440_)
  );
  NOT _805_ (
    .A(_428_),
    .Y(_447_)
  );
  NOR _806_ (
    .A(_447_),
    .B(_440_),
    .Y(_449_)
  );
  NOT _807_ (
    .A(_449_),
    .Y(_451_)
  );
  NOR _808_ (
    .A(_447_),
    .B(_417_),
    .Y(_452_)
  );
  NOR _809_ (
    .A(_428_),
    .B(_440_),
    .Y(_454_)
  );
  NOR _810_ (
    .A(_454_),
    .B(_452_),
    .Y(_456_)
  );
```

```
    .A(_281_),
    .Y(_475_)
  );
  NOR _828_ (
    .A(_475_),
    .B(_474_),
    .Y(_476_)
  );
  assign _194_ = A[7];
  assign _303_ = f[2];
  assign _457_ = f[0];
  assign _479_ = f[1];
  assign _510_ = B[1];
  assign _195_ = A[1];
  assign _226_ = B[0];
  assign _238_ = A[0];
  assign _270_ = B[3];
  assign _281_ = A[3];
  assign _304_ = B[2];
  assign _316_ = A[2];
  assign _395_ = B[7];
  assign _417_ = B[6];
  assign _428_ = A[6];
  assign _448_ = B[5];
  assign _450_ = A[5];
  assign _453_ = B[4];
  assign _455_ = A[4];
  assign c = _471_;
  assign w[0] = _216_;
  assign w[1] = _231_;
  assign w[2] = _250_;
  assign w[3] = _268_;
  assign w[4] = _283_;
  assign w[5] = _297_;
  assign w[6] = _313_;
  assign w[7] = _328_;
endmodule
```



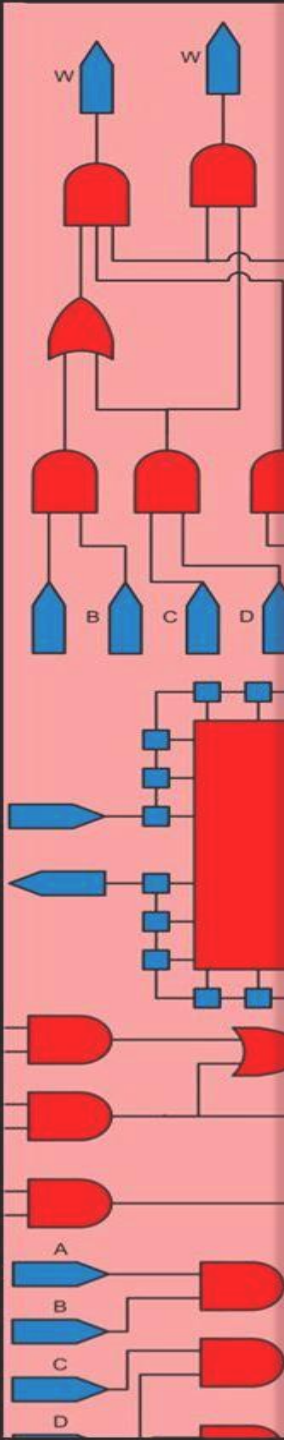
Combinational synthesis

Number of cells in ALU

- Attention to number of cell according to difference codes

```
1  module ALU (input [7:0]A,B, input [2:0]f, output reg[7:0]w, output reg c);
2
3      always @ (A, B, f) begin
4      w = 8'b00000000;
5      c = 1'b0;
6      case (f)
7          3'b000: {c,w} = A+B;
8          3'b001: {c,w} ={A, 1'b0};
9          3'b010: w = A>B ? A: B;
10         3'b011: w = A&B;
11         3'b100: w = A|B;
12         3'b101: w =~A + 1;
13         3'b110: w =A;
14         3'b111: w =~A;
15         default: w =8'b0;
16     endcase
17     end
18 endmodule
```

```
=== ALU ===

   Number of wires:                198
   Number of wire bits:            221
   Number of public wires:           5
   Number of public wire bits:      28
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                202
     $_AND_                          7
     $_AOI3_                        20
     $_AOI4_                         5
     $_MUX_                         25
     $_NAND_                        27
     $_NOR_                         27
     $_NOT_                         25
     $_OAI3_                        21
     $_OAI4_                         6
     $_OR_                          15
     $_XNOR_                        15
     $_XOR_                          9
```



Combinational synthesis

Number of cells in ALU

- Attention to number of cell according to difference codes

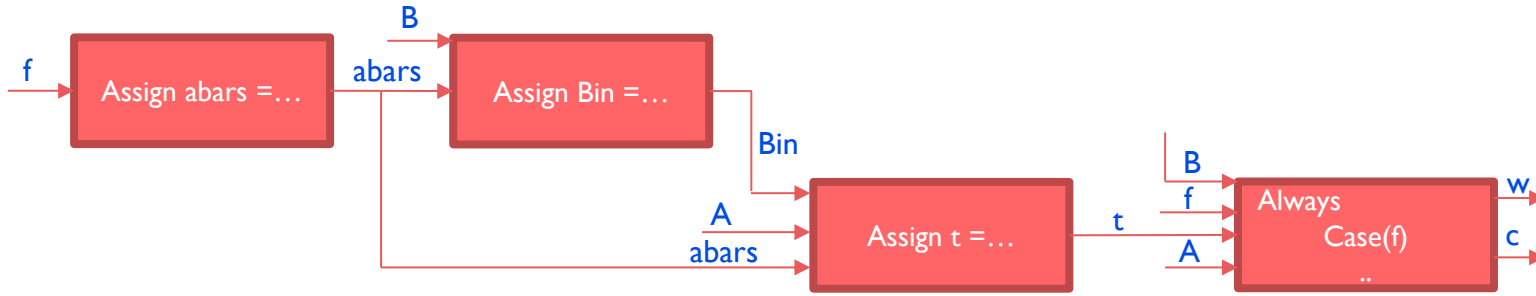
```
1  module ALU_20 (input [7:0]A, B, input [2:0]f, output reg[7:0]w, output reg c);
2      always @ (A, B, f) begin
3      w = 8'b0;
4      c = 1'b0;
5      case (f)
6          3'b000:  {c,w} = A + B;
7          3'b001:  {c,w} = A - B;
8          3'b010:  w = A;
9          3'b011:  w = A&B;
10         3'b100:  w = A|B;
11         3'b101:  w = ~B + 1;
12         3'b110:  w = A;
13         3'b111:  w = ~A;
14         default: w = 8'b0;
15     endcase
16     end
17 endmodule
```

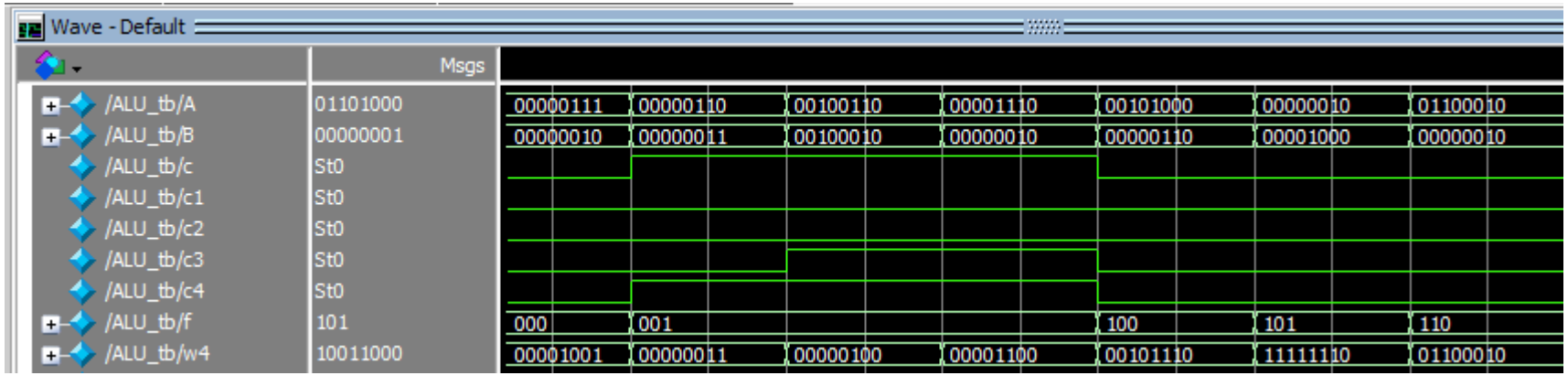
```
=== ALU ===

   Number of wires:                197
   Number of wire bits:            220
   Number of public wires:           5
   Number of public wire bits:      28
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                201
     $_AND_                         13
     $_AOI3_                        25
     $_AOI4_                         2
     $_MUX_                         10
     $_NAND_                        20
     $_NOR_                         29
     $_NOT_                         23
     $_OAI3_                        21
     $_OAI4_                         8
     $_OR_                          21
     $_XNOR_                        21
     $_XOR_                          8
```

Combinational synthesis

- Better design in hardware visual

```
1  module ALU_21 (input [7:0]A,B, input [2:0]f, output reg[7:0]w, output reg c);
2    reg abars;
3    wire [8:0] t;
4    wire [7:0] Bin;
5    assign abars = (f == 3'b001) ? 1'b1 : 1'b0;
6    assign Bin = {8{abars}} ^ B;
7    assign t = A + Bin + abars;
8      always @ (A, B, f, t) begin
9      w = 8'b00000000;
10     c = 1'b0;
11     case (f)
12       3'b000:  {c,w}= t;
13       3'b001:  {c,w}= t;
14       3'b010:  w = A;
15       3'b011:  w = A&B;
16       3'b100:  w = A|B;
17       3'b101:  w = ~B + 1;
18       3'b110:  w = A;
19       3'b111:  w = ~A;
20       default: w = 8'b0;
21     endcase
22   end
23 endmodule
```





Combinational synthesis

Number of cells in ALU (final)

- 3 different designs in order to make better number of cells

① Too many cells

```
1  module ALU_20 (input [7:0]A, B, input [2:0]f, output reg[7:0]w, output reg c);
3      always @ (A, B, f) begin
4      w = 8'b0;
5      c = 1'b0;
       case (f)
         3'b000:  {c,w} = A + B;
         3'b001:  {c,w} = A - B;
         3'b010:  w = A;
         3'b011:  w = A&B;
         3'b100:  w = A|B;
         3'b101:  w = ~B + 1;
         3'b110:  w = A;
         3'b111:  w = ~A;
         default: w = 8'b0;
       endcase
     end
17 endmodule
```

```
=== ALU_20 ===

   Number of wires:                198
   Number of wire bits:            221
   Number of public wires:           5
   Number of public wire bits:      28
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                202
     $_AND_                         13
     $_AOI3_                        25
     $_AOI4_                         2
     $_MUX_                          9
     $_NAND_                        21
     $_NOR_                         29
     $_NOT_                         24
     $_OAI3_                        21
     $_OAI4_                         9
     $_OR_                          20
     $_XNOR_                        19
     $_XOR_                         10
```

② good number of cells

```
1  module ALU_21 (input [7:0]A,B, input [2:0]f, output reg[7:0]w, output reg c);
     reg abars;
     wire [8:0] t;
     wire [7:0] Bin;
     assign abars = (f == 3'b001) ? 1'
     assign Bin = {8{abars}} ^ B;
     assign t = A + Bin + abars;
       always @ (A, B, f, t) begin
       w = 8'b00000000;
       c = 1'b0;
       case (f)
         3'b000:  {c,w}= t;
         3'b001:  {c,w}= t;
         3'b010:  w = A;
         3'b011:  w = A&B;
         3'b100:  w = A|B;
         3'b101:  w = ~B + 1;
         3'b110:  w = A;
         3'b111:  w = ~A;
         default: w = 8'b0;
       endcase
     end
23 endmodule
```

```
=== ALU_21 ===

   Number of wires:                148
   Number of wire bits:            171
   Number of public wires:           5
   Number of public wire bits:      28
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                152
     $_AND_                          3
     $_AOI3_                        11
     $_AOI4_                         6
     $_MUX_                          9
     $_NAND_                        18
     $_NOR_                         19
     $_NOT_                         20
     $_OAI3_                        17
     $_OAI4_                         5
     $_OR_                          15
     $_XNOR_                        23
     $_XOR_                          6
```

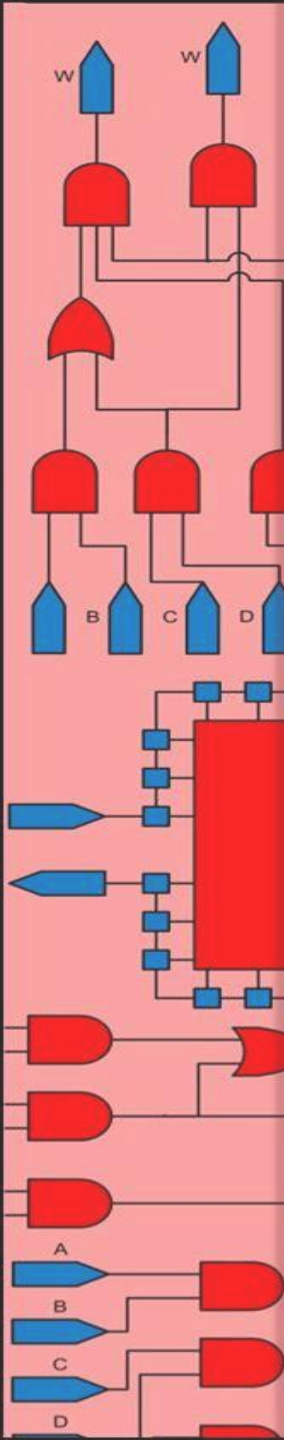
③

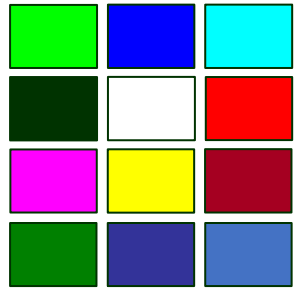
```
1  module ALU_22 (input [7:0]A, B, input [2:0]f, output reg[7:0]w, output reg c);
     reg abars, comp;
     wire [8:0] t;
     wire [7:0] Bin, w;
     assign abars = (f == 3'b001 | f == 3'b101) ? 1'b1 : 1'b0;
     assign comp = (f == 3'b101) ? 1'b1 : 1'b0;
     assign Bin = {8{abars}} ^ B;
     assign w =  comp ?  8'b00000000: A;
     assign t =   w + Bin + abars;
       always @ (A, B, f, t) begin
       w = 8'b00000000;
       c = 1'b0;
       case (f)
         3'b000:  {c,w} = t;
         3'b001:  {c,w} = t;
         3'b010:  w = A;
         3'b011:  w = A&B;
         3'b100:  w = A|B;
         3'b101:  w = t[7:0];  // ~B +1
         3'b110:  w = A;
         3'b111:  w = ~A;
         default: w = 8'b0;
       endcase
     end
25 endmodule
```

better number of cells

```
=== ALU_22 ===

   Number of wires:                122
   Number of wire bits:            145
   Number of public wires:           5
   Number of public wire bits:      28
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                126
     $_AND_                         10
     $_AOI3_                        13
     $_MUX_                          9
     $_NAND_                         9
     $_NOR_                         21
     $_NOT_                         12
     $_OAI3_                        14
     $_OAI4_                         8
     $_OR_                           7
     $_XNOR_                        21
     $_XOR_                          2
```

Synthesis rules (reminder)





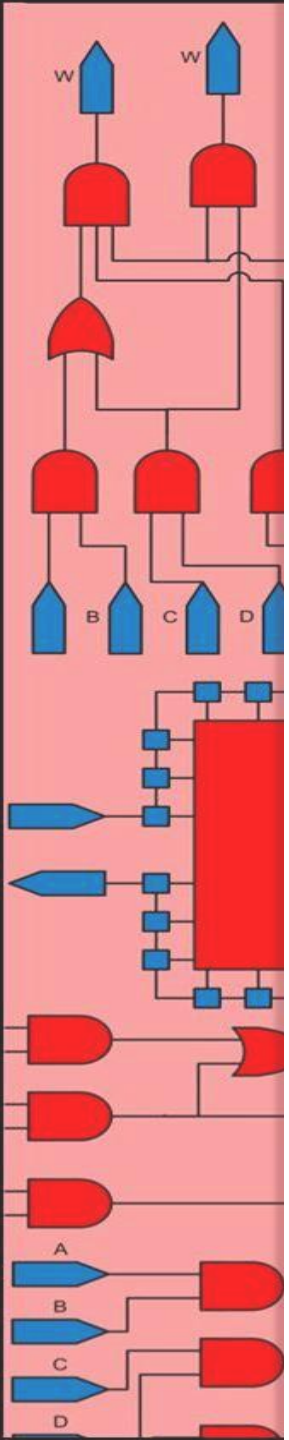
- Output rules

Output: All outputs must receive a value in the always statement.

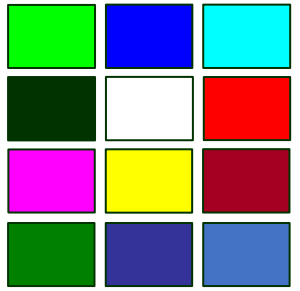
```
1   module ALU (input [7:0]A,B, input [2:0]f, output reg[7:0]w, output reg c);
2
3       always @ (A, B, f) begin
4       w = 8'b00000000;
5       c = 1'b0;
6       case (f)
7           3'b000: {c,w} = A + B;
8           3'b001: {c,w} = A - B;
9           3'b010: w = A;
10          3'b011: w = A&B;
11          3'b100: w = A|B;
12          3'b101: w =~A + 1;
13          3'b110: w =A;
14          3'b111: w =~A;
15          default: w =8'b0;
16      endcase
17      end
18  endmodule
19
20
```

Each signal on the left hand side must receive the values

Any violation to the rules will lead to a *LATCH.*



Synthesis rules (reminder)



- Input rules

Input: All input signals read must be in the sensitivity list.

```
1   module ALU (input [7:0]A,B, input [2:0]f, output reg[7:0]w, output reg c);
2
3       always @ (A, B, f) begin
4       w = 8'b00000000;
5       c = 1'b0;
6       case (f)
7           3'b000: {c,w} = A + B;
8           3'b001: {c,w} = A - B;
9           3'b010: w = A;
10          3'b011: w = A&B;
11          3'b100: w = A|B;
12          3'b101: w =~A + 1;
13          3'b110: w =A;
14          3'b111: w =~A;
15          default: w =8'b0;
16      endcase
17      end
18  endmodule
19
20
```

Each signal on the right hand side must be in the sensitivity list

Synthesis rules (reminder)

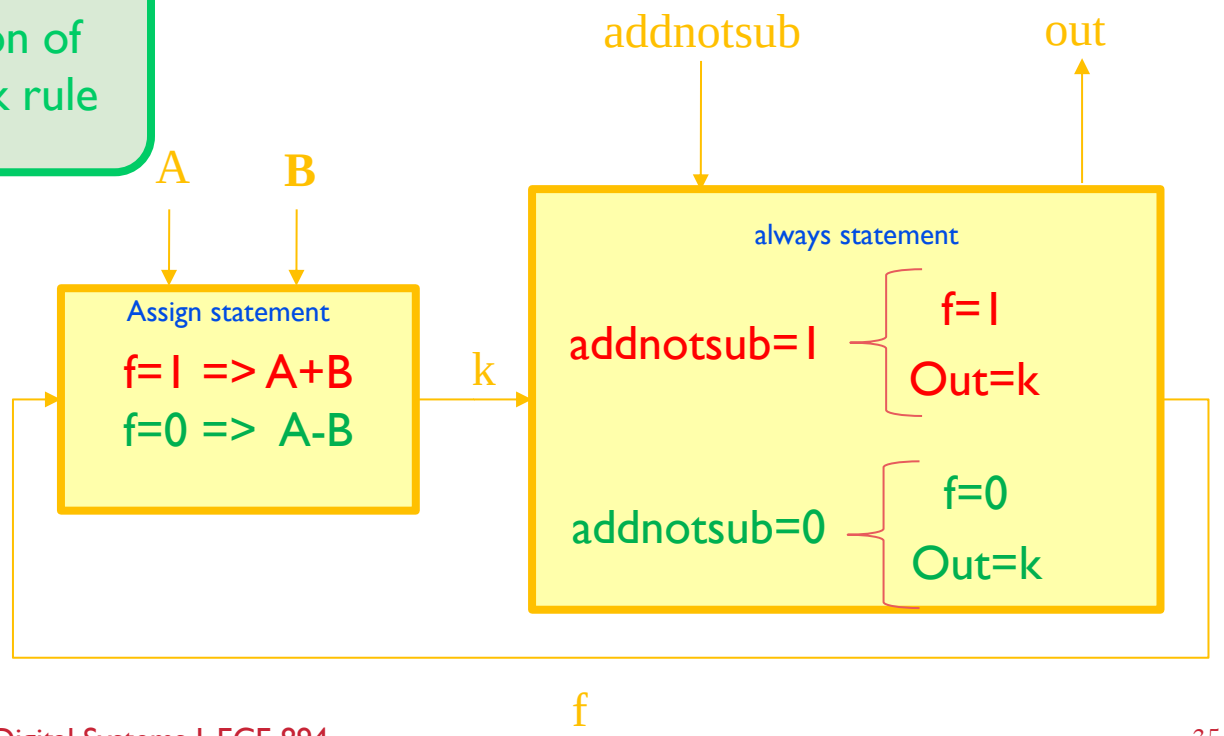
- No feedback

Feedback: Never use feedback in hardware implementation

Violation of feedback rule

```
module feedback (input [2:0]A,B, input addnotsub , output reg [2:0]out);
  wire [2:0]k;
  reg f;
  assign k = f ? A + B : A - B;
  always @ (addnotsub, k) begin
        f = 1'b1;
        out = 3'b0;
        if (addnotsub == 1'b1) begin
            f = 1'b1;
            out = k;
        end
        else if(addnotsub == 1'b0) begin
            f = 1'b0;
            out = k;
        end
    end
endmodule
```

```
=== feedback ===

   Number of wires:                 23
   Number of wire bits:             31
   Number of public wires:           6
   Number of public wire bits:      14
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 20
     $_AND_                          2
     $_AOI3_                         2
     $_MUX_                          3
     $_NOR_                          1
     $_NOT_                          2
     $_OR_                           1
     $_XNOR_                         4
     $_XOR_                          5
```



Synthesis rules (reminder)

In previous example

What if we violate these rules

- Output rules

```
module ALU_22 (input [7:0]A, B, input [2:0]f, output reg[7:0]w, output reg c);
  reg abars, comp;
  wire [8:0] t;
  wire [7:0] Bin, w;
  assign abars = (f == 3'b001 | f == 3'b101) ? 1'b1 : 1'b0;
  assign comp = (f == 3'b101) ? 1'b1 : 1'b0;
  assign Bin = {8{abars}} ^ B;
  assign w =  comp ?  8'b00000000: A;
  assign t =   w + Bin + abars;
    always @ (A, B, f, t) begin
    w = 8'b00000000;
    c = 1'b0;
    case (f)
        3'b000:  {c,w} = t;
        3'b001:  {c,w} = t;
        3'b010:  w = A;
        3'b011:  w = A&B;
        3'b100:  w = A|B;
        3'b101:  w = t[7:0];  // ~B +1
        3'b110:  w = A;
        3'b111:  w = ~A;
        default: w = 8'b0;
      endcase
    end
endmodule
```

```
=== ALU_22 ===

   Number of wires:                122
   Number of wire bits:            145
   Number of public wires:           5
   Number of public wire bits:      28
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                126
     $_AND_                         10
     $_AOI3_                        13
     $_MUX_                          9
     $_NAND_                         9
     $_NOR_                         21
     $_NOT_                         12
     $_OAI3_                        14
     $_OAI4_                         8
     $_OR_                           7
     $_XNOR_                        21
     $_XOR_                          2
```

Violation in output rules

```
module ALU_22w (input [7:0]A, B, input [2:0]f, output reg[7:0]w, output reg c);
  reg abars, comp;
  wire [8:0] t;
  wire [7:0] Bin, w;
  assign abars = (f == 3'b001 | f == 3'b101) ? 1'b1 : 1'b0;
  assign comp = (f == 3'b101) ? 1'b1 : 1'b0;
  assign Bin = {8{abars}} ^ B;
  assign w =  comp ?  8'b00000000: A;
  assign t =   w + Bin + abars;
    always @ (A, B, f, t) begin
    // w = 8'b00000000;
    // c = 1'b0;
    case (f)
        3'b000:  {c,w} = t;
        3'b001:  {c,w} = t;
        3'b010:  w = A;
        3'b011:  w = A&B;
        3'b100:  w = A|B;
        3'b101:  w = t[7:0];  // ~B +1
        3'b110:  w = A;
        3'b111:  w = ~A;
        default: w = 8'b0;
      endcase
    end
endmodule
```

```
=== ALU_22w ===

   Number of wires:                124
   Number of wire bits:            155
   Number of public wires:           6
   Number of public wire bits:      37
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                128
     $_AND_                         10
     $_DLATCH_P_                     1
     $_NAND_                        10
     $_NOR_                         20
     $_NOT_                         14
     $_OAI3_                        15
     $_OAI4_                         8
     $_OR_                           6
     $_XNOR_                        22
     $_XOR_                          1
```

DLATCH

Synthesis rules (reminder)

In previous example

What if we violate these rules

- Input rules

```
module ALU_22 (input [7:0]A, B, input [2:0]f, output reg[7:0]w, output reg c);
  reg abars, comp;
  wire [8:0] t;
  wire [7:0] Bin, w;
  assign abars = (f == 3'b001 | f == 3'b101) ? 1'b1 : 1'b0;
  assign comp = (f == 3'b101) ? 1'b1 : 1'b0;
  assign Bin = {8{abars}} ^ B;
  assign w =  comp ?  8'b00000000: A;
  assign t =   w + Bin + abars;
    always @ (A, B, f, t) begin
    w = 8'b00000000;
    c = 1'b0;
    case (f)
        3'b000:  {c,w} = t;
        3'b001:  {c,w} = t;
        3'b010:  w = A;
        3'b011:  w = A&B;
        3'b100:  w = A|B;
        3'b101:  w = t[7:0];  // ~B +1
        3'b110:  w = A;
        3'b111:  w = ~A;
        default: w = 8'b0;
      endcase
    end
endmodule
```

```
=== ALU_22 ===

   Number of wires:                122
   Number of wire bits:            145
   Number of public wires:           5
   Number of public wire bits:      28
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                126
     $_AND_                         10
     $_AOI3_                        13
     $_MUX_                          9
     $_NAND_                         9
     $_NOR_                         21
     $_NOT_                         12
     $_OAI3_                        14
     $_OAI4_                         8
     $_OR_                           7
     $_XNOR_                        21
     $_XOR_                          2
```

Violation in input rules

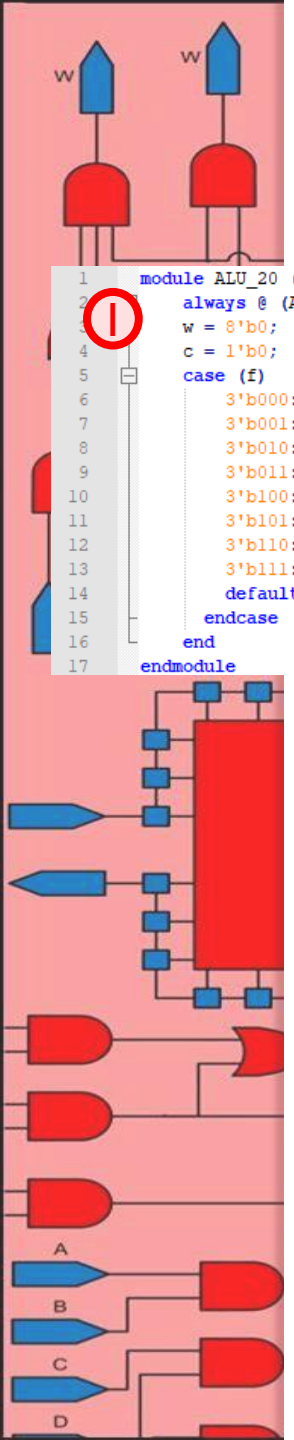
```
1  module ALU_21t (input [7:0]A,B, input [2:0]f, output reg[7:0]w, output reg c
2    reg abars;
3    wire [8:0] t;
4    wire [7:0] Bin;
5    assign abars = (f == 3'b001) ? 1'b1 : 1'b0;
6    assign Bin = {8{abars}} ^ B;
7    assign t = A + Bin + abars;
8      always @ (A, B, f) begin
9      w = 8'b00000000;
10     c = 1'b0;
11     case (f)
12         3'b000:  {c,w}= t;
13         3'b001:  {c,w}= t;
14         3'b010:  w = A;
15         3'b011:  w = A&B;
16         3'b100:  w = A|B;
17         3'b101:  w = ~B + 1;
18         3'b110:  w = A;
19         3'b111:  w = ~A;
20        default: w = 8'b0;
21      endcase
22    end
23 endmodule
```

```
=== ALU_21t ===

   Number of wires:                148
   Number of wire bits:            171
   Number of public wires:           5
   Number of public wire bits:      28
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                152
     $_AND_                          4
     $_AOI3_                        12
     $_AOI4_                         6
     $_MUX_                          9
     $_NAND_                        21
     $_NOR_                         18
     $_NOT_                         18
     $_OAI3_                        16
     $_OAI4_                         5
     $_OR_                          14
     $_XNOR_                        24
     $_XOR_                          5
```

In fact *t* is made by *A, B* which is in the sensitivity list so there is no problem with the input rule

Despite removing *t* from the sensitivity list, there is no LATCH

It will be removed




```
1  module ALU_20 (input [7:0]A, B, input [2:0]f, output reg[7:0]w, output reg c);
2      always @ (A, B, f) begin
3      w = 8'b0;
4      c = 1'b0;
5      case (f)
6          3'b000:  {c,w} = A + B;
7          3'b001:  {c,w} = A - B;
8          3'b010:  w = A;
9          3'b011:  w = A&B;
10         3'b100:  w = A|B;
11         3'b101:  w = ~B + 1;
12         3'b110:  w = A;
13         3'b111:  w = ~A;
14         default: w = 8'b0;
15     endcase
16     end
17 endmodule
```

```
=== ALU_20 ===

   Number of wires:                198
   Number of wire bits:            221
   Number of public wires:           5
   Number of public wire bits:      28
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                202
     $_AND_                         13
     $_AOI3_                        25
     $_AOI4_                         2
     $_MUX_                          9
     $_NAND_                        21
     $_NOR_                         29
     $_NOT_                         24
     $_OAI3_                        21
     $_OAI4_                         9
     $_OR_                          20
     $_XNOR_                        19
     $_XOR_                         10
```

Violation in input rule

```
1  module ALU_20IN (input [7:0]A, B, input [2:0]f, output reg[7:0]w, output reg c);
2      always @ (A, B) begin
3      w = 8'b0;
4      c = 1'b0;
5      case (f)
6          3'b000:  {c,w} = A + B;
7          3'b001:  {c,w} = A - B;
8          3'b010:  w = A;
9          3'b011:  w = A&B;
10         3'b100:  w = A|B;
11         3'b101:  w = ~B + 1;
12         3'b110:  w = A;
13         3'b111:  w = ~A;
14         default: w = 8'b0;
15     endcase
16     end
17 endmodule
```

```
=== ALU_20IN ===

   Number of wires:                198
   Number of wire bits:            221
   Number of public wires:           5
   Number of public wire bits:      28
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                202
     $_AND_                         13
     $_AOI3_                        25
     $_AOI4_                         2
     $_MUX_                          9
     $_NAND_                        21
     $_NOR_                         29
     $_NOT_                         24
     $_OAI3_                        21
     $_OAI4_                         9
     $_OR_                          20
     $_XNOR_                        19
     $_XOR_                         10
```

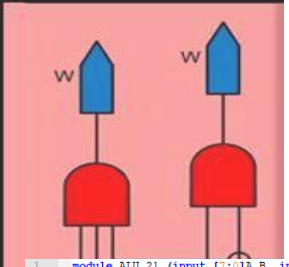
```
1  module ALU_20I (input [7:0]A, B, input [2:0]f, output reg[7:0]w, output reg c);
2      always @ (A, f) begin
3      w = 8'b0;
4      c = 1'b0;
5      case (f)
6          3'b000:  {c,w} = A + B;
7          3'b001:  {c,w} = A - B;
8          3'b010:  w = A;
9          3'b011:  w = A&B;
10         3'b100:  w = A|B;
11         3'b101:  w = ~B + 1;
12         3'b110:  w = A;
13         3'b111:  w = ~A;
14         default: w = 8'b0;
15     endcase
16     end
17 endmodule
```

```
=== ALU_20I ===

   Number of wires:                198
   Number of wire bits:            221
   Number of public wires:           5
   Number of public wire bits:      28
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                202
     $_AND_                         13
     $_AOI3_                        25
     $_AOI4_                         2
     $_MUX_                          9
     $_NAND_                        21
     $_NOR_                         29
     $_NOT_                         24
     $_OAI3_                        21
     $_OAI4_                         9
     $_OR_                          20
     $_XNOR_                        19
     $_XOR_                         10
```

```
1  module ALU_20IB (input [7:0]A, B, input [2:0]f, output reg[7:0]w, output reg c);
2      always @ (f) begin
3      w = 8'b0;
4      c = 1'b0;
5      case (f)
6          3'b000:  {c,w} = A + B;
7          3'b001:  {c,w} = A - B;
8          3'b010:  w = A;
9          3'b011:  w = A&B;
10         3'b100:  w = A|B;
11         3'b101:  w = ~B + 1;
12         3'b110:  w = A;
13         3'b111:  w = ~A;
14         default: w = 8'b0;
15     endcase
16     end
17 endmodule
```

```
=== ALU_20IB ===

   Number of wires:                198
   Number of wire bits:            221
   Number of public wires:           5
   Number of public wire bits:      28
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                202
     $_AND_                         13
     $_AOI3_                        25
     $_AOI4_                         2
     $_MUX_                          9
     $_NAND_                        21
     $_NOR_                         29
     $_NOT_                         24
     $_OAI3_                        21
     $_OAI4_                         9
     $_OR_                          20
     $_XNOR_                        19
     $_XOR_                         10
```

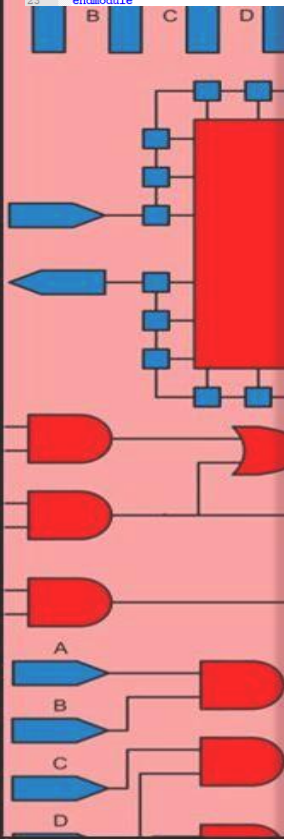
It will be removed




```
1  module ALU_21 (input [7:0]A,B, input [2:0]f, output reg[7:0]w, output reg c);
2    reg abars;
3    wire [8:0] t;
4    wire [7:0] Bin;
5    assign abars = (f == 3'b001) ? 1'b1 : 1'b0;
6    assign Bin = {8{abars}} ^ B;
7    assign t = A + Bin + abars;
8      always @ (A, B, f, t) begin
9      w = 8'b00000000;
10     c = 1'b0;
11     case (f)
12         3'b000:  {c,w}= t;
13         3'b001:  {c,w}= t;
14         3'b010:  w = A;
15         3'b011:  w = A&B;
16         3'b100:  w = A|B;
17         3'b101:  w = ~B + 1;
18         3'b110:  w = A;
19         3'b111:  w = ~A;
20         default: w = 8'b0;
21       endcase
22     end
23 endmodule
```

2

```
=== ALU_21 ===

  Number of wires:                148
  Number of wire bits:            171
  Number of public wires:           5
  Number of public wire bits:      28
  Number of memories:               0
  Number of memory bits:            0
  Number of processes:              0
  Number of cells:                152
    $_AND_                          4
    $_AOI3_                        12
    $_AOI4_                         6
    $_MUX_                          9
    $_NAND_                        21
    $_NOR_                         18
    $_NOT_                         18
    $_OAI3_                        16
    $_OAI4_                         5
    $_OR_                          14
    $_XNOR_                        24
    $_XOR_                          5
```



Violation in input rule

```
1  module ALU_21f (input [7:0]A,B, input [2:0]f, output reg[7:0]w, output reg c);
2    reg abars;
3    wire [8:0] t;
4    wire [7:0] Bin;
5    assign abars = (f == 3'b001) ? 1'b1 : 1'b0;
6    assign Bin = {8{abars}} ^ B;
7    assign t = A + Bin + abars;
8      always @ (A, B, t) begin
9      w = 8'b00000000;
10     c = 1'b0;
11     case (f)
12         3'b000:  {c,w}= t;
13         3'b001:  {c,w}= t;
14         3'b010:  w = A;
15         3'b011:  w = A&B;
16         3'b100:  w = A|B;
17         3'b101:  w = ~B + 1;
18         3'b110:  w = A;
19         3'b111:  w = ~A;
20         default: w = 8'b0;
21       endcase
22     end
23 endmodule
```

```
=== ALU_21f ===

  Number of wires:                148
  Number of wire bits:            171
  Number of public wires:           5
  Number of public wire bits:      28
  Number of memories:               0
  Number of memory bits:            0
  Number of processes:              0
  Number of cells:                152
    $_AND_                          4
    $_AOI3_                        12
    $_AOI4_                         6
    $_MUX_                          9
    $_NAND_                        21
    $_NOR_                         18
    $_NOT_                         18
    $_OAI3_                        16
    $_OAI4_                         5
    $_OR_                          14
    $_XNOR_                        24
    $_XOR_                          5
```

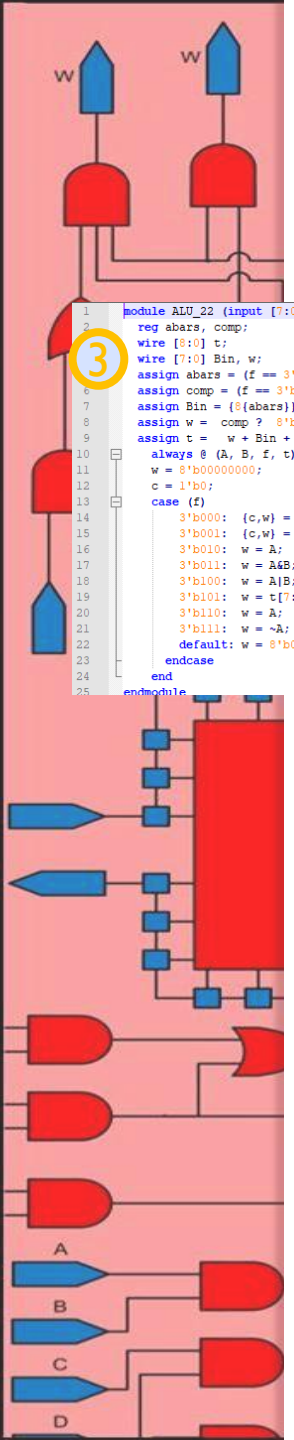
```
1  module ALU_21ft (input [7:0]A,B, input [2:0]f, output reg[7:0]w, output reg c);
2    reg abars;
3    wire [8:0] t;
4    wire [7:0] Bin;
5    assign abars = (f == 3'b001) ? 1'b1 :
6    assign Bin = {8{abars}} ^ B;
7    assign t = A + Bin + abars;
8      always @ (A, B) begin
9      w = 8'b00000000;
10     c = 1'b0;
11     case (f)
12         3'b000:  {c,w}= t;
13         3'b001:  {c,w}= t;
14         3'b010:  w = A;
15         3'b011:  w = A&B;
16         3'b100:  w = A|B;
17         3'b101:  w = ~B + 1;
18         3'b110:  w = A;
19         3'b111:  w = ~A;
20         default: w = 8'b0;
21       endcase
22     end
23 endmodule
```

```
=== ALU_21ft ===

  Number of wires:                148
  Number of wire bits:            171
  Number of public wires:           5
  Number of public wire bits:      28
  Number of memories:               0
  Number of memory bits:            0
  Number of processes:              0
  Number of cells:                152
    $_AND_                          4
    $_AOI3_                        12
    $_AOI4_                         6
    $_MUX_                          9
    $_NAND_                        21
    $_NOR_                         18
    $_NOT_                         18
    $_OAI3_                        16
    $_OAI4_                         5
    $_OR_                          14
    $_XNOR_                        24
    $_XOR_                          5
```

```
1  module ALU_21A (input [7:0]A,B, input [2:0]f, output reg[7:0]w, output reg c);
2    reg abars;
3    wire [8:0] t;
4    wire [7:0] Bin;
5    assign abars = (f == 3'b001) ? 1'b1 : 1'b0;
6    assign Bin = {8{abars}} ^ B;
7    assign t = A + Bin + abars;
8      always @ ( B, f, t) begin
9      w = 8'b00000000;
10     c = 1'b0;
11     case (f)
12         3'b000:  {c,w}= t;
13         3'b001:  {c,w}= t;
14         3'b010:  w = A;
15         3'b011:  w = A&B;
16         3'b100:  w = A|B;
17         3'b101:  w = ~B + 1;
18         3'b110:  w = A;
19         3'b111:  w = ~A;
20         default: w = 8'b0;
21       endcase
22     end
23 endmodule
```

```
=== ALU_21A ===

  Number of wires:                148
  Number of wire bits:            171
  Number of public wires:           5
  Number of public wire bits:      28
  Number of memories:               0
  Number of memory bits:            0
  Number of processes:              0
  Number of cells:                152
    $_AND_                          4
    $_AOI3_                        12
    $_AOI4_                         6
    $_MUX_                          9
    $_NAND_                        21
    $_NOR_                         18
    $_NOT_                         18
    $_OAI3_                        16
    $_OAI4_                         5
    $_OR_                          14
    $_XNOR_                        24
    $_XOR_                          5
```

It will be removed




```
module ALU_22 (input [7:0]A, B, input [2:0]f, output reg[7:0]w, output reg c);
  reg abars, comp;
  wire [8:0] t;
  wire [7:0] Bin, w;
  assign abars = (f == 3'b001 | f == 3'b101) ? 1'b1 : 1'b0;
  assign comp = (f == 3'b101) ? 1'b1 : 1'b0;
  assign Bin = {8{abars}} ^ B;
  assign w =  comp ?  8'b00000000: A;
  assign t =   w + Bin + abars;
    always @ (A, B, f, t) begin
    w = 8'b00000000;
    c = 1'b0;
    case (f)
        3'b000:  {c,w} = t;
        3'b001:  {c,w} = t;
        3'b010:  w = A;
        3'b011:  w = A&B;
        3'b100:  w = A|B;
        3'b101:  w = t[7:0];  // ~B +1
        3'b110:  w = A;
        3'b111:  w = ~A;
        default: w = 8'b0;
      endcase
    end
endmodule
```

```
=== ALU_22 ===

  Number of wires:                122
  Number of wire bits:            145
  Number of public wires:           5
  Number of public wire bits:      28
  Number of memories:               0
  Number of memory bits:            0
  Number of processes:              0
  Number of cells:                126
    $_AND_                         10
    $_AOI3_                        13
    $_MUX_                          9
    $_NAND_                         9
    $_NOR_                         21
    $_NOT_                         12
    $_OAI3_                        14
    $_OAI4_                         8
    $_OR_                           7
    $_XNOR_                        21
    $_XOR_                          2
```

Violation in input rule

```
module ALU_22B (input [7:0]A, B, input [2:0]f, output reg[7:0]w, output reg c);
  reg abars, comp;
  wire [8:0] t;
  wire [7:0] Bin, w;
  assign abars = (f == 3'b001 | f == 3'b101) ? 1'b1 : 1'b0;
  assign comp = (f == 3'b101) ? 1'b1 : 1'b0;
  assign Bin = {8{abars}} ^ B;
  assign w =  comp ?  8'b00000000: A;
  assign t =   w + Bin + abars;
    always @ (A, f, t) begin
    w = 8'b00000000;
    c = 1'b0;
    case (f)
        3'b000:  {c,w} = t;
        3'b001:  {c,w} = t;
        3'b010:  w = A;
        3'b011:  w = A&B;
        3'b100:  w = A|B;
        3'b101:  w = t[7:0];  // ~B +1
        3'b110:  w = A;
        3'b111:  w = ~A;
        default: w = 8'b0;
      endcase
    end
endmodule
```

```
=== ALU_22B ===

  Number of wires:                122
  Number of wire bits:            145
  Number of public wires:           5
  Number of public wire bits:      28
  Number of memories:               0
  Number of memory bits:            0
  Number of processes:              0
  Number of cells:                126
    $_AND_                         10
    $_AOI3_                        13
    $_MUX_                          9
    $_NAND_                         9
    $_NOR_                         21
    $_NOT_                         12
    $_OAI3_                        14
    $_OAI4_                         8
    $_OR_                           7
    $_XNOR_                        21
    $_XOR_                          2
```

```
module ALU_22fB (input [7:0]A, B, input [2:0]f, output reg[7:0]w, output reg c);
  reg abars, comp;
  wire [8:0] t;
  wire [7:0] Bin, w;
  assign abars = (f == 3'b001 | f == 3'b101) ? 1'b1 : 1'b0;
  assign comp = (f == 3'b101) ? 1'b1 : 1'b0;
  assign Bin = {8{abars}} ^ B;
  assign w =  comp ?  8'b00000000: A;
  assign t =   w + Bin + abars;
    always @ (A, t) begin
    w = 8'b00000000;
    c = 1'b0;
    case (f)
        3'b000:  {c,w} = t;
        3'b001:  {c,w} = t;
        3'b010:  w = A;
        3'b011:  w = A&B;
        3'b100:  w = A|B;
        3'b101:  w = t[7:0];  // ~B +1
        3'b110:  w = A;
        3'b111:  w = ~A;
        default: w = 8'b0;
      endcase
    end
endmodule
```

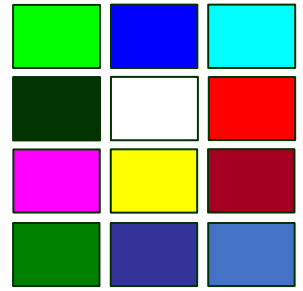
```
=== ALU_22fB ===

  Number of wires:                122
  Number of wire bits:            145
  Number of public wires:           5
  Number of public wire bits:      28
  Number of memories:               0
  Number of memory bits:            0
  Number of processes:              0
  Number of cells:                126
    $_AND_                         10
    $_AOI3_                        13
    $_MUX_                          9
    $_NAND_                         9
    $_NOR_                         21
    $_NOT_                         12
    $_OAI3_                        14
    $_OAI4_                         8
    $_OR_                           7
    $_XNOR_                        21
    $_XOR_                          2
```

```
module ALU_22t (input [7:0]A, B, input [2:0]f, output reg[7:0]w, output reg c);
  reg abars, comp;
  wire [8:0] t;
  wire [7:0] Bin, w;
  assign abars = (f == 3'b001 | f == 3'b101) ? 1'b1 : 1'b0;
  assign comp = (f == 3'b101) ? 1'b1 : 1'b0;
  assign Bin = {8{abars}} ^ B;
  assign w =  comp ?  8'b00000000: A;
  assign t =   w + Bin + abars;
    always @ (A, B, f) begin
    w = 8'b00000000;
    c = 1'b0;
    case (f)
        3'b000:  {c,w} = t;
        3'b001:  {c,w} = t;
        3'b010:  w = A;
        3'b011:  w = A&B;
        3'b100:  w = A|B;
        3'b101:  w = t[7:0];  // ~B +1
        3'b110:  w = A;
        3'b111:  w = ~A;
        default: w = 8'b0;
      endcase
    end
endmodule
```

```
=== ALU_22t ===

  Number of wires:                122
  Number of wire bits:            145
  Number of public wires:           5
  Number of public wire bits:      28
  Number of memories:               0
  Number of memory bits:            0
  Number of processes:              0
  Number of cells:                126
    $_AND_                         10
    $_AOI3_                        13
    $_MUX_                          9
    $_NAND_                         9
    $_NOR_                         21
    $_NOT_                         12
    $_OAI3_                        14
    $_OAI4_                         8
    $_OR_                           7
    $_XNOR_                        21
    $_XOR_                          2
```

Synthesis rules (reminder)

In previous example



What if we violate these rules

- No feedback

```
1  module ALU_17 (input [7:0]A,B, input [2:0]f, output reg[7:0]w,output reg c);
2    wire [8:0]t;
3    wire [7:0]Bin;
4    reg abars;
5    assign Bin = abars ? ~B : B;
6    //assign Bin = {8{abars}} ^ B;
7    assign t = A + Bin + abars;
8      always @ (A, B, f, t) begin
9          w = 8'b00000000;
10         c = 1'b0;
11         abars =1'b0;
12         case (f)
13            3'b000: begin abars =1'b0; {c,w}= t; end
14            3'b001: begin abars =1'b1; {c,w}= t; end
15            3'b010:  w = A;
16            3'b011:  w = A&B;
17            3'b100:  w = A|B;
18            3'b101:  w = ~A + 1;
19            3'b110:  w = A;
20            3'b111:  w = ~A;
21            default: w = 8'b0;
22         endcase
23     end
24 endmodule
```

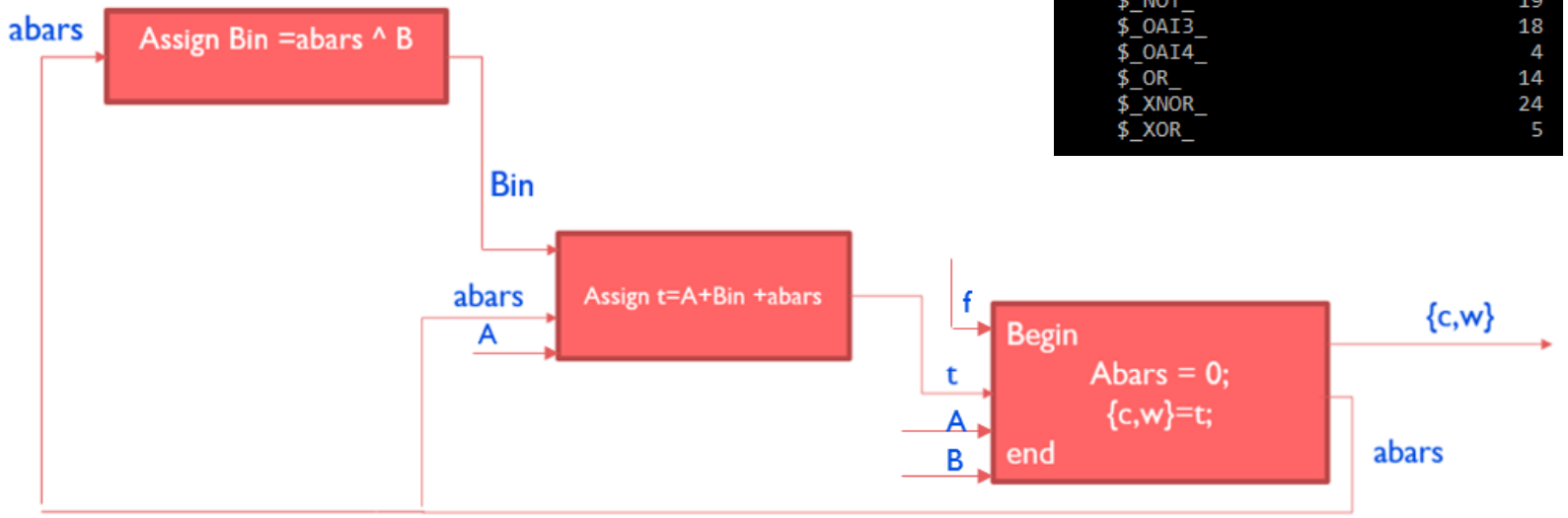
Feedback

Input rules

Outpu t rules

```
=== ALU_17 ===

   Number of wires:                149
   Number of wire bits:            172
   Number of public wires:           5
   Number of public wire bits:      28
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                153
     $_AND_                          6
     $_AOI3_                        11
     $_AOI4_                         6
     $_MUX_                         11
     $_NAND_                        13
     $_NOR_                         22
     $_NOT_                         19
     $_OAI3_                        18
     $_OAI4_                         4
     $_OR_                          14
     $_XNOR_                        24
     $_XOR_                          5
```

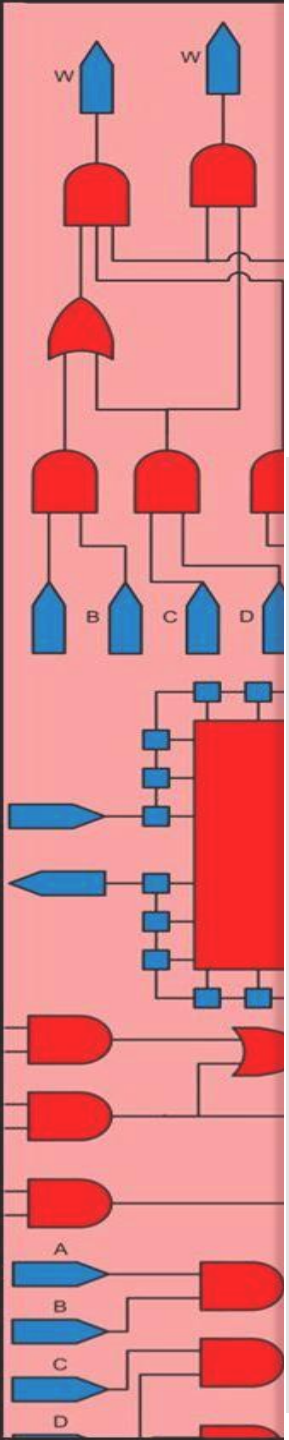
It's not acceptable



Combinational synthesis

Multiplication Vs. Shift

- 2 methods for multiplication by constant number




```
18    (* src = "mult.v:3" *)
19    output [3:0] out;
20    NOT _10_ (
21      .A(_01_),
22      .Y(_07_)
23    );
24    NOR _11_ (
25      .A(_03_),
26      .B(_07_),
27      .Y(_04_)
28    );
29    NOT _12_ (
30      .A(_03_),
31      .Y(_08_)
32    );
33    NAND _13_ (
34      .A(_08_),
35      .B(_01_),
36      .Y(_09_)
37    );
38    NAND _14_ (
39      .A(_03_),
40      .B(_07_),
41      .Y(_02_)
42    );
43    NAND _15_ (
44      .A(_02_),
45      .B(_09_),
46      .Y(_05_)
47    );
48    NOR _16_ (
49      .A(_08_),
50      .B(_07_),
51      .Y(_06_)
52    );
53    assign out[0] = a[0];
```

```
1   module mult (a, out);
2       input [1:0] a;
3       output  [3:0] out;
4
5       assign  out = a * 3;
6   endmodule
```

```
=== mult ===

   Number of wires:                 3
   Number of wire bits:             7
   Number of public wires:          2
   Number of public wire bits:      6
   Number of memories:              0
   Number of memory bits:           0
   Number of processes:             0
   Number of cells:                 4
     $_AND_                         1
     $_NOR_                         1
     $_NOT_                         1
     $_XOR_                         1
```

```
1   module mult_shift (a,out);
2       input [1:0] a;
3       output  [3:0] out;
4
5       assign  out = (a<<1) + a;
6   endmodule
```

```
=== mult_shift ===

   Number of wires:                 3
   Number of wire bits:             7
   Number of public wires:          2
   Number of public wire bits:      6
   Number of memories:              0
   Number of memory bits:           0
   Number of processes:             0
   Number of cells:                 4
     $_AND_                         1
     $_NOR_                         1
     $_NOT_                         1
     $_XOR_                         1
```

```
19    output [3:0] out;
20    NOT _10_ (
21      .A(_01_),
22      .Y(_07_)
23    );
24    NOT _11_ (
25      .A(_02_),
26      .Y(_08_)
27    );
28    NOR _12_ (
29      .A(_08_),
30      .B(_07_),
31      .Y(_04_)
32    );
33    NAND _13_ (
34      .A(_02_),
35      .B(_07_),
36      .Y(_09_)
37    );
38    NAND _14_ (
39      .A(_08_),
40      .B(_01_),
41      .Y(_03_)
42    );
43    NAND _15_ (
44      .A(_03_),
45      .B(_09_),
46      .Y(_05_)
47    );
48    NOR _16_ (
49      .A(_08_),
50      .B(_01_),
51      .Y(_06_)
52    );
53    assign out[0] = a[0];
```

Combinational synthesis

Multiplication by 5

- 2 designs for multiplication by 5

Yosys input

```
1   module mult5 (a,  out);
2       input [1:0] a;
3       output  [3:0] out;
4
5       assign  out = a * 5;
6   endmodule
```

```
=== mult5 ===

   Number of wires:                 2
   Number of wire bits:             6
   Number of public wires:          2
   Number of public wire bits:      6
   Number of memories:              0
   Number of memory bits:           0
   Number of processes:             0
   Number of cells:                 0
```

Yosys output

```
3   module mult5(a, out);
4     input [1:0] a;
5     output [3:0] out;
6     assign out = { a, a };
7   endmodule
```

Another Yosys input

```
1   module mult_shift5 (a,out);
2       input [1:0] a;
3       output  [3:0] out;
4
5       assign  out = (a<<2) + a;
6   endmodule
```

```
=== mult_shift5 ===

   Number of wires:                 2
   Number of wire bits:             6
   Number of public wires:          2
   Number of public wire bits:      6
   Number of memories:              0
   Number of memory bits:           0
   Number of processes:             0
   Number of cells:                 0
```

Yosys output

```
3   module mult_shift5(a, out);
4     input [1:0] a;
5     output [3:0] out;
6     assign out = { a, a };
7   endmodule
```

Combinational synthesis

Multiplication by 17

- multiplication by 17

```
1  module mult17 (a,out);
2      input [1:0] a;
3      output  [6:0] out;
4
5      assign  out = a * 17;
6  endmodule
7
```

Yosys input

```
=== mult17 ===

   Number of wires:                 2
   Number of wire bits:             9
   Number of public wires:          2
   Number of public wire bits:      9
   Number of memories:              0
   Number of memory bits:           0
   Number of processes:             0
   Number of cells:                 0
```

```
3  module mult17(a, out);
4    input [1:0] a;
5    output [6:0] out;
6
7    assign out = { 1'b0, a, 2'b00, a };
8
9  endmodule
```

Yosys output

Combinational synthesis

Multiply a by b

- multiplication a * b

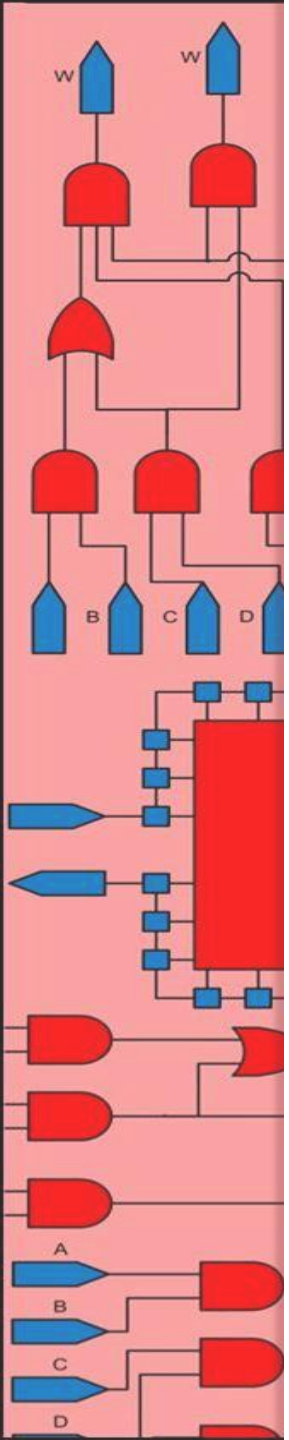
```
1   module multiple (a,b,out);
2       input [1:0] a,b;
3       output  [3:0] out;
4
5       assign  out = a * b;
6   endmodule
```

Yosys input

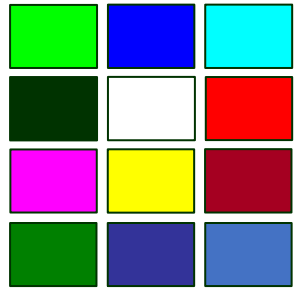
```
=== multiple ===

   Number of wires:                 7
   Number of wire bits:            12
   Number of public wires:          3
   Number of public wire bits:      8
   Number of memories:              0
   Number of memory bits:           0
   Number of processes:             0
   Number of cells:                 8
     $_AND_                         1
     $_NAND_                        3
     $_NOR_                         1
     $_OR_                          1
     $_XOR_                         2
```

Part of Yosys output

```
33      output [3:0] out;
34      NAND _21_ (
35        .A(_17_),
36        .B(_16_),
37        .Y(_20_)
38      );
39      NAND _22_ (
40        .A(_05_),
41        .B(_04_),
42        .Y(_07_)
43      );
44      NOR _23_ (
45        .A(_07_),
46        .B(_20_),
47        .Y(_18_)
48      );
49      NOT _24_ (
50        .A(_20_),
51        .Y(_19_)
52      );
53      NOT _25_ (
54        .A(_04_),
55        .Y(_09_)
56      );
57      NOT _26_ (
58        .A(_17_),
59        .Y(_10_)
60      );
61      NOR _27_ (
62        .A(_10_),
63        .B(_09_),
64        .Y(_11_)
65      );
66      NOT _28_ (
67        .A(_05_),
68        .Y(_12_)
      );
```



Outline



- A-1 Basics of Yosys
- A-2 Synthesize to gates
- A-3 Combinational RTL
- A-4 Sequential RTL
  - A-5-1 New library
  - A-5-2 D LATCH
  - A-5-3 D flip flop
  - A-5-4 Register
  - A-5-5 Shifter
  - A-5-6 Counter
  - A-5-7 Blocking vs non blocking
- A-5 FSM
- A-6 Complete RTL synthesis

Sequential synthesis

Introducing library and adding DFF cell

```
library(demo) {
  cell(BUF) {
    area: 6;
    pin(A) { direction: input; }
    pin(Y) { direction: output;
             function: "A"; }
  }
  cell(NOT) {
    area: 3;
    pin(A) { direction: input; }
    pin(Y) { direction: output;
             function: "A'"; }
  }
  cell(NAND) {
    area: 4;
    pin(A) { direction: input; }
    pin(B) { direction: input; }
    pin(Y) { direction: output;
             function: "(A*B)'"; }
  }
  cell(NOR) {
    area: 4;
    pin(A) { direction: input; }
    pin(B) { direction: input; }
    pin(Y) { direction: output;
             function: "(A+B)'"; }
  }
  cell(DFF) {
    area: 18;
    ff(IQ, IQN) { clocked_on: C;
                  next_state: D; }
    pin(C) { direction: input;
             clock: true; }
    pin(D) { direction: input; }
    pin(Q) { direction: output;
             function: "IQ"; }
  }
```

mycells.lib

mycells.v

```
`timescale 1ns/1ns
module NOT(A, Y);
input A;
output Y;
assign #3 Y = ~A;
endmodule

module NAND(A, B, Y);
input A, B;
output Y;
assign #5 Y = ~(A & B);
endmodule

module NOR(A, B, Y);
input A, B;
output Y;
assign #5 Y = ~(A | B);
endmodule

module DFF(C, D, Q);
input C, D;
output reg Q;
always @(posedge C)
    Q <= D;
endmodule
```

Example of library

There are some of cells in library

Mycells.v

```
//  |---v---|---v---|---v---|---v---|---v---|---v---|---v---|---v---|---v---|---v---|---v---|
//-
//-     $_BUF_ (A, Y)
//-
//- A buffer. This cell type is always optimized away by the opt_clean pass.
//-
//- Truth table:    A | Y
//-                 ---+---
//-                  0 | 0
//-                  1 | 1
//-
module \$_BUF_ (A, Y);
input A;
output Y;
assign Y = A;
endmodule
```

```
//  |---v---|---v---|---v---|---v---|---v---|---v---|---v---|---v---|---v---|---v---|---v---|
//-
//-     $_NOT_ (A, Y)
//-
//- An inverter gate.
//-
//- Truth table:    A | Y
//-                 ---+---
//-                  0 | 1
//-                  1 | 0
//-
module \$_NOT_ (A, Y);
input A;
output Y;
assign Y = ~A;
endmodule
```

```
//  |---v---|---v---|---v---|---v---|---v---|---v---|---v---|---v---|---v---|---v---|---v---|
//-
//-     $_XNOR_ (A, B, Y)
//-
//- A 2-input XNOR gate.
//-
//- Truth table:    A B | Y
//-                 -----+---
//-                  0 0 | 1
//-                  0 1 | 0
//-                  1 0 | 0
//-                  1 1 | 1
//-
module \$_XNOR_ (A, B, Y);
input A, B;
output Y;
assign Y = ~(A ^ B);
endmodule
```

```
//  |---v---|---v---|---v---|---v---|---v---|---v---|---v---|---v---|---v---|---v---|---v---|
//-
//-     $_ANDNOT_ (A, B, Y)
//-
//- A 2-input AND-NOT gate.
//-
//- Truth table:    A B | Y
//-                 -----+---
//-                  0 0 | 0
//-                  0 1 | 0
//-                  1 0 | 1
//-                  1 1 | 0
//-
module \$_ANDNOT_ (A, B, Y);
input A, B;
output Y;
assign Y = A & (~B);
endmodule
```

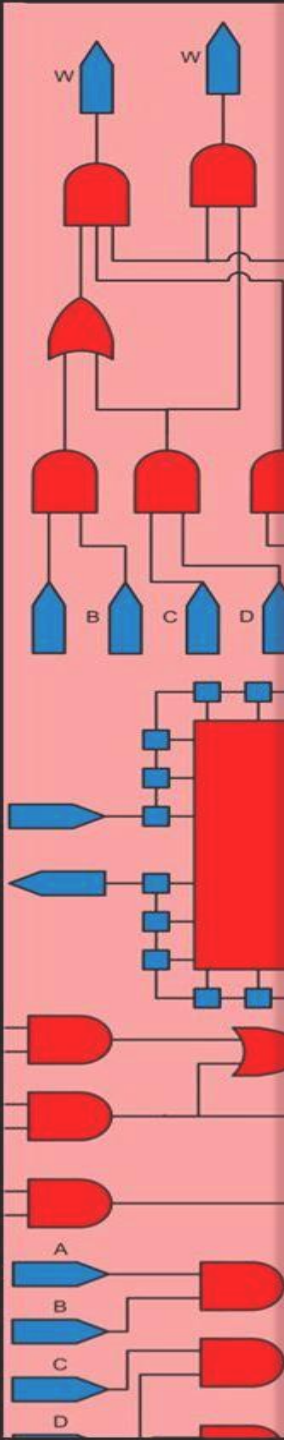
```
//- A 4-input MUX gate.
//-
//- Truth table:    A B C D S T | Y
//-                 -------------+---
//-                  a - - - 0 0 | a
//-                  - b - - 1 0 | b
//-                  - - c - 0 1 | c
//-                  - - - d 1 1 | d
//-
module \$_MUX4_ (A, B, C, D, S, T, Y);
input A, B, C, D, S, T;
output Y;
assign Y = T ? (S ? D : C) :
               (S ? B : A);
endmodule
```

```
//  |---v---|---v---|---v---|---v---|---v---|---v---|---v---|---v---|---v---|
//-
//-     $_MUX8_ (A, B, C, D, E, F, G, H, S, T, U, Y)
//-
//- An 8-input MUX gate.
//-
//- Truth table:    A B C D E F G H S T U | Y
//-                 -----------------------+---
//-                  a - - - - - - - 0 0 0 | a
//-                  - b - - - - - - 1 0 0 | b
//-                  - - c - - - - - 0 1 0 | c
//-                  - - - d - - - - 1 1 0 | d
//-                  - - - - e - - - 0 0 1 | e
//-                  - - - - - f - - 1 0 1 | f
//-                  - - - - - - g - 0 1 1 | g
//-                  - - - - - - - h 1 1 1 | h
//-
module \$_MUX8_ (A, B, C, D, E, F, G, H, S, T, U, Y);
input A, B, C, D, E, F, G, H, S, T, U;
output Y;
assign Y = U ? T ? (S ? H : G) :
                   (S ? F : E) :
               T ? (S ? D : C) :
                   (S ? B : A);
endmodule
```

```
//- A 3-input And-Or-Invert gate.
//-
//- Truth table:    A B C | Y
//-                 -------+---
//-                  0 0 0 | 1
//-                  0 0 1 | 0
//-                  0 1 0 | 1
//-                  0 1 1 | 0
//-                  1 0 0 | 1
//-                  1 0 1 | 0
//-                  1 1 0 | 0
//-                  1 1 1 | 0
//-
module \$_AOI3_ (A, B, C, Y);
input A, B, C;
output Y;
assign Y = ~((A & B) | C);
endmodule
```

```
//  |---v---|---v---|---v---|---v---|
//-
//-     $_OAI3_ (A, B, C, Y)
//-
//- A 3-input Or-And-Invert gate.
//-
//- Truth table:    A B C | Y
//-                 -------+---
//-                  0 0 0 | 1
//-                  0 0 1 | 1
//-                  0 1 0 | 1
//-                  0 1 1 | 0
//-                  1 0 0 | 1
//-                  1 0 1 | 0
//-                  1 1 0 | 1
//-                  1 1 1 | 0
//-
module \$_OAI3_ (A, B, C, Y);
input A, B, C;
output Y;
assign Y = ~((A | B) & C);
endmodule
```

Example of library




```
572  //  |---v---|---v---|---v---|---v---|---v---|---v---|---v---|---v---|---v--
573  //
574  //     $_DFF_P_ (D, C, Q)
575  //
576  // A positive edge D-type flip-flop.
577  //
578  // Truth table:    D C | Q
579  //                 -----+---
580  //                  d / | d
581  //                  - - | q
582  //
583  module \$_DFF_P_ (D, C, Q);
584  input D, C;
585  output reg Q;
586  always @(posedge C) begin
587      Q <= D;
588  end
589  endmodule
590
591  //  |---v---|---v---|---v---|---v---|---v---|---v---|---v---|---v---|---v--
592  //
593  //     $_DFFE_NN_ (D, C, E, Q)
594  //
595  // A negative edge D-type flip-flop with negative polarity enable.
596  //
597  // Truth table:    D C E | Q
598  //                 -------+---
599  //                  d \ 0 | d
600  //                  - - - | q
601  //
602  module \$_DFFE_NN_ (D, C, E, Q);
603  input D, C, E;
604  output reg Q;
605  always @(negedge C) begin
606      if (!E) Q <= D;
607  end
608  endmodule
```

Mycells.v

```
1065  //     $_DLATCH_N_ (E, D, Q)
1066  //
1067  // A negative enable D-type latch.
1068  //
1069  // Truth table:    E D | Q
1070  //                 -----+---
1071  //                  0 d | d
1072  //                  - - | q
1073  //
1074  module \$_DLATCH_N_ (E, D, Q);
1075  input E, D;
1076  output reg Q;
1077  always @* begin
1078      if (E == 0)
1079          Q <= D;
1080  end
1081  endmodule
1082
1083  //  |---v---|---v---|---v---|---v---
1084  //
1085  //     $_DLATCH_P_ (E, D, Q)
1086  //
1087  // A positive enable D-type latch.
1088  //
1089  // Truth table:    E D | Q
1090  //                 -----+---
1091  //                  1 d | d
1092  //                  - - | q
1093  //
1094  module \$_DLATCH_P_ (E, D, Q);
1095  input E, D;
1096  output reg Q;
1097  always @* begin
1098      if (E == 1)
1099          Q <= D;
1100  end
1101  endmodule
```

Mycells.lib

```
1   library(yosys_cells) {
2       cell(DFF_N) {
3           ff(IQ, IQN) {
4               clocked_on: "!C";
5               next_state: "D";
6           }
7           pin(D) { direction: input; }
8           pin(C) { direction: input; clock: true; }
9           pin(Q) { direction: output; function: "IQ"; }
10      }
11      cell(DFF_P) {
12          ff(IQ, IQN) {
13              clocked_on: "C";
14              next_state: "D";
15          }
16          pin(D) { direction: input; }
17          pin(C) { direction: input; clock: true; }
18          pin(Q) { direction: output; function: "IQ"; }
19      }
20      cell(DFF_NN0) {
21          ff(IQ, IQN) {
22              clocked_on: "!C";
23              next_state: "D";
24              clear: "!R";
25          }
26          pin(D) { direction: input; }
27          pin(R) { direction: input; }
28          pin(C) { direction: input; clock: true; }
29          pin(Q) { direction: output; function: "IQ"; }
30      }
31      cell(DFF_NN1) {
32          ff(IQ, IQN) {
33              clocked_on: "!C";
34              next_state: "D";
35              preset: "!R";
36          }
37          pin(D) { direction: input; }
38          pin(R) { direction: input; }
39          pin(C) { direction: input; clock: true; }
40          pin(Q) { direction: output; function: "IQ"; }
41      }
42      cell(DFF_NP0) {
43          ff(IQ, IQN) {
```

Sequential synthesis

D LATCH

Synthesis of D LATCH

```
1  module DLATCH (D, c, Q);
2      input D,c;
3      output reg Q;
4      always @ (c, D)
5      begin
6          if (c)
7              Q <= D;
8      end
9  endmodule
```

```
=== DLATCH ===

   Number of wires:                 3
   Number of wire bits:             3
   Number of public wires:          3
   Number of public wire bits:      3
   Number of memories:              0
   Number of memory bits:           0
   Number of processes:             0
   Number of cells:                 1
     $_DLATCH_P_                    1
```

```
3   (* top =  1  *)
4   (* src = "DLATCH.v:1" *)
5   module DLATCH(D, c, Q);
6     (* src = "DLATCH.v:2" *)
7     input D;
8     (* src = "DLATCH.v:3" *)
9     output Q;
10    (* src = "DLATCH.v:2" *)
11    input c;
12    \$_DLATCH_P_  _0_ (
13      .D(D),
14      .E(c),
15      .Q(Q)
16    );
17  endmodule
```

Sequential synthesis

D flip flop

Synthesis of D flip-flop

```
1   module Dff_s (Din,rst,clk,Qout);
2       input Din,rst,clk;
3       output reg Qout;
4       always @ (posedge clk)
5       begin
6           if (rst)
7               Qout <= 1'b0;
8           else
9               Qout <= Din;
10      end
11  endmodule
```

```
=== Dff_s ===

   Number of wires:                 6
   Number of wire bits:             6
   Number of public wires:          4
   Number of public wire bits:      4
   Number of memories:              0
   Number of memory bits:           0
   Number of processes:             0
   Number of cells:                 3
     $_DFF_P_                       1
     $_NOR_                         1
     $_NOT_                         1
```

```
5   module Dff_s(Din, rst, clk, Qout);
6     (* src = "Dff_s.v:4" *)
7     wire _0_;
8     wire _1_;
9     wire _2_;
10    wire _3_;
11    wire _4_;
12    wire _5_;
13    (* src = "Dff_s.v:2" *)
14    input Din;
15    (* src = "Dff_s.v:3" *)
16    output Qout;
17    (* src = "Dff_s.v:2" *)
18    input clk;
19    (* src = "Dff_s.v:2" *)
20    input rst;
21    NOT _6_ (
22      .A(_4_),
23      .Y(_5_)
24    );
25    NOR _7_ (
26      .A(_5_),
27      .B(_2_),
28      .Y(_3_)
29    );
30    DFF _8_ (
31      .C(clk),
32      .D(_0_),
33      .Q(Qout)
34    );
35    assign _2_ = rst;
36    assign _0_ = _3_;
37    assign _4_ = Din;
38  endmodule
```

```
1   module Dff (Din,rst,clk,Qout);
2       input Din,rst,clk;
3       output reg Qout;
4       always @ (posedge clk, posedge rst)
5       begin
6           if (rst)
7               Qout <= 1'b0;
8           else
9               Qout <= Din;
10      end
11  endmodule
```

```
=== Dff ===

   Number of wires:                 4
   Number of wire bits:             4
   Number of public wires:          4
   Number of public wire bits:      4
   Number of memories:              0
   Number of memory bits:           0
   Number of processes:             0
   Number of cells:                 1
     $_DFF_PP0_                     1
```

```
5   module Dff(Din, rst, clk, Qout);
6     (* src = "Dff.v:2" *)
7     input Din;
8     (* src = "Dff.v:3" *)
9     output Qout;
10    reg Qout;
11    (* src = "Dff.v:2" *)
12    input clk;
13    (* src = "Dff.v:2" *)
14    input rst;
15    (* src = "Dff.v:4" *)
16    always @(posedge clk or posedge rst)
17      if (rst)
18        Qout <= 0;
19      else
20        Qout <= Din;
21  endmodule
```

There is *DFF_PP0* cell in main library of Yosys

There is not *DFF_PP0* cell in our library of Yosys

Sequential synthesis

Register

Synthesis of Register

```
1  module register8_s (Parin, rst, clk, en, Parout);
2      input rst, clk, en;
3      input [7:0] Parin;
4      output [7:0] Parout;
5      reg [7:0] regin;
6      always @ (posedge clk)
7      begin
8          if (rst)
9              regin <= 8'd0;
10         else
11             regin <= (en) ? Parin: regin ;
12     end
13     assign Parout = regin;
14 endmodule
```

```
=== register8_s ===

   Number of wires:                 16
   Number of wire bits:             44
   Number of public wires:           6
   Number of public wire bits:      27
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 25
     $_AND_                          8
     $_DFF_P_                        8
     $_MUX_                          8
     $_NOT_                          1
```

```
349  DFF _131_ (
350    .C(clk),
351    .D(_000_[5]),
352    .Q(regin[5])
353  );
354  DFF _132_ (
355    .C(clk),
356    .D(_000_[6]),
357    .Q(regin[6])
358  );
359  DFF _133_ (
360    .C(clk),
361    .D(_000_[7]),
362    .Q(regin[7])
363  );
364  assign Parout = regin;
365  assign _010_ = regin[0];
```

```
1  module register8_a (Parin, rst, clk, en, Parout);
2      input rst, clk, en;
3      input [7:0] Parin;
4      output [7:0] Parout;
5      reg [7:0] regin;
6      always @ (posedge clk, posedge rst)
7      begin
8          if (rst)
9              regin <= 8'd0;
10         else
11             regin <= (en) ? Parin: regin ;
12     end
13     assign Parout = regin;
14 endmodule
```

```
=== register8_a ===

   Number of wires:                  7
   Number of wire bits:             35
   Number of public wires:           6
   Number of public wire bits:      27
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 16
     $_DFF_PP0_                      8
     $_MUX_                          8
```

There is *DFF_PP0* cell in materials of Yosys

```
217  always @(posedge clk or posedge rst)
218    if (rst)
219      regin[5] <= 0;
220    else
221      regin[5] <= _00_[5];
222  (* src = "register.v:6" *)
223  always @(posedge clk or posedge rst)
224    if (rst)
225      regin[6] <= 0;
226    else
227      regin[6] <= _00_[6];
228  (* src = "register.v:6" *)
229  always @(posedge clk or posedge rst)
230    if (rst)
231      regin[7] <= 0;
232    else
233      regin[7] <= _00_[7];
234  assign Parout = regin;
235  assign _01_ = regin[0];
236  assign _02_ = Parin[0];
```

There is not *DFF_PP0* cell in our library of Yosys

Sequential synthesis

Shifter

Synthesis of 4 bits Shifter

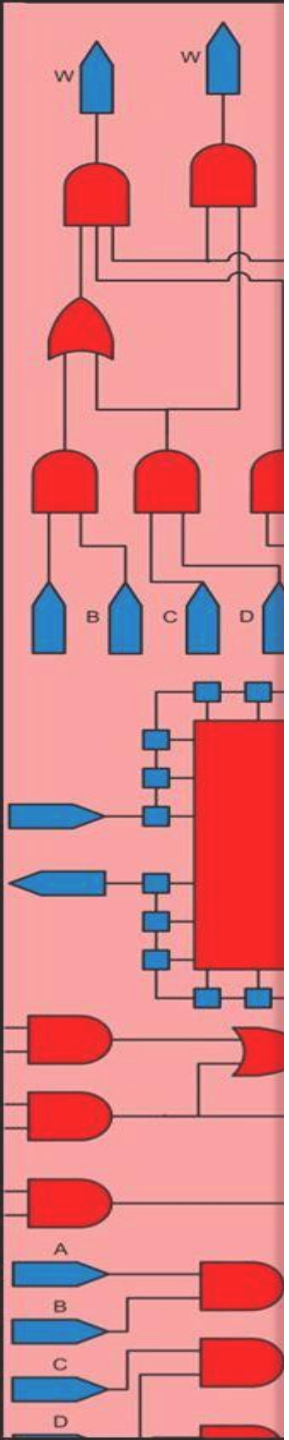
```
1   module Shift4Rl (PI, rst, clk, en, sel, siL, siR, soR, soL, PO);
2       input rst, clk, en;
3       input [3:0] PI;
4       input [1:0] sel;
5       input siR, siL;
6       output soR, soL;
7       output reg [3:0] PO;
8
9       always @ (posedge clk, posedge rst)
10      begin
11          if (rst)
12              PO <= 3'b0;
13          else
14              case(sel)
15                  2'b00: PO <= PO;
16                  2'b01: PO <= {siL, PO[3:1]};
17                  2'b10: PO <= {PO[2:0],siR};
18                  2'b11: PO <= PI;
19              endcase
20      end
21      assign soR = PO[0];
22      assign soL = PO[3];
23  endmodule
```

```
=== Shift4Rl ===

   Number of wires:                 28
   Number of wire bits:             38
   Number of public wires:          10
   Number of public wire bits:      17
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 25
     $_AND_                          1
     $_AOI4_                         4
     $_DFF_PP0_                      4
     $_MUX_                          4
     $_NAND_                         1
     $_NOR_                          1
     $_NOT_                          5
     $_OAI3_                         4
     $_OR_                           1
```

There is *DFF_PP0* cell in materials of Yosys

```
300   NOR _113_ (
301     .A(_070_),
302     .B(_064_),
303     .Y(_046_)
304   );
305   (* src = "Shift4Rl.v:9" *)
306   always @(posedge clk or posedge rst)
307     if (rst)
308       PO[0] <= 0;
309     else
310       PO[0] <= _000_[0];
311   (* src = "Shift4Rl.v:9" *)
312   always @(posedge clk or posedge rst)
313     if (rst)
314       PO[1] <= 0;
315     else
316       PO[1] <= _000_[1];
317   (* src = "Shift4Rl.v:9" *)
318   always @(posedge clk or posedge rst)
319     if (rst)
320       PO[2] <= 0;
321     else
322       PO[2] <= _000_[2];
323   (* src = "Shift4Rl.v:9" *)
324   always @(posedge clk or posedge rst)
325     if (rst)
326       PO[3] <= 0;
327     else
328       PO[3] <= _000_[3];
329   assign soL = PO[3];
```

There is not *DFF_PP0* cell in our library of Yosys



Sequential synthesis

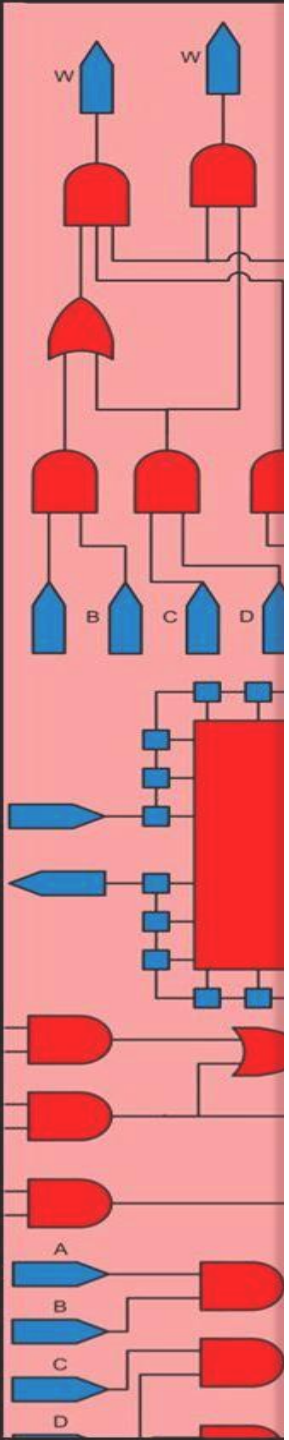
Counter

Synthesis of 4 bits Counter

Counter with incorrect Carry Out (5 DFFs)

```
1   module counter_5 (clk, rst, cin, co, out);
2       input clk, rst,cin;
3       output reg [3:0] out;
4       output reg co;
5
6       always @ (posedge clk)begin
7           if(rst)
8               out <= 4'd0;
9           else begin
10              out <= cin ? out + 1: 4'b0000;
11          end
12          co <= & {cin,out};
13      end
14  endmodule
```

```
=== counter_5 ===

   Number of wires:                 19
   Number of wire bits:             25
   Number of public wires:           5
   Number of public wire bits:       8
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 22
     $_AND_                          1
     $_DFF_P_                        5
     $_NAND_                         4
     $_NOR_                          6
     $_NOT_                          3
     $_XNOR_                         3
```



Sequential synthesis

Counter

Synthesis of 4 bits Counter

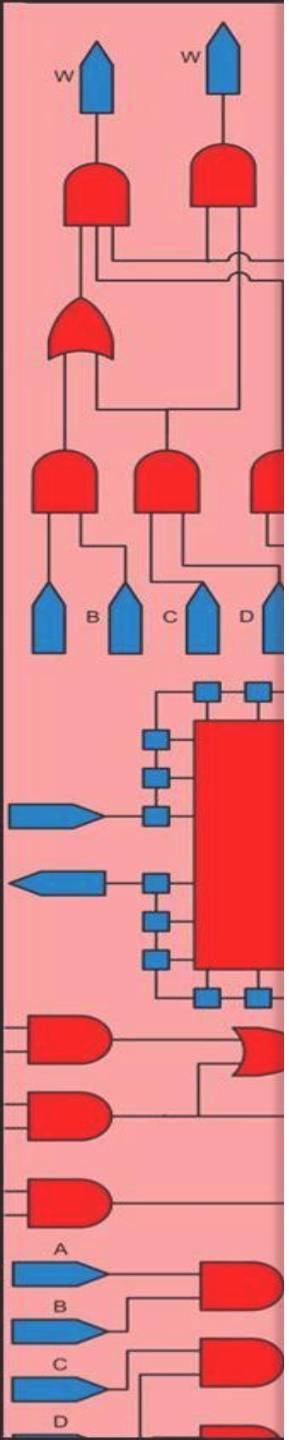
Counter with correct Carry Out (4 DFFs)

```
1   module counter_4 (clk, rst,cin, co, out);
2       input clk, rst,cin;
3       output reg [3:0] out;
4       output co;
5
6       always @ (posedge clk)begin
7           if(rst)
8           out <= 4'd0;
9           else begin
10          out <= cin ? out + 1: 4'b0000;
11          end
12      end
13      assign  co = & {cin,out};
14  endmodule
```

```
=== counter_4 ===

   Number of wires:                 18
   Number of wire bits:             24
   Number of public wires:           5
   Number of public wire bits:       8
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 21
     $_AND_                          1
     $_DFF_P_                        4
     $_NAND_                         4
     $_NOR_                          6
     $_NOT_                          3
     $_XNOR_                         3
```

Sequential synthesis

Introducing new library and adding 2 extra cells



mycells.lib

```
library(demo) {
  cell(BUF) {
    area: 6;
    pin(A) { direction: input; }
    pin(Y) { direction: output;
             function: "A"; }
  }
  cell(NOT) {
    area: 3;
    pin(A) { direction: input; }
    pin(Y) { direction: output;
             function: "A'"; }
  }
  cell(NAND) {
    area: 4;
    pin(A) { direction: input; }
    pin(B) { direction: input; }
    pin(Y) { direction: output;
             function: "(A*B)'"; }
  }
  cell(NOR) {
    area: 4;
    pin(A) { direction: input; }
    pin(B) { direction: input; }
    pin(Y) { direction: output;
             function: "(A+B)'"; }
  }
  cell(DFF) {
    area: 18;
    ff(IQ, IQN) { clocked_on: C;
                  next_state: D; }
    pin(C) { direction: input;
             clock: true; }
    pin(D) { direction: input; }
    pin(Q) { direction: output;
             function: "IQ"; }
  }
  cell(DFF_PP0) {
      ff(IQ, IQN) {
          clocked_on: "C";
          next_state: "D";
          clear: "R";
      }
      pin(D) { direction: input; }
      pin(R) { direction: input; }
      pin(C) { direction: input; clock: true; }
      pin(Q) { direction: output; function: "IQ"; }
  }
  cell(DFF_PP1) {
      ff(IQ, IQN) {
          clocked_on: "C";
          next_state: "D";
          preset: "R";
      }
      pin(D) { direction: input; }
      pin(R) { direction: input; }
      pin(C) { direction: input; clock: true; }
      pin(Q) { direction: output; function: "IQ"; }
  }
}
```

$_DFF_PP0_ (D, C, R, Q):

A positive edge D-type flip-flop with positive polarity reset.

$_DFF_PP1_ (D, C, R, Q):

A positive edge D-type flip-flop with positive polarity set..

mycells.v

```
`timescale 1ns/1ns
module NOT(A, Y);
input A;
output Y;
assign #3 Y = ~A;
endmodule

module NAND(A, B, Y);
input A, B;
output Y;
assign #5 Y = ~(A & B);
endmodule

module NOR(A, B, Y);
input A, B;
output Y;
assign #5 Y = ~(A | B);
endmodule

module DFF(C, D, Q);
input C, D;
output reg Q;
always @(posedge C)
    Q <= D;
endmodule

module DFF_PP0 (D, C, R, Q);
input D, C, R;
output reg Q;
always @(posedge C or posedge R) begin
    if (R == 1)
        Q <= 0;
    else
        Q <= D;
end
endmodule
module DFF_PP1 (D, C, R, Q);
input D, C, R;
output reg Q;
always @(posedge C or posedge R) begin
    if (R == 1)
        Q <= 1;
    else
        Q <= D;
end
endmodule
```

Sequential synthesis

D flip flop

Synthesis of D flip-flop (there are *DFF_PP0 & DFF_PP1 cells*)

```
1   module Dff_s (Din,rst,clk,Qout);
2       input Din,rst,clk;
3       output reg Qout;
4       always @ (posedge clk)
5       begin
6           if (rst)
7               Qout <= 1'b0;
8           else
9               Qout <= Din;
10      end
11  endmodule
```

```
=== Dff_s ===

   Number of wires:                 6
   Number of wire bits:             6
   Number of public wires:          4
   Number of public wire bits:      4
   Number of memories:              0
   Number of memory bits:           0
   Number of processes:             0
   Number of cells:                 3
     $_DFF_P_                       1
     $_NOR_                         1
     $_NOT_                         1
```

```
5   module Dff_s(Din, rst, clk, Qout);
6     (* src = "Dff_s.v:4" *)
7     wire _0_;
8     wire _1_;
9     wire _2_;
10    wire _3_;
11    wire _4_;
12    wire _5_;
13    (* src = "Dff_s.v:2" *)
14    input Din;
15    (* src = "Dff_s.v:3" *)
16    output Qout;
17    (* src = "Dff_s.v:2" *)
18    input clk;
19    (* src = "Dff_s.v:2" *)
20    input rst;
21    NOT _6_ (
22      .A(_4_),
23      .Y(_5_)
24    );
25    NOR _7_ (
26      .A(_5_),
27      .B(_2_),
28      .Y(_3_)
29    );
30    DFF _8_ (
31      .C(clk),
32      .D(_0_),
33      .Q(Qout)
34    );
35    assign _2_ = rst;
36    assign _0_ = _3_;
37    assign _4_ = Din;
38  endmodule
```

```
1   module Dff (Din,rst,clk,Qout);
2       input Din,rst,clk;
3       output reg Qout;
4       always @ (posedge clk, posedge rst)
5       begin
6           if (rst)
7               Qout <= 1'b0;
8           else
9               Qout <= Din;
10      end
11  endmodule
```

```
=== Dff ===

   Number of wires:                 4
   Number of wire bits:             4
   Number of public wires:          4
   Number of public wire bits:      4
   Number of memories:              0
   Number of memory bits:           0
   Number of processes:             0
   Number of cells:                 1
     $_DFF_PP0_                     1
```

```
5   module Dff(Din, rst, clk, Qout);
6     (* src = "Dff.v:2" *)
7     input Din;
8     (* src = "Dff.v:3" *)
9     output Qout;
10    (* src = "Dff.v:2" *)
11    input clk;
12    (* src = "Dff.v:2" *)
13    input rst;
14    DFF_PP0 _0_ (
15      .C(clk),
16      .D(Din),
17      .Q(Qout),
18      .R(rst)
19    );
20  endmodule
```

There is *DFF_PP0* cell in main library of Yosys

There is *DFF_PP0* cell in our library of Yosys

Sequential synthesis

Register

Synthesis of 8 bits Register with new library (there are *DFF_PP0 & DFF_PP1 cells*)

```
1  module register8_s (Parin, rst, clk, en, Parout);
2      input rst, clk, en;
3      input [7:0] Parin;
4      output [7:0] Parout;
5      reg [7:0] regin;
6      always @ (posedge clk)
7      begin
8          if (rst)
9              regin <= 8'd0;
10         else
11             regin <= (en) ? Parin: regin ;
12     end
13     assign Parout = regin;
14 endmodule
```

```
=== register8_s ===

   Number of wires:                 16
   Number of wire bits:             44
   Number of public wires:           6
   Number of public wire bits:      27
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 25
     $_AND_                          8
     $_DFF_P_                        8
     $_MUX_                          8
     $_NOT_                          1
```

```
349    DFF _131_ (
350      .C(clk),
351      .D(_000_[5]),
352      .Q(regin[5])
353    );
354    DFF _132_ (
355      .C(clk),
356      .D(_000_[6]),
357      .Q(regin[6])
358    );
359    DFF _133_ (
360      .C(clk),
361      .D(_000_[7]),
362      .Q(regin[7])
363    );
364    assign Parout = regin;
365    assign _010_ = regin[0];
```

```
1  module register8_a (Parin, rst, clk, en, Parout);
2      input rst, clk, en;
3      input [7:0] Parin;
4      output [7:0] Parout;
5      reg [7:0] regin;
6      always @ (posedge clk, posedge rst)
7      begin
8          if (rst)
9              regin <= 8'd0;
10         else
11             regin <= (en) ? Parin: regin ;
12     end
13     assign Parout = regin;
14 endmodule
```

```
=== register8_a ===

   Number of wires:                  7
   Number of wire bits:             35
   Number of public wires:           6
   Number of public wire bits:      27
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 16
     $_DFF_PP0_                      8
     $_MUX_                          8
```

```
216    DFF_PP0 _73_ (
217      .C(clk),
218      .D(_00_[5]),
219      .Q(regin[5]),
220      .R(rst)
221    );
222    DFF_PP0 _74_ (
223      .C(clk),
224      .D(_00_[6]),
225      .Q(regin[6]),
226      .R(rst)
227    );
228    DFF_PP0 _75_ (
229      .C(clk),
230      .D(_00_[7]),
231      .Q(regin[7]),
232      .R(rst)
233    );
234    assign Parout = regin;
235    assign _01_ = regin[1];
```

There is *DFF_PP0* cell in main library of Yosys

There is *DFF_PP0* cell in our library of Yosys

Sequential synthesis

Shifter

Synthesis of 4 bits Shifter with new library (there are *DFF_PP0 & DFF_PP1 cells*)

```
1  module Shift4Rl (PI, rst, clk, en, sel, siL, siR, soR, soL, PO);
2      input rst, clk, en;
3      input [3:0] PI;
4      input [1:0] sel;
5      input siR, siL;
6      output soR, soL;
7      output reg [3:0] PO;
8
9      always @ (posedge clk, posedge rst)
10     begin
11         if (rst)
12             PO <= 3'b0;
13         else
14             case(sel)
15                 2'b00: PO <= PO;
16                 2'b01: PO <= {siL, PO[3:1]};
17                 2'b10: PO <= {PO[2:0],siR};
18                 2'b11: PO <= PI;
19             endcase
20     end
21     assign soR = PO[0];
22     assign soL = PO[3];
23 endmodule
```

```
=== Shift4Rl ===

   Number of wires:                 28
   Number of wire bits:             38
   Number of public wires:          10
   Number of public wire bits:      17
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 25
     $_AND_                          1
     $_AOI4_                         4
     $_DFF_PP0_                      4
     $_MUX_                          4
     $_NAND_                         1
     $_NOR_                          1
     $_NOT_                          5
     $_OAI3_                         4
     $_OR_                           1
```

There is *DFF_PP0* cell in main library of Yosys

```
299  NOR _113_ (
300      .A(_069_),
301      .B(_063_),
302      .Y(_044_)
303  );
304  DFF_PP0 _114_ (
305      .C(clk),
306      .D(_000_[0]),
307      .Q(PO[0]),
308      .R(rst)
309  );
310  DFF_PP0 _115_ (
311      .C(clk),
312      .D(_000_[1]),
313      .Q(PO[1]),
314      .R(rst)
315  );
316  DFF_PP0 _116_ (
317      .C(clk),
318      .D(_000_[2]),
319      .Q(PO[2]),
320      .R(rst)
321  );
322  DFF_PP0 _117_ (
323      .C(clk),
324      .D(_000_[3]),
325      .Q(PO[3]),
326      .R(rst)
327  );
328  assign soL = PO[3];
```

There is *DFF_PP0* cell in our library of Yosys

Sequential synthesis

Non blocking assignment

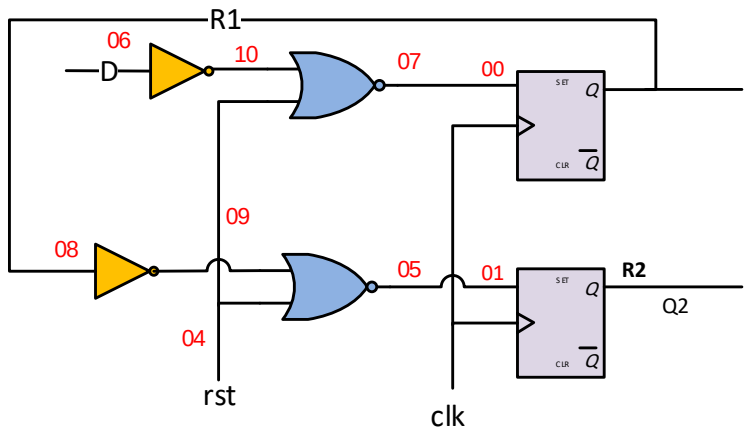
```
1  module nonblocking (clk, rst, D, Q1,Q2);
2      input clk,rst,D;
3      reg R1,R2;
4      output Q1,Q2;
5
6      always @(posedge clk) begin
7          if(rst) begin
8              R1<=1'b0; R2<=1'b0;
9          end
10         else begin
11             R1 <= D;
12             R2 <= R1;
13         end
14     end
15     assign Q1 = R1;
16     assign Q2 = R2;
17 endmodule
```

Synthesize

```
3  module nonblockingsynth(clk, rst, D, Q1, Q2);
4    wire _00_;
5    wire _01_;
6    wire _02_;
7    wire _03_;
8    wire _04_;
9    wire _05_;
10   wire _06_;
11   wire _07_;
12   wire _08_;
13   wire _09_;
14   wire _10_;
15   input D;
16   output Q1;
17   output Q2;
18   wire R1;
19   wire R2;
20   input clk;
21   input rst;
22   NOT _11_ (
23     .A(_08_),
24     .Y(_09_)
25   );
26   NOR _12_ (
27     .A(_09_),
28     .B(_04_),
29     .Y(_05_)
30   );
31   NOT _13_ (
32     .A(_06_),
33     .Y(_10_)
34   );
35   NOR _14_ (
36     .A(_10_),
37     .B(_04_),
38     .Y(_07_)
39   );
40   DFF _15_ (
41     .C(clk),
42     .D(_00_),
43     .Q(R1)
44   );
45   DFF _16_ (
46     .C(clk),
47     .D(_01_),
48     .Q(R2)
49   );
50   assign Q1 = R1;
51   assign Q2 = R2;
52   assign _04_ = rst;
53   assign _01_ = _05_;
54   assign _06_ = D;
55   assign _00_ = _07_;
56   assign _08_ = R1;
57 endmodule
```

A better coding style

```
19 module nonblocking (clk, rst, D, Q1,Q2);
20     input clk,rst,D;
21     reg R1,R2;
22     output Q1,Q2;
23
24     always @(posedge clk) begin
25         if(rst) begin
26             R1<=1'b0;
27         end
28         else begin
29             R1 <= D;
30
31         end
32     end
33     always @(posedge clk) begin
34         if(rst) begin
35             R2<=1'b0;
36         end
37         else begin
38             R2 <= R1;
39         end
40     end
41
42     assign Q1 = R1;
43     assign Q2 = R2;
44 endmodule
```

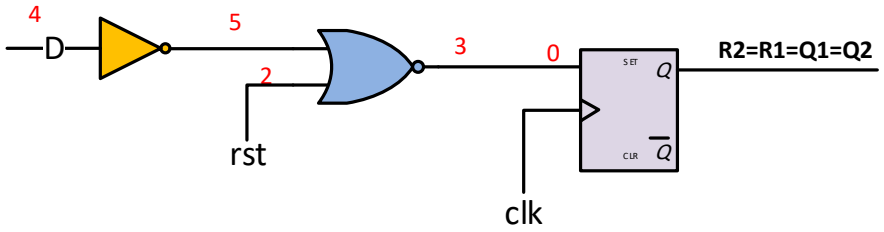


Sequential synthesis

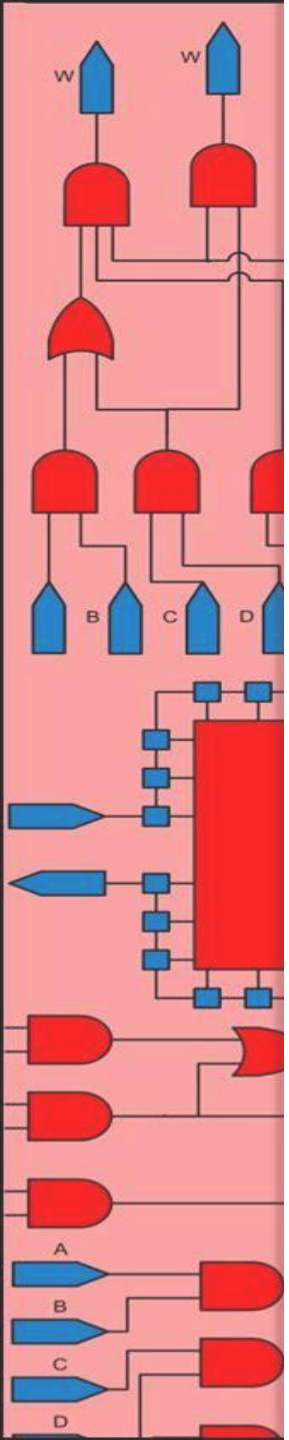
Blocking assignment

```
1  module blocking (clk, rst, D, Q1,Q2);
2      input clk,rst,D;
3      reg R1,R2;
4      output Q1,Q2;
5
6      always @(posedge clk) begin
7          if(rst) begin
8              R1=1'b0; R2=1'b0;
9          end
10         else begin
11             R1 = D;
12             R2 = R1;
13         end
14     end
15     assign Q1 = R1;
16     assign Q2 = R2;
17 endmodule
```

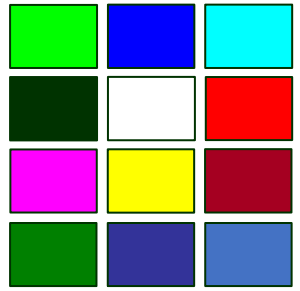
Synthesize

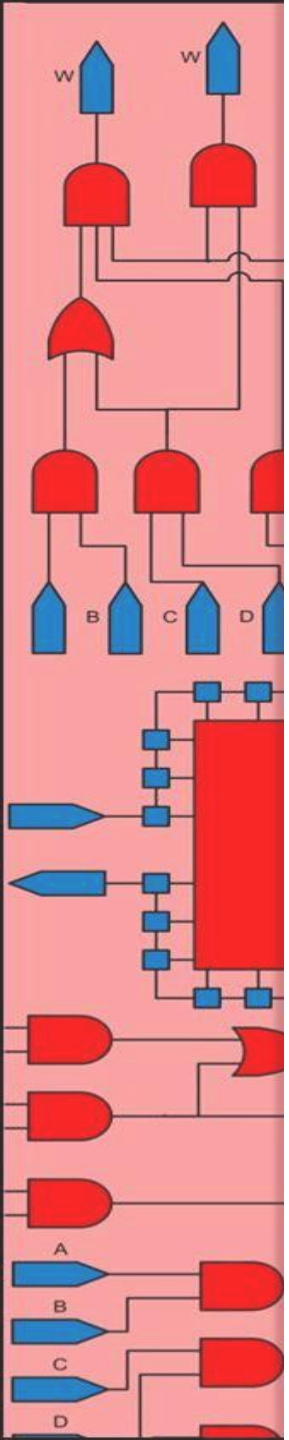



```
3  module blocking(clk, rst, D, Q1, Q2);
4    wire _0_;
5    wire _1_;
6    wire _2_;
7    wire _3_;
8    wire _4_;
9    wire _5_;
10   input D;
11   output Q1;
12   output Q2;
13   wire R1;
14   wire R2;
15   input clk;
16   input rst;
17   NOT _6_ (
18     .A(_4_),
19     .Y(_5_)
20   );
21   NOR _7_ (
22     .A(_5_),
23     .B(_2_),
24     .Y(_3_)
25   );
26   DFF _8_ (
27     .C(clk),
28     .D(_0_),
29     .Q(R2)
30   );
31   assign Q1 = R2;
32   assign Q2 = R2;
33   assign R1 = R2;
34   assign _2_ = rst;
35   assign _0_ = _3_;
36   assign _4_ = D;
37 endmodule
```



Outline





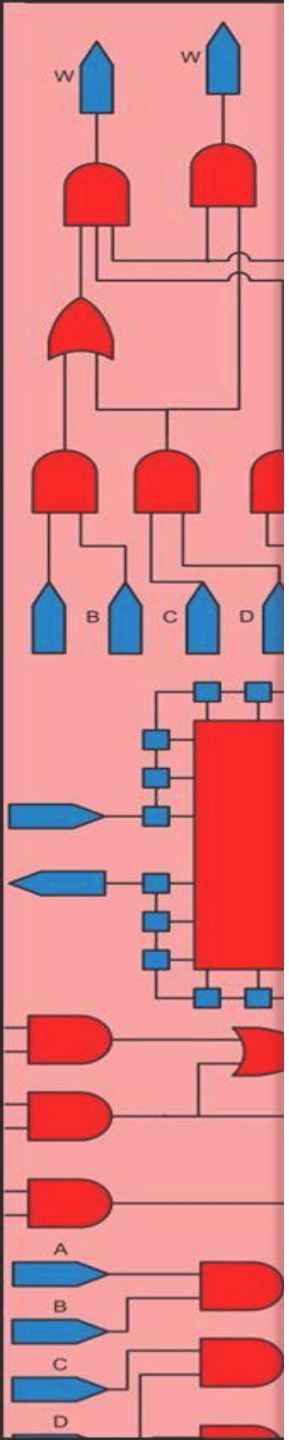
FSM of counter

- Synthesis of a 3 bits counter

```
1   module counter_sync (clk, rst,up, count);
2       input clk, rst,up;
3       output [2:0] count;
4       reg [2:0] ps,ns;
5
6       always @ (ps,up)begin
7           ns = 3'd0;
8           case(ps)
9               0: ns = up ? 3'd1: 3'd4;
10              1: ns = up ? 3'd2: 3'd0;
11              2: ns = up ? 3'd3: 3'd1;
12              3: ns = up ? 3'd4: 3'd2;
13              4: ns = up ? 3'd0: 3'd3;
14          endcase
15      end
16      always @ (posedge clk)begin
17          if(rst)
18          ps <= 3'd0;
19          else
20          ps <= ns;
21      end
22
23      assign  count = ps;
24  endmodule
```

```
207       .B(_60_),
208       .Y(_55_)
209     );
210     NAND _90_ (
211       .A(_55_),
212       .B(_33_),
213       .Y(_56_)
214     );
215     NAND _91_ (
216       .A(_56_),
217       .B(_28_),
218       .Y(_57_)
219     );
220     NOR _92_ (
221       .A(_57_),
222       .B(_54_),
223       .Y(_48_)
224     );
225     DFF _93_ (
226       .C(clk),
227       .D(_00_[0]),
228       .Q(ps[0])
229     );
230     DFF _94_ (
231       .C(clk),
232       .D(_00_[1]),
233       .Q(ps[1])
234     );
235     DFF _95_ (
236       .C(clk),
237       .D(_00_[2]),
238       .Q(ps[2])
239     );
240     assign count = ps;
241     assign _24_ = ps[2];
242     assign _36_ = ps[1];
243     assign  58  = rst;
```

```
=== counter_sync ===

   Number of wires:                 29
   Number of wire bits:             35
   Number of public wires:           5
   Number of public wire bits:       9
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 29
     $_AND_                          2
     $_AOI3_                         2
     $_DFF_P_                        3
     $_MUX_                          1
     $_NAND_                         2
     $_NOR_                          7
     $_NOT_                          5
     $_OAI3_                         3
     $_OR_                           4
```



Mealy Vs. Moore

- number of cells in Mealy vs. Moore

Mealy 1011 detector

```
always@(ps, j) begin
    ns = A;
    case (ps)
        A: ns= j ? B : A;
        B: ns= j ? B : C;
        C: ns= j ? D : A;
        D: ns= j ? B : C;
        default: ns= A;
    endcase
end
assign w= (ps==D) ? j : 1'b0;
always@(posedge clk, posedge rst) begin
    if (rst)
        ps <= A;
    else
        ps <= ns;
end
```

```
=== Mealy1011 ===

   Number of wires:                 13
   Number of wire bits:             16
   Number of public wires:           5
   Number of public wire bits:       8
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 13
     $_AND_                          2
     $_AOI3_                         1
     $_DFF_PP0_                      3
     $_DFF_PP1_                      1
     $_NOR_                          4
     $_NOT_                          2
```

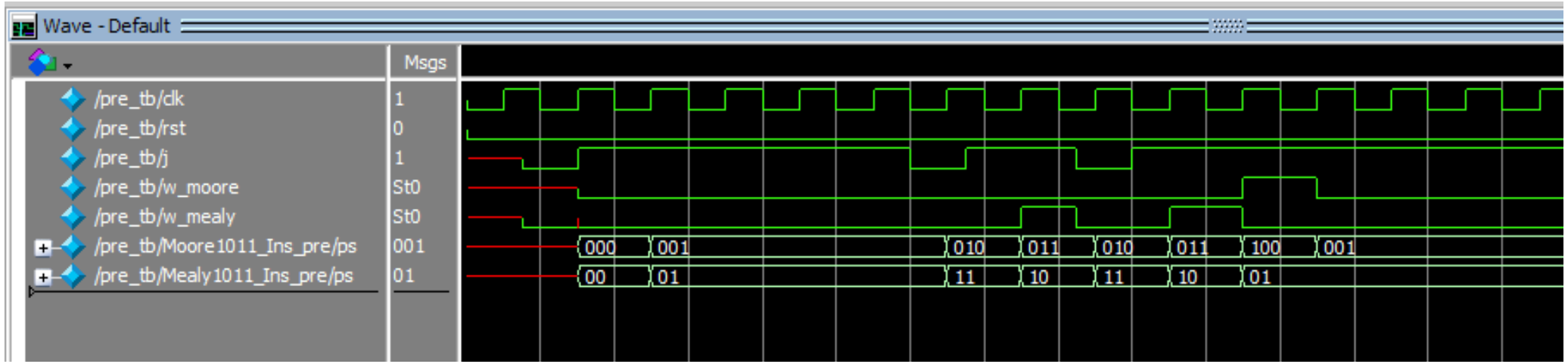
Moore 1011 detector

```
always@(ps, j) begin
    ns = A;
    case (ps)
        A: ns= j ? B : A;
        B: ns= j ? B : C;
        C: ns= j ? D : A;
        D: ns= j ? E : C;
        E: ns= j ? B : C;
        default: ns= A;
    endcase
end
assign w= (ps==E) ? 1'b1 : 1'b0;
always@(posedge clk, posedge rst) begin
    if (rst)
        ps <= A;
    else
        ps <= ns;
end
```

```
=== Moore1011 ===

   Number of wires:                 15
   Number of wire bits:             19
   Number of public wires:           5
   Number of public wire bits:       9
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 15
     $_AND_                          2
     $_AOI3_                         2
     $_DFF_PP0_                      4
     $_DFF_PP1_                      1
     $_NOR_                          3
     $_NOT_                          3
```

Mealy Vs. Moore

- Simulation of Mealy vs. Moore

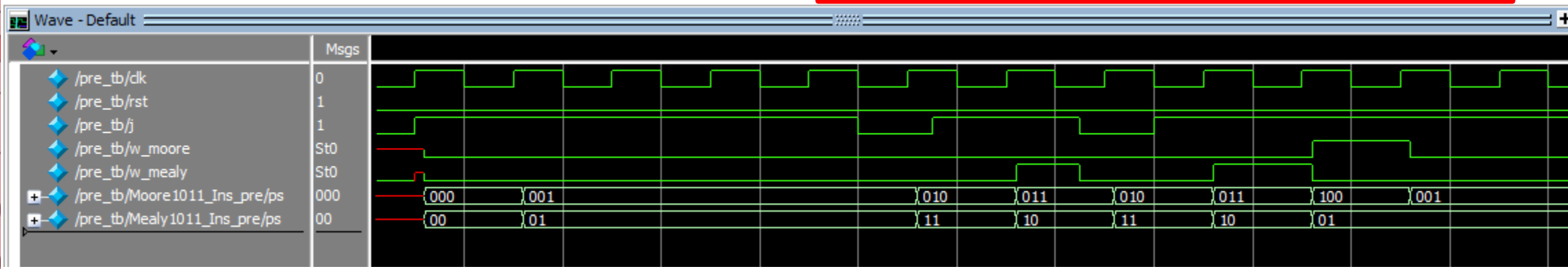


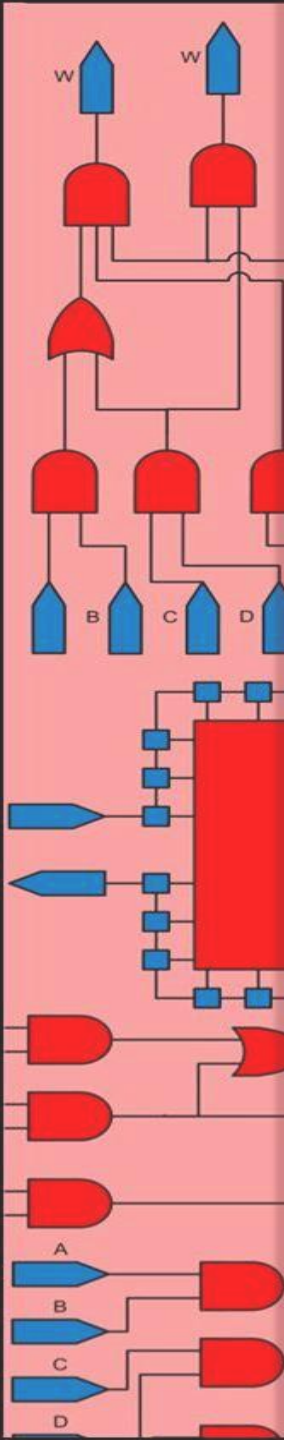
Mealy Vs. Moore

- Simulation of Mealy vs. Moore with delay on edge

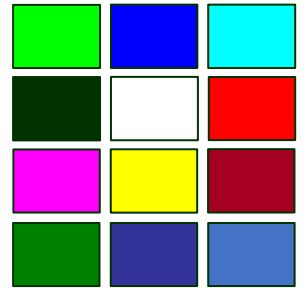
```
module Mealy1011 (input clk, rst, j, output w);
    reg [1:0] ns,ps;
    parameter [1:0] A=2'b00, B=2'b01, C=2'b11, D=2'b10;
    always@(ps,j) begin
        ns = A;
        case (ps)
            A: ns= j ? B : A;
            B: ns= j ? B : C;
            C: ns= j ? D : A;
            D: ns= j ? B : C;
            default: ns=A;
        endcase
    end
    assign w= (ps==D) ? j : 1'b0;
    always@ ( posedge clk, posedge rst) begin
        if (rst)
            ps <= A;
        else
            #2 ps <= ns;
    end
endmodule
```

```
module Moore1011 (input clk, rst, j, output w);
    reg [2:0] ns,ps;
    parameter [2:0] A=3'b000, B=3'b001, C=3'b010, D=3'b011, E=3'b100;
    always@(ps,j) begin
        ns = A;
        case (ps)
            A: ns= j ? B : A;
            B: ns= j ? B : C;
            C: ns= j ? D : A;
            D: ns= j ? E : C;
            E: ns= j ? B : C;
            default: ns=A;
        endcase
    end
    assign w= (ps==E) ? 1'b1 : 1'b0;
    always@ (posedge clk, posedge rst) begin
        if (rst)
            ps <= A;
        else
            #2 ps <= ns;
    end
endmodule
```



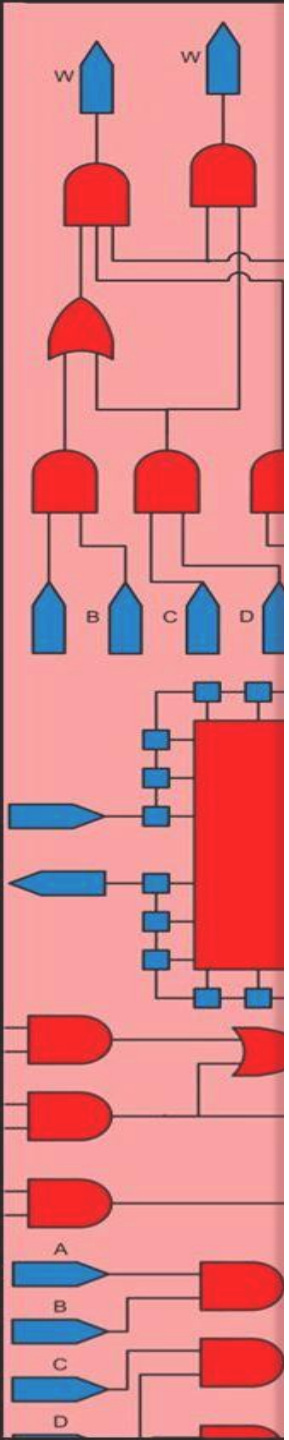


Example of LFSR



Token

- A token can be defined as a digitally encoded signature used to authenticate and authorize a user to access specific resources on a network.
- A token is always generated in the form of an OTP (One-Time Password), which depicts that it could only be used once and is generated randomly for every transaction.
- JWT (JSON Web Token) is used to provide open industry standard called RFC-7519, a way for two parties to communicate securely. JWT is commonly used for managing authorization.



Example of LFSR

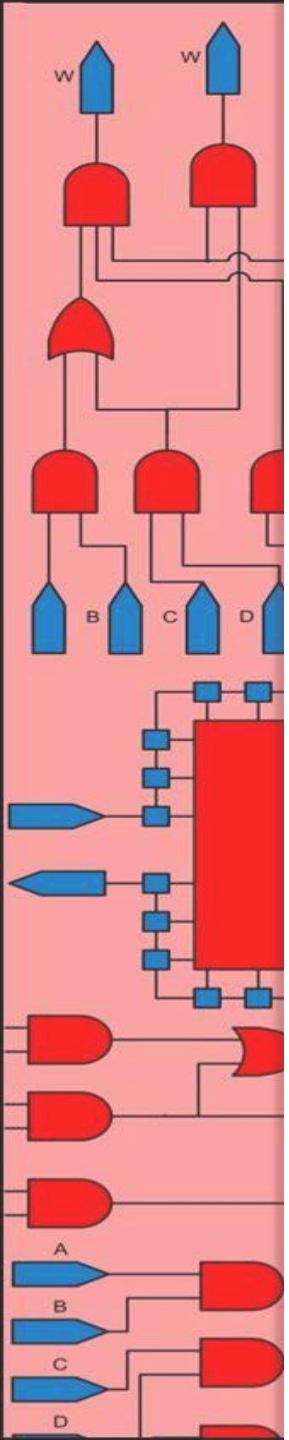
Token

Advantages

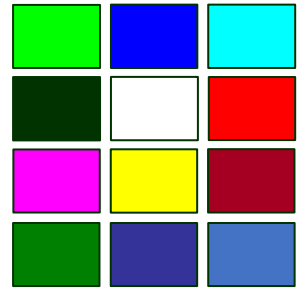
- **Token-based Authentication is more Scalable and Efficient**
- **Flexibility and Performance**

Disadvantages

- **Compromised Secret Key because using just one key**
- **Shorter Lifespan**



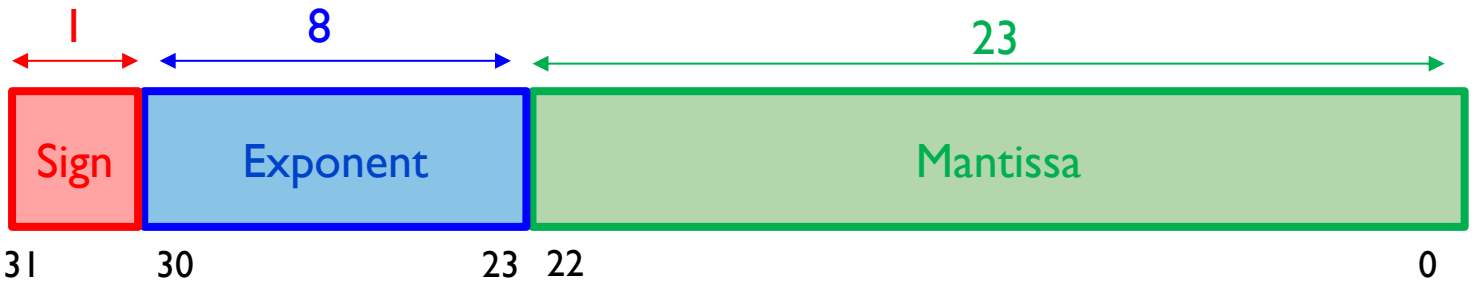
Outline



A-1 Basics of Yosys
A-2 Synthesize to gates
A-3 Combinational RTL
A-3 Memory elements
A-4 Timing
A-5 Sequential RTL
A-6 FSM
A-7 Complete RTL synthesis
- A-7-1 **IEEE** Standard **754 (**Floating-Point)
- A-7-2 Shift-Add multiplier
- A-7-3 Exponential
- A-7-4 Solution of HW10-2

IEEE 754 Floating-Point Standard

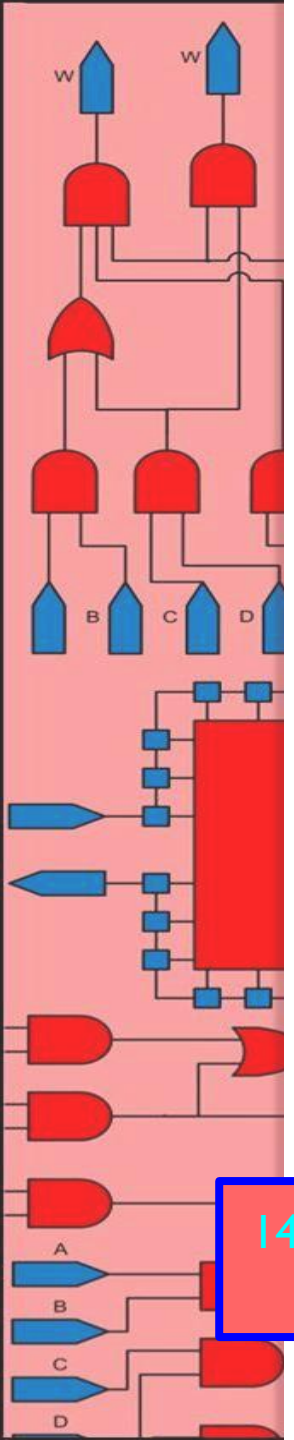
- Single Precision Format (32-bit)
  - Bias: 127
  - Range: [0:255]
- Double Precision Format (64-bit)
  - Bias: 1023
  - Range: [0:2047]

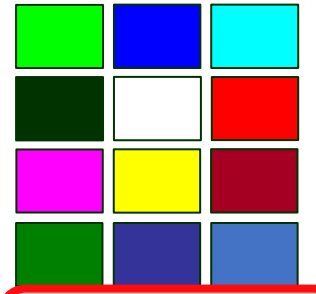


SP754N

$$(-1)^{sign} \times (1.mantissa) \times 2^{exponent-127}$$

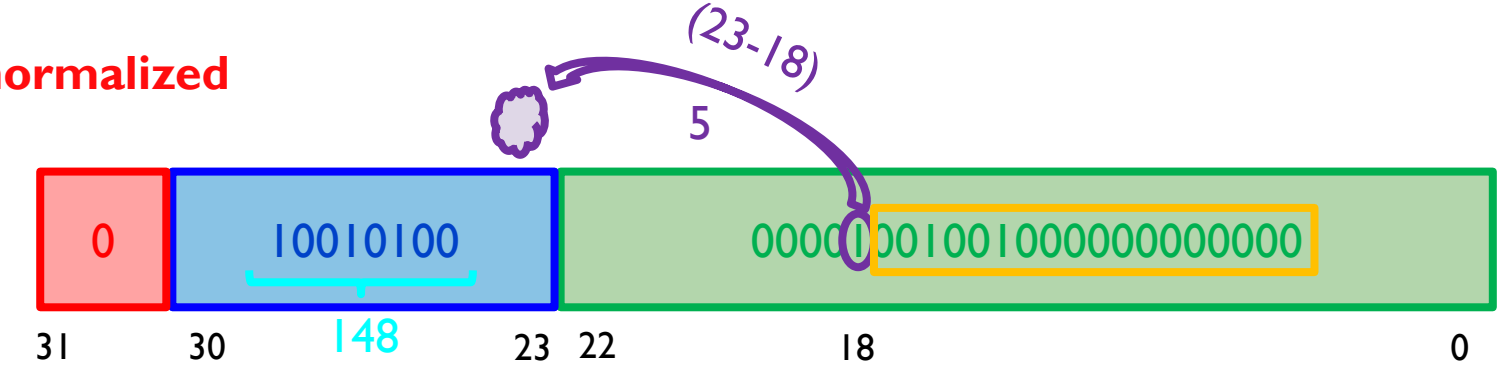
Floating-Point Normalization





Denormalized bias is 126

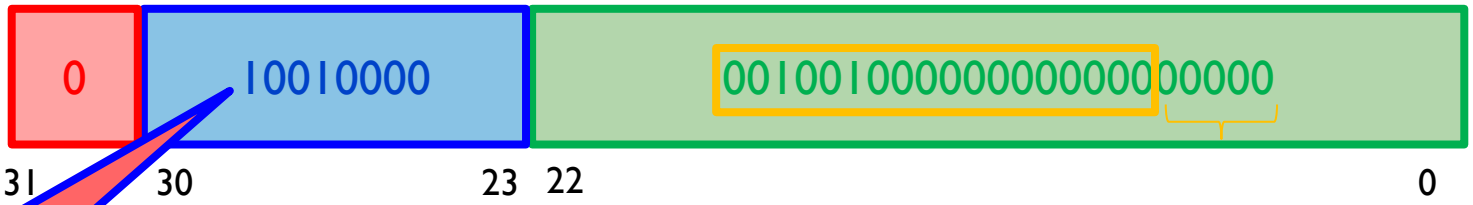
Denormalized



Represents 22

$$(-1)^0 \times (.00001001001000000000000) \times 2^{148-126}$$

Normalized



Normalized bias is 127

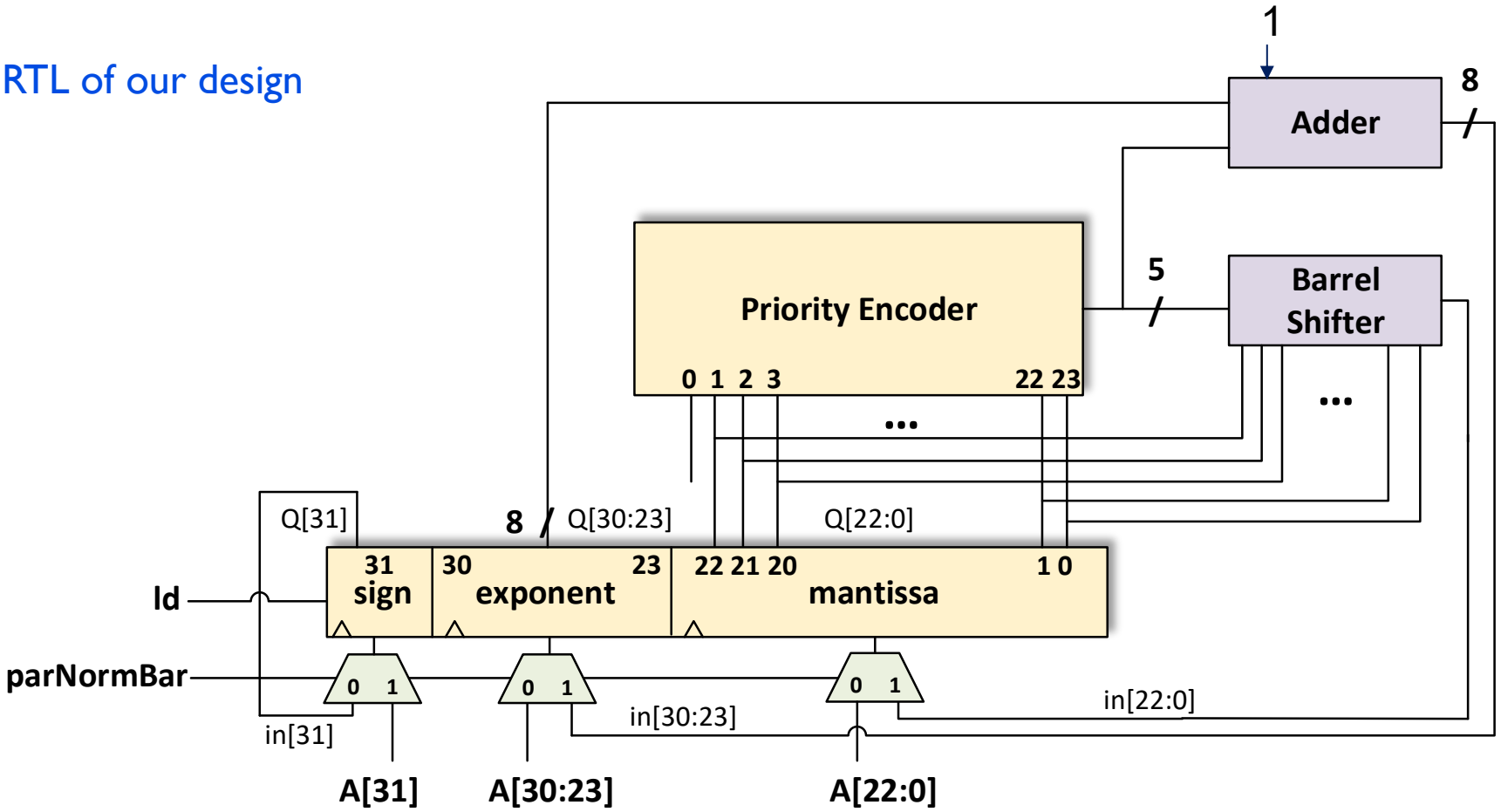
148 – 5 +1 = 144

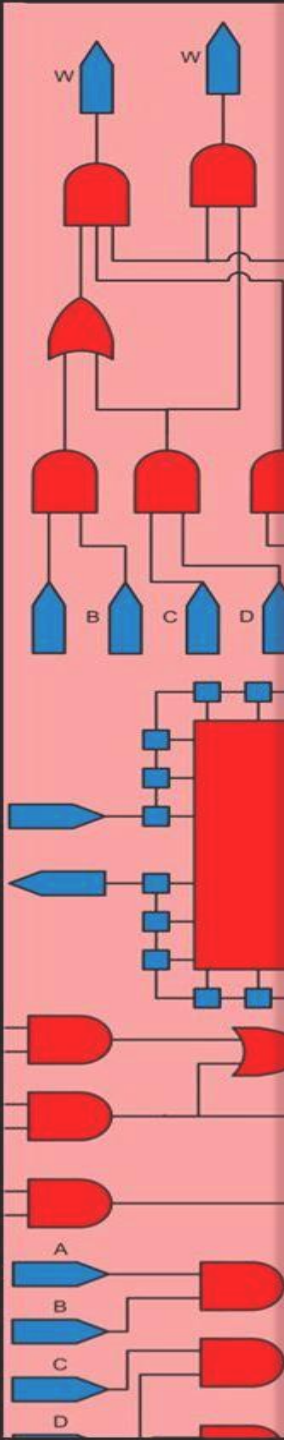
Represents 17

$$(-1)^0 \times (1.00100100000000000000000) \times 2^{144-127}$$

RTL of FP Normalization

- There is RTL of our design





Components of design

Register

```
16  module Register (clk, rst, ld, parNormBar, normIn, parallelIn, Q);
17  parameter  REG_WIDTH  = 8;
18
19  input  clk, rst, ld, parNormBar;
20  input  [REG_WIDTH-1:0] normIn, parallelIn;
21  output reg  [REG_WIDTH-1:0] Q;
22
23      always @(posedge clk or posedge rst)
24          if(rst == 1'b1)
25              Q <= 0;
26          else
27              if (ld == 1'b1)
28                  Q <= parNormBar ?  parallelIn : normIn;
29  endmodule
```

REG_WIDTH=1

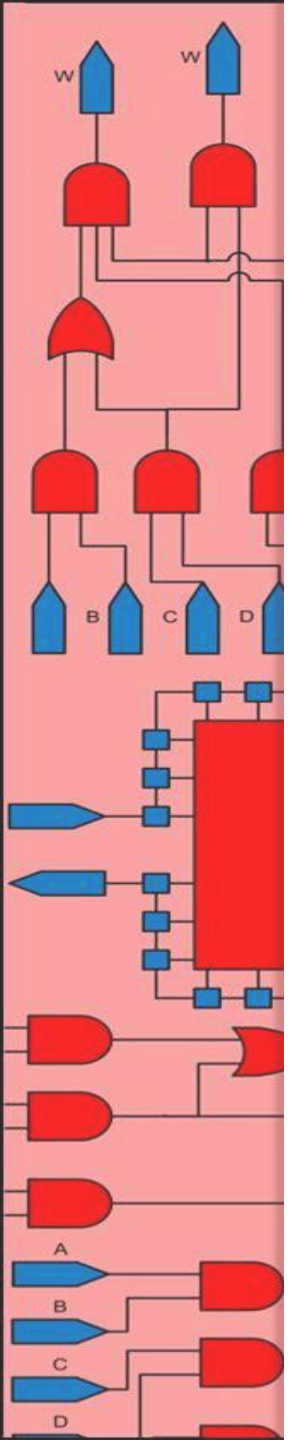
```
=== $paramod\Register\REG_WIDTH=1 ===

   Number of wires:                  9
   Number of wire bits:              9
   Number of public wires:           7
   Number of public wire bits:       7
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                  3
     $_DFF_PP0_                      1
     $_MUX_                          2
```

REG_WIDTH=8

```
=== $paramod\Register\REG_WIDTH=8 ===

   Number of wires:                 16
   Number of wire bits:             44
   Number of public wires:           7
   Number of public wire bits:      28
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 24
     $_DFF_PP0_                      8
     $_MUX_                         16
```

REG_WIDTH=23

```
=== $paramod\Register\REG_WIDTH=23 ===

   Number of wires:                 31
   Number of wire bits:            119
   Number of public wires:           7
   Number of public wire bits:      73
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 69
     $_DFF_PP0_                     23
     $_MUX_                         46
```

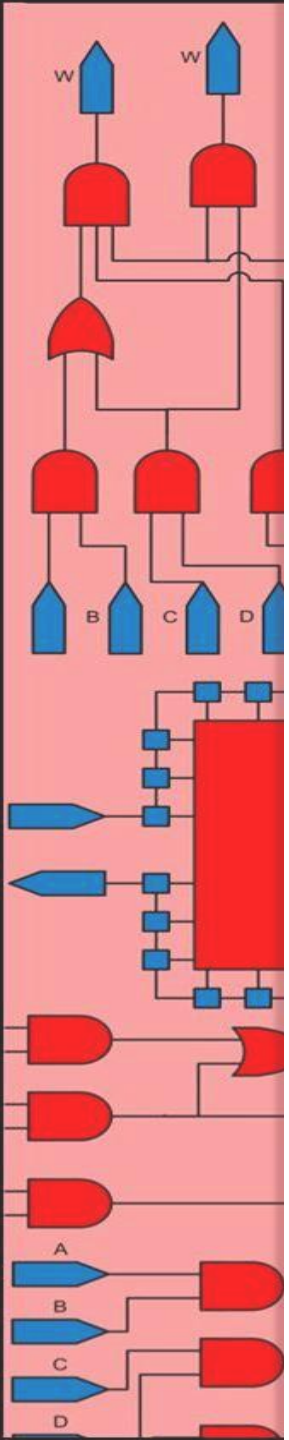


Components of design

Adder

```
50  module Adder (input  [7:0] A,
51                input  [4:0] B,
52                output [7:0] out);
53    assign out = A + (~B + 1) + 1;
54    endmodule
```

```
=== Adder ===

  Number of wires:                 42
  Number of wire bits:             60
  Number of public wires:           3
  Number of public wire bits:      21
  Number of memories:               0
  Number of memory bits:            0
  Number of processes:              0
  Number of cells:                 47
    $_AOI3_                         3
    $_NAND_                         2
    $_NOR_                          8
    $_NOT_                          8
    $_OAI3_                         3
    $_OR_                           4
    $_XNOR_                        13
    $_XOR_                          6
```

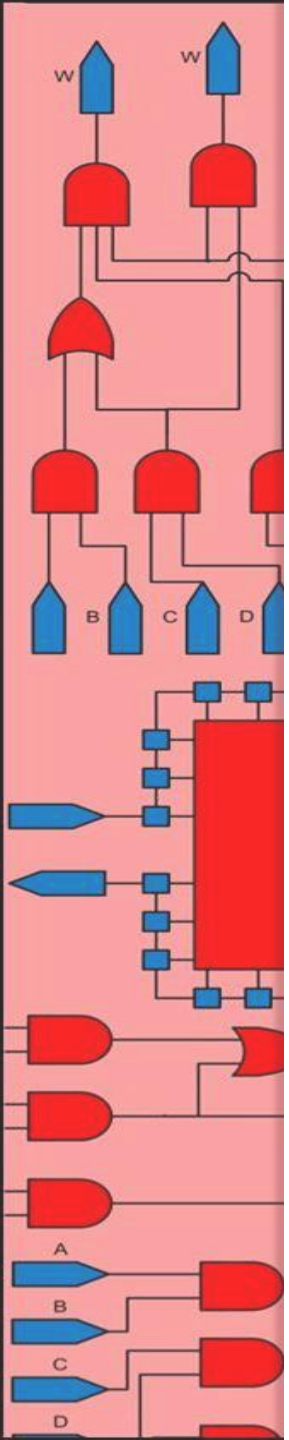


Components of design

Priority Encoder

```
1  module Priority_Encoder(input [23:0] encoder_in,
2                          output reg [4:0] binary_out);
3    integer i=0;
4
5      always @ (encoder_in) begin
6          binary_out = 5'b0;
7          for (i=0; i<24; i=i+1) begin
8              if (encoder_in[i] == 1'b1) begin
9                  binary_out = 5'd23 - i;
10             end
11         end
12     end
13 endmodule
```

```
=== Priority_Encoder ===

   Number of wires:                 90
   Number of wire bits:            117
   Number of public wires:           2
   Number of public wire bits:      29
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 93
     $_AND_                          3
     $_AOI3_                        18
     $_NAND_                        10
     $_NOR_                         28
     $_NOT_                         19
     $_OAI3_                         5
     $_OR_                          10
```



Components of design

Barrel shifter

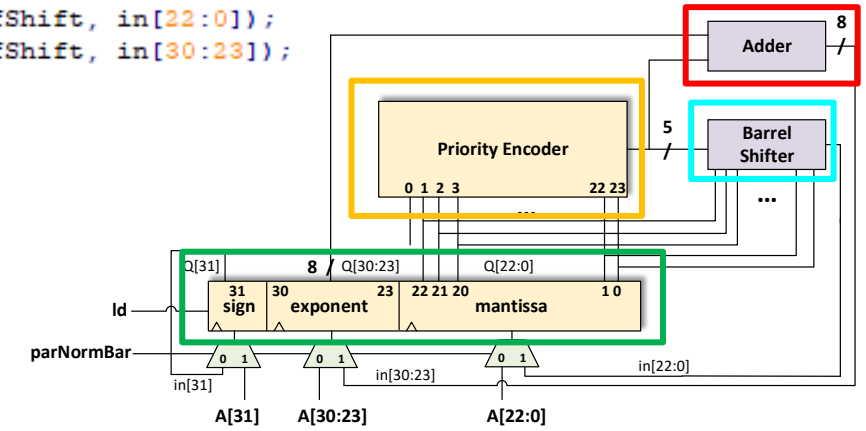
```
1  module Shifter_Left (input  [22:0] in,
2                       input  [4:0]  numOfShift,
3                       output [22:0] out);
4  assign out = {in << numOfShift};
5  endmodule
6
7
```

```
=== Shifter_Left ===

   Number of wires:                103
   Number of wire bits:            151
   Number of public wires:           3
   Number of public wire bits:      51
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                123
     $_AND_                         25
     $_MUX_                         82
     $_NAND_                         5
     $_NOR_                          4
     $_NOT_                          5
     $_OAI3_                         2
```

RTL of design

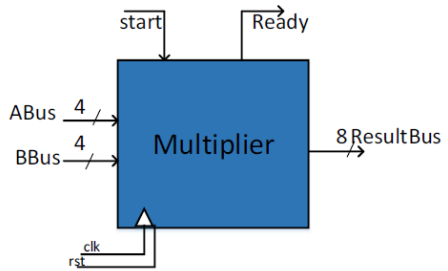
TOP module

```
1  module Datapath (input  clk, rst, ld, parNormBar,
2                   input  [31:0] parallelIn,
3                   output [31:0] out);
4    wire [4:0]  numOfShift;
5    wire [31:0] in, Q;
6
7      Register       #(1) SignReg        (clk, rst, ld, parNormBar, in[31],    parallelIn[31],    Q[31]);
8      Register       #(8) ExponentReg    (clk, rst, ld, parNormBar, in[30:23], parallelIn[30:23], Q[30:23]);
9      Register       #(23)MantissaReg (clk, rst, ld, parNormBar, in[22:0],  parallelIn[22:0],  Q[22:0]);
10     Priority_Encoder  PriorityEncoder({1'b0, Q[22:0]}, numOfShift);
11     Shifter_Left      ShiftLeft       (Q[22:0],  numOfShift, in[22:0]);
12     Adder             Add             (Q[30:23], numOfShift, in[30:23]);
13     assign in[31] = Q[31];
14     assign out = Q;
15 endmodule
```

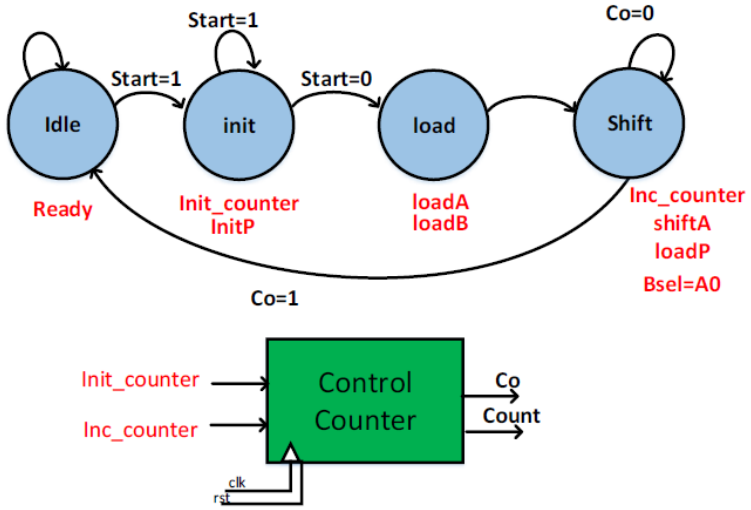



```
=== design hierarchy ===

   Datapath                              1
     $paramod\Register\REG_WIDTH=1       1
     $paramod\Register\REG_WIDTH=23      1
     $paramod\Register\REG_WIDTH=8       1
     Adder                             1
     Priority_Encoder                  1
     Shifter_Left                      1

   Number of wires:                  300
   Number of wire bits:              637
   Number of public wires:            38
   Number of public wire bits:       346
   Number of memories:                 0
   Number of memory bits:              0
   Number of processes:                0
   Number of cells:                  359
     $_AND_                           28
     $_AOI3_                          21
     $_DFF_PP0_                       32
     $_MUX_                          146
     $_NAND_                          17
     $_NOR_                           40
     $_NOT_                           32
     $_OAI3_                          10
     $_OR_                            14
     $_XNOR_                          13
     $_XOR_                            6
```

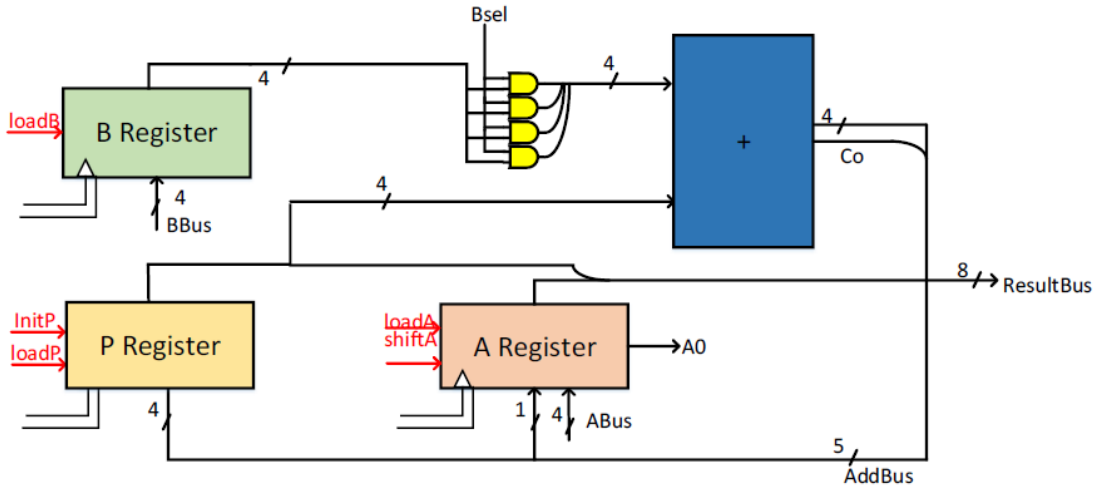
RTL of Shift-Add Multiplier



- There is Datapath and Controller of design



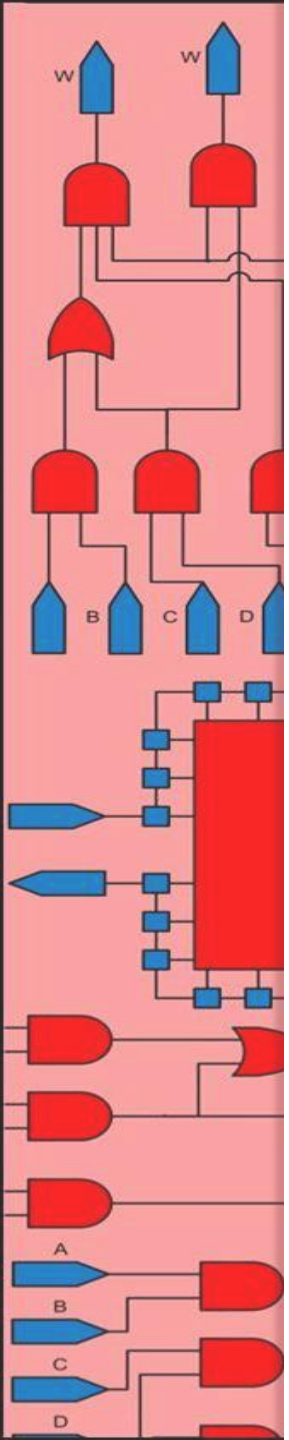
Controller

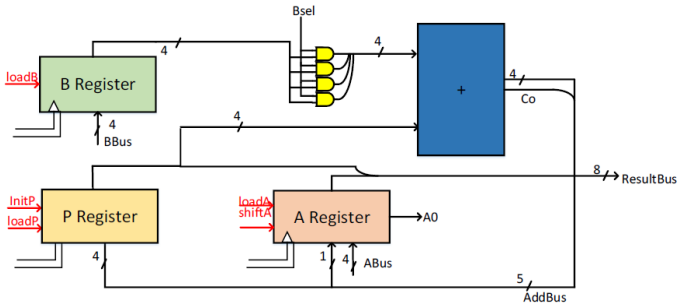


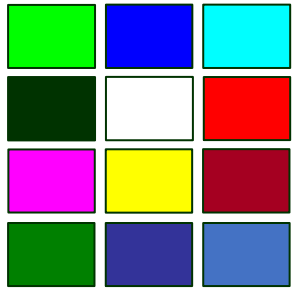
Datapath

Components of design

Datapath





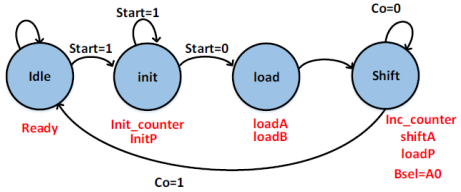


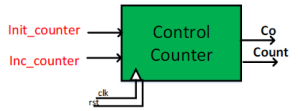
```
51  module MULTDP (input clk, rst, loadA, loadB, loadP, shiftA, InitP, Bsel,
52                 input [3:0] Abus, BBus, output [7:0] ResultBus, output A0);
53      reg  [3:0] Areg, Breg, Preg;
54      wire [3:0] B_AND;
55      wire [5:0] AddBus;
56
57          always @(posedge clk, posedge rst) begin
58              if (rst)
59                  Breg <= 4'b0;
60              else
61                  if(loadB)
62                      Breg <= BBus;
63          end
64          always @(posedge clk, posedge rst) begin
65              if (rst)
66                  Preg <= 4'b0;
67            else  begin
68                  if(InitP)
69                      Preg <= 4'b0;
70                  else
71                      if (loadP)
72                          Preg <= AddBus [4:1];
73                  end
74          end
75          always @(posedge clk, posedge rst) begin
76              if (rst)
77                  Areg <= 4'b0;
78            else begin
79                  if(loadA)
80                      Areg <= Abus;
81                  else
82                      if (shiftA)
83                          Areg <= {AddBus[0], Areg[3:1]};
84                  end
85          end
86          assign B_AND= Bsel ? Breg : 4'b0;
87          assign AddBus= B_AND + Preg;
88          assign ResultBus= {Preg, Areg};
89          assign A0=Areg[0];
90  endmodule
```

```
=== MULTDP ===

   Number of wires:                 52
   Number of wire bits:             83
   Number of public wires:          15
   Number of public wire bits:      37
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 58
     $_AND_                          5
     $_AOI3_                         2
     $_DFF_PP0_                     12
     $_MUX_                         16
     $_NAND_                         4
     $_NOR_                          5
     $_NOT_                          4
     $_OAI3_                         2
     $_XNOR_                         8
```

Components of design

Controller





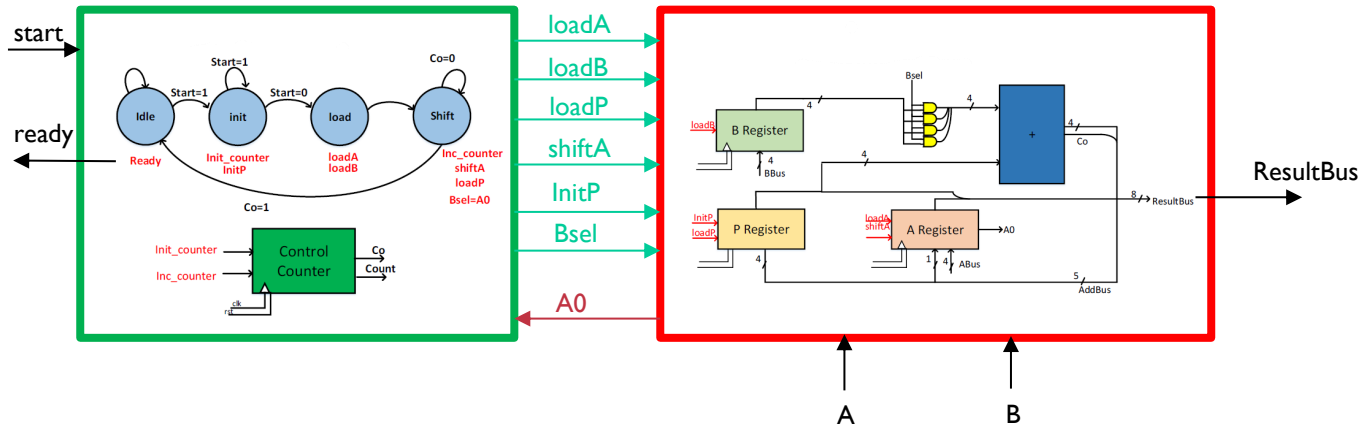

```
module MULTCU(input clk, rst, start, A0,  output reg loadA, shiftA, loadB, loadP, InitP, Bsel, ready);
    wire Co;
    reg Init_counter, Inc_counter;
    reg [1:0] pstate, nstate;
    reg [1:0] Count;
    parameter [1:0]
    Idle = 0, Init = 1, load = 2, shift = 3;

    always@(pstate,start, A0, Co) begin
        nstate=0;
        {loadA, shiftA, loadB, loadP, InitP, Bsel, ready}=7'b0;
        {Init_counter, Inc_counter}=2'b0;
            case(pstate)
                Idle:  begin nstate= start ? Init : Idle; ready=1'b1; end
                Init:  begin nstate= start ? Init : load; Init_counter=1'b1; InitP=1'b1; end
                load:  begin nstate= shift; loadA=1'b1; loadB=1'b1; end
                shift: begin nstate= Co ? Idle : shift; loadP=1'b1; shiftA=1'b1; Inc_counter=1'b1; Bsel=A0; end
                default: nstate=Idle;
            endcase
    end
    always@(posedge clk, posedge rst) begin
        if (rst)
        pstate <= Idle;
        else
        pstate <= nstate;
    end
    always@(posedge clk, posedge rst) begin
        if (rst) Count <= 2'b0;
        else
            if(Init_counter) Count <= 2'b0;
                    else
                        if (Inc_counter) Count <= Count +1;
    end
    assign Co= & Count;
endmodule
```

```
=== MULTCU ===

   Number of wires:                 33
   Number of wire bits:             38
   Number of public wires:          15
   Number of public wire bits:      19
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 29
     $_AND_                          3
     $_DFF_PP0_                      5
     $_DFF_PP1_                      1
     $_MUX_                          1
     $_NAND_                         1
     $_NOR_                          6
     $_NOT_                          5
     $_OAI3_                         2
     $_OR_                           3
     $_XNOR_                         2
```

RTL of design

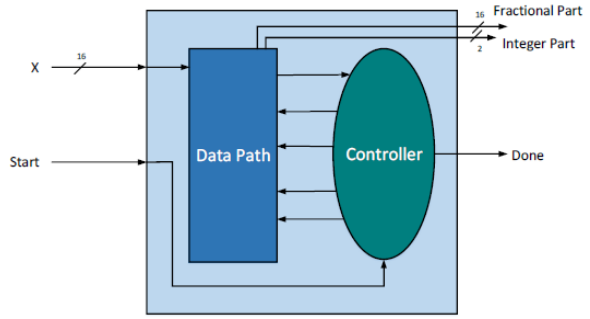
TOP module

```
1  module MULT_TOP (input clk, rst, start, input [3:0] A, B, output [7:0] ResultBus, output ready);
2  wire  A0;
3  wire loadA, shiftA, loadB, loadP, InitP, Bsel;
4
5   MULTDP  dp(clk, rst, loadA, loadB, loadP, shiftA, InitP, Bsel, A, B, ResultBus, A0);
6   MULTCU  cu(clk, rst, start, A0, loadA, shiftA, loadB, loadP, InitP, Bsel, ready);
7
8  endmodule
```

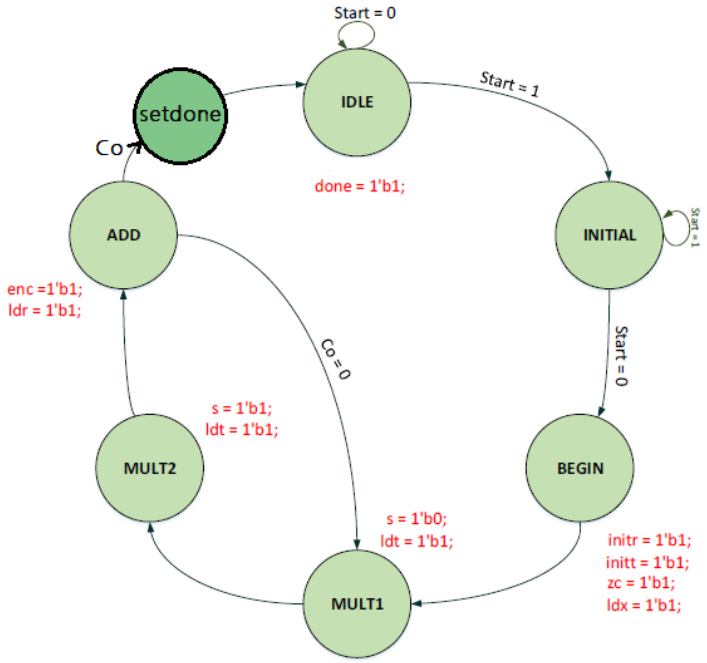



```
=== design hierarchy ===

  MULT_TOP                      1
    MULTCU                      1
    MULTDP                      1

  Number of wires:             99
  Number of wire bits:        148
  Number of public wires:      44
  Number of public wire bits:  83
  Number of memories:           0
  Number of memory bits:        0
  Number of processes:          0
  Number of cells:             87
    $_AND_                      8
    $_AOI3_                     2
    $_DFF_PP0_                 17
    $_DFF_PP1_                  1
    $_MUX_                     17
    $_NAND_                     5
    $_NOR_                     11
    $_NOT_                      9
    $_OAI3_                     4
    $_OR_                       3
    $_XNOR_                    10
```

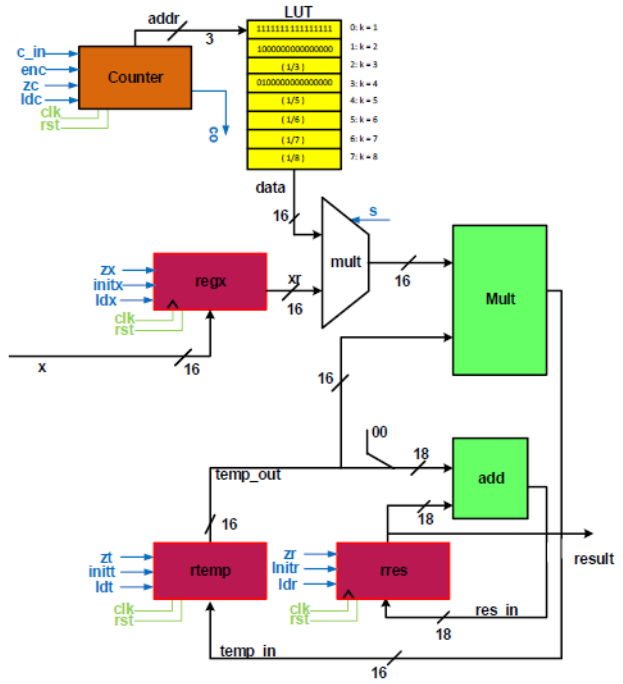
RTL of Exponential



- There is Datapath and Controller of design



Controller



Datapath

Components of design

Datapath

```
module datapath(input clk,rst, zx,initx,ldx, zt,initt,ldt, zr,initr,ldr, zc,ldc,enc, s,
        input [15:0] x, output co,output reg[17:0]result);
    wire [15:0] xptwo;
    wire [2:0] c_in = 0;
    wire [15:0] data,m_out,temp_in,ymux;
    reg [15:0] xptwor,temp_out,xr;
    wire [17:0] res_in;
    wire [3:0] adr;

    register regx(clk,rst,zx,initx,ldx,x,xr);

    counter count(clk,rst,zc,ldc,enc,c_in,adr,co);

    LUT lut(adr,data);

    mux2to1 mux(xr,data,s,m_out);

    multiplier mult(m_out,temp_out,temp_in);

    register rtemp(clk,rst,zt,initt,ldt,temp_in,temp_out);

    adder add(result,{2'b00,temp_out},res_in);

    register18 rres(clk,rst,zr,initr,ldr,res_in,result);
endmodule
```

16

4

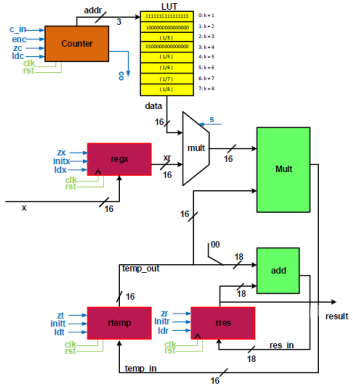
16

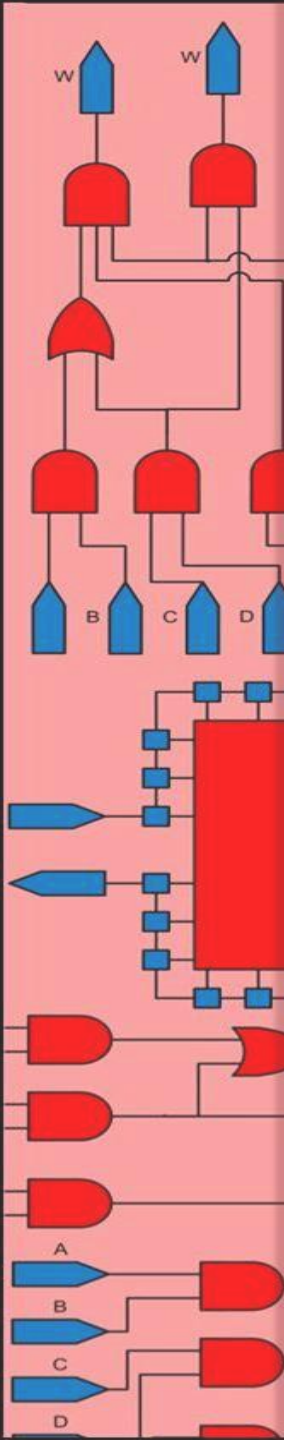
18

```
=== register ===

   Number of wires:                 57
   Number of wire bits:            102
   Number of public wires:           7
   Number of public wire bits:      37
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 81
     $_AOI3_                        16
     $_DFF_PP0_                     16
     $_MUX_                         16
     $_NOT_                         33
```

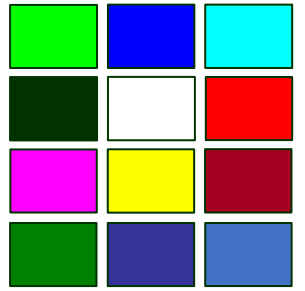
```
=== counter ===

   Number of wires:                 31
   Number of wire bits:             39
   Number of public wires:           8
   Number of public wire bits:      13
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 31
     $_DFF_PP0_                      4
     $_MUX_                          6
     $_NAND_                         2
     $_NOR_                          5
     $_NOT_                          6
     $_OAI3_                         2
     $_OR_                           2
     $_XNOR_                         4
```

```
=== register18 ===

   Number of wires:                 60
   Number of wire bits:            111
   Number of public wires:           7
   Number of public wire bits:      41
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 88
     $_AND_                          2
     $_AOI3_                        18
     $_DFF_PP0_                     18
     $_MUX_                         18
     $_NOR_                          1
     $_NOT_                         33
```





Components of design



Datapath

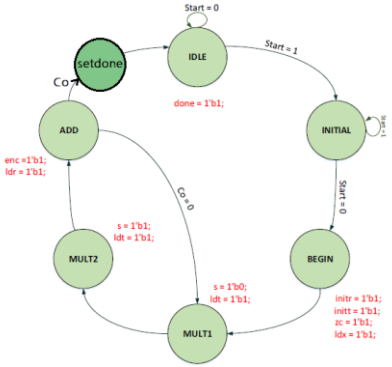
LUT

```
module LUT (input [3:0] adr, output [15:0] data);
reg [15:0] datat;
    always @(adr) begin
    datat = 16'h0000;
        case(adr)
            0: datat = 16'hFFFF;
            1: datat = 16'h8000;//1/2
            2: datat = 16'h5555;//1/3
            3: datat = 16'h4000;// 1/4
            4: datat = 16'h3333;// 1/5
            5: datat = 16'h2AAA;// 1/6
            6: datat = 16'h2492;// 1/7
            7: datat = 16'h2000;// 1/8
        endcase
    end
assign data = datat;
endmodule
```

```
=== LUT ===

   Number of wires:                 28
   Number of wire bits:             61
   Number of public wires:           3
   Number of public wire bits:      36
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 36
     $_AND_                          3
     $_AOI3_                         5
     $_AOI4_                         7
     $_NAND_                         5
     $_NOR_                          6
     $_NOT_                          2
     $_OAI3_                         3
     $_OR_                           5
```

Components of design

Controller




```
module controller(input clk,rst,start,co,
                  output reg done, zx,initx,ldx, zt,initt,ldt, zr,initr,ldr, zc,ldc,enc, s);

    reg [2:0] ps, ns;
    parameter [2:0]
    Idle = 0, Initialization = 1, Begin = 2, Mult1 = 3, Mult2 = 4, Add=5, setdone=6;
    always@(ps,co,start)begin
        ns = Idle;
        case(ps)
            Idle:
                ns = (start)? Initialization : Idle;
            Initialization:
                ns = Begin;
            Begin:
                ns = Mult1;
            Mult1:
                ns = Mult2;
            Mult2:
                ns = Add;
            Add:
                ns = (co)? setdone : Mult1;
            setdone:
                ns = Idle;
        endcase
    end

    always@(ps,co,start)begin
    done = 1'b0; zx = 1'b0; initx = 1'b0; ldx = 1'b0; zt = 1'b0; initt = 1'b0; ldt = 1'b0;
    zr = 1'b0; initr = 1'b0; ldr = 1'b0;  zc = 1'b0; ldc = 1'b0; enc = 1'b0; s = 1'b0;
        case(ps)
            Idle:begin
```

```
=== controller ===

   Number of wires:                 27
   Number of wire bits:             33
   Number of public wires:          19
   Number of public wire bits:      25
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 16
     $_AND_                          2
     $_DFF_PP0_                      6
     $_DFF_PP1_                      1
     $_NOT_                          4
     $_OAI3_                         2
     $_OR_                           1
```

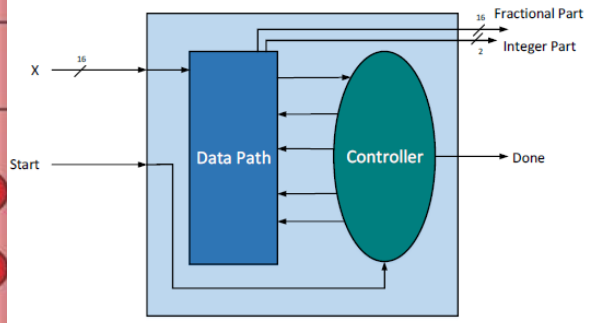
RTL of design

TOP module

```
module exponential(input clk,rst,start, input [15:0] x,
                   output done, output[1:0] intpart, output [15:0]fracpart);

    wire co,zx,initx,ldx,zt,initt,ldt,zr,initr,ldr,zc,ldc,enc,s;

    controller control(clk,rst,start,co,done, zx,initx,ldx,
                       zt,initt,ldt, zr,initr,ldr, zc,ldc,enc, s);

    datapath dP(clk,rst, zx,initx,ldx, zt,initt,ldt,
                zr,initr,ldr, zc,ldc,enc, s,x,co,{intpart,fracpart});
endmodule
```

7

16+16+4+18



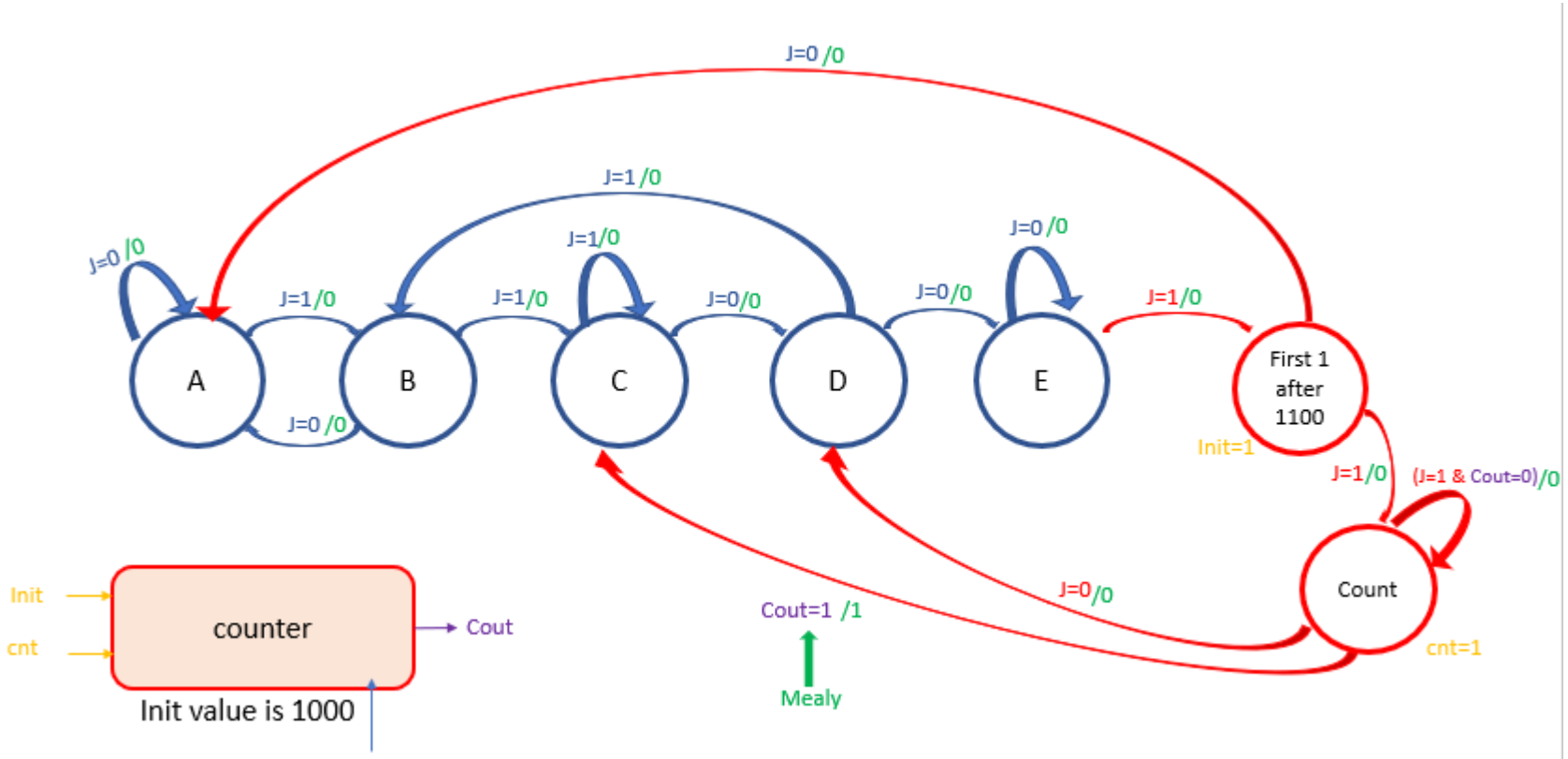

```
=== design hierarchy ===

  exponential                     1
    controller                    1
    datapath                      1
      LUT                         1
      adder                       1
      counter                     1
      multiplier                  1
      mux2to1                     1
      register                    2
      register18                  1

  Number of wires:             1802
  Number of wire bits:         2320
  Number of public wires:       108
  Number of public wire bits:   576
  Number of memories:             0
  Number of memory bits:          0
  Number of processes:            0
  Number of cells:             1868
    $_AND_                      315
    $_AOI3_                     212
    $_AOI4_                       7
    $_DFF_PP0_                   60
    $_DFF_PP1_                    1
    $_MUX_                       72
    $_NAND_                      96
    $_NOR_                      306
    $_NOT_                      183
    $_OAI3_                      41
    $_OR_                        47
    $_XNOR_                     364
    $_XOR_                      164
```

7+18+16+16+4

Solution of HW10- Q2

- Here is your design

7 States



Solution of HW10- Q2

- Here is Verilog description by use of parameter
- Synthesis

Quartus

Flow Status	Successful - Tue Jan 04 10:56:33 2022
Quartus Prime Version	20.1.0 Build 711 06/05/2020 SJ Lite Edition
Revision Name	Sol_2
Top-level Entity Name	Sol_2
Family	Cyclone IV E
Total logic elements	15 / 6,272 (< 1 %)
Total registers	11
Total pins	4 / 92 (4 %)
Total virtual pins	0
Total memory bits	0 / 276,480 (0 %)
Embedded Multiplier 9-bit elements	0 / 30 (0 %)
Total PLLs	0 / 2 (0 %)
Device	EP4CE6E22C6
Timing Models	Final

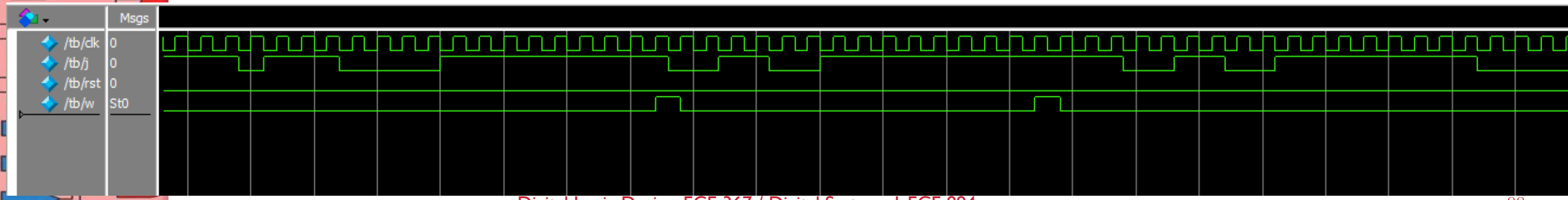
Yosys

```
=== Sol_2 ===

   Number of wires:                 42
   Number of wire bits:             54
   Number of public wires:           8
   Number of public wire bits:      17
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 49
     $_AND_                          2
     $_AOI3_                         1
     $_DFF_PP0_                     10
     $_DFF_PP1_                      1
     $_MUX_                          3
     $_NAND_                         6
     $_NOR_                         11
     $_NOT_                          7
     $_OAI3_                         1
     $_OAI4_                         2
     $_OR_                           1
     $_XNOR_                         4
```

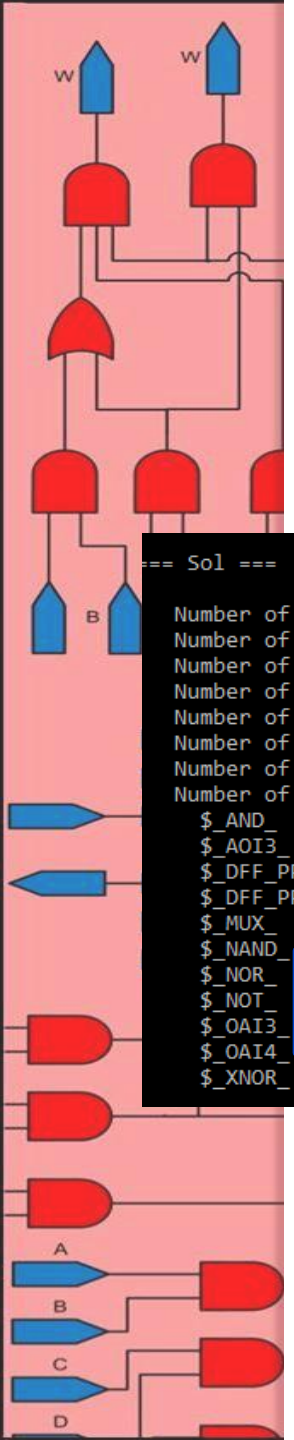
7+4

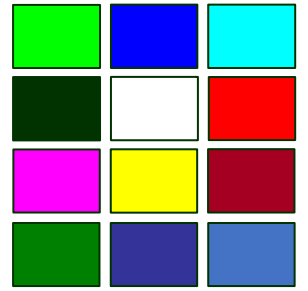
```
module Sol #(parameter S=9)(input clk, rst, j, output  w);
    reg init, cnt;
    reg[2:0] ns, ps;
    reg[$clog2(S)-1:0] count;
    wire Co;
    parameter[2:0] A=3'b000 ,B=3'b001 ,C=3'b010 ,D=3'b011 ,E=3'b100 ,F=3'b101,G=3'b110 ;
    always @(ps, j, Co)begin
        init = 0; cnt = 0;
        ns = 0;
        case(ps)
            A :ns = j ? B:A;
            B :ns = j ? C:A;
            C :ns = j ? C:D;
            D :ns = j ? B:E;
            E :ns = j ? F:E;
            F :begin init = 1 ; ns = j ? G:A; end
            G :begin cnt = 1 ; ns = ( j==1 && Co==0 )? G:(Co==1)?C:D; end
            default: ns = A;
        endcase
    end
    always @(posedge clk, posedge rst) begin
        if (rst)
        ps <= A;
        else
        ps <= ns;
    end
    always @(posedge clk, posedge rst) begin
        if (rst)
        count <= 0;
        else
            if(init)
                count <=(2**($clog2(S)))- S + 1;
            else
                if(cnt)
                    count <= count +1;
    end
    assign Co= (&count);
    assign w= (ps == G)?Co:1'b0;
endmodule
```

- Simulation



Solution of HW10- Q2





S=20 ⟶ 5 bits

```
=== Sol ===

  Number of wires:                 49
  Number of wire bits:             63
  Number of public wires:           8
  Number of public wire bits:      18
  Number of memories:               0
  Number of memory bits:            0
  Number of processes:              0
  Number of cells:                 58
    $_AND_                          5
    $_AOI3_                         1
    $_DFF_PP0_                     11
    $_DFF_PP1_                      1
    $_MUX_                          4
    $_NAND_                         9
    $_NOR_                         11
    $_NOT_                          8
    $_OAI3_                         1
    $_OAI4_                         2
    $_XNOR_                         5
```

7+5

Yosys

Revision Name	Sol
Top-level Entity Name	Sol
Family	Cyclone IV E
Total logic elements	17 / 6,272 (< 1 %)
Total registers	12
Total pins	4 / 92 (4 %)
Total virtual pins	0
Total memory bits	0 / 276,480 (0 %)
Embedded Multiplier 9-bit elements	0 / 30 (0 %)
Total PLLs	0 / 2 (0 %)
Device	EP4CE6E22C6
Timing Models	Final

Quartus

S=177 ⟶ 8 bits

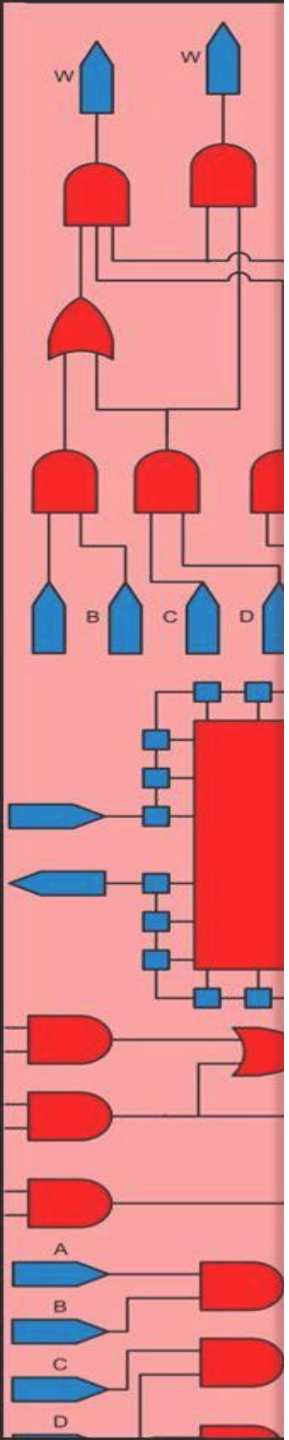
```
=== Sol ===

  Number of wires:                 62
  Number of wire bits:             82
  Number of public wires:           8
  Number of public wire bits:      21
  Number of memories:               0
  Number of memory bits:            0
  Number of processes:              0
  Number of cells:                 77
    $_AND_                          2
    $_AOI3_                         1
    $_DFF_PP0_                     14
    $_DFF_PP1_                      1
    $_MUX_                          7
    $_NAND_                        10
    $_NOR_                         18
    $_NOT_                         11
    $_OAI3_                         2
    $_OAI4_                         1
    $_OR_                           2
    $_XNOR_                         8
```

7+8

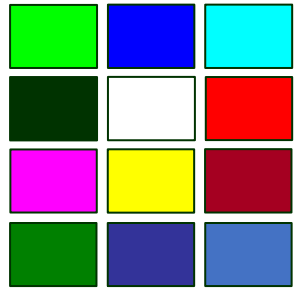
Yosys

Revision Name	Sol
Top-level Entity Name	Sol
Family	Cyclone IV E
Total logic elements	21 / 6,272 (< 1 %)
Total registers	15
Total pins	4 / 92 (4 %)
Total virtual pins	0
Total memory bits	0 / 276,480 (0 %)
Embedded Multiplier 9-bit elements	0 / 30 (0 %)
Total PLLs	0 / 2 (0 %)
Device	EP4CE6E22C6
Timing Models	Final

Quartus



Acknowledgement



These Slides:

- Prepared by Zahra Jahanpeima & Zahra Mahdavi
- Revised by Mozhgan Rezaie Manavand