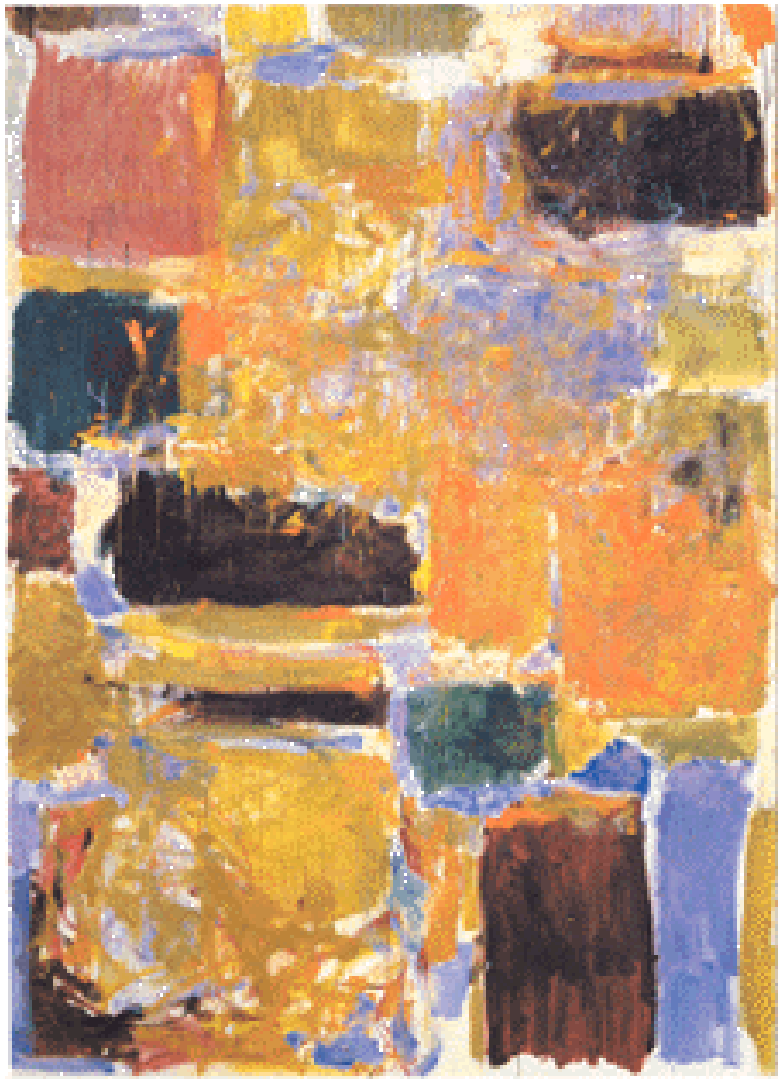


CS150 Discussion 3

Simon Scott

Coding Guidelines

1. When modeling sequential logic, use nonblocking assignments.

2. When modeling latches, use nonblocking assignments.

3. When modeling combinational logic with an always block, use blocking assignments.

4. When modeling both sequential and combinational logic within the same always block, use nonblocking assignments.

5. Do not mix blocking and nonblocking assignments in the same always block.

6. Do not make assignments to the same variable from more than one always block.

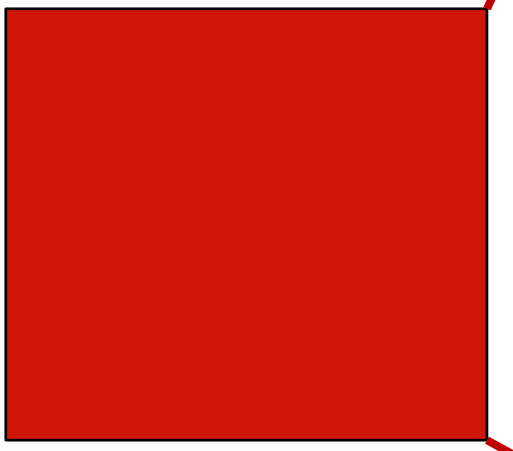
Blocking vs Non-blocking

```
module m_nb (
   input clk,
   input in,
   output [7:0] cout
);
   reg [7:0] accum;
   reg [7:0] accum_d;

   always@(posedge clk) begin
      accum <= accum + in;
      accum_d <= accum;
      cout <= accum_d;
   end
endmodule
```

```
module m_b (
   input clk,
   input in,
   output [7:0] cout
);
   reg [7:0] accum;
   reg [7:0] accum_d;

   always@(posedge clk) begin
      accum = accum + in;
      accum_d = accum;
      cout = accum_d;
   end
endmodule
```

Examples of using (non)-blocking assign

```
module statemachine (
   input clk,
   input rst,
   input in_a,
   output out_a
);
   reg [1:0] curr_state;
   reg [1:0] next_state;
```



```
endmodule
```

```
always @(posedge clk)
        if (rst) curr_state = STATE_0;
        else curr_state = next_state;

always @(*)
   case (curr_state)

    STATE_0: if (in_a) begin
                    out_a = 1'b1;
                    next_state = STATE_1;
             end
             else begin
                    out_a = 1'b0;
                    next_state = STATE_0;
              end

    STATE_1: if (in_a) begin
                    ……
                    end
     default: …

   endcase
end
```

Boolean Algebra

Simplify: ab + aa’ + bc + ab’ + c

DeMorgan’s Law:

$(x + y + z + \ldots)' = x'y'z'$ $(x\ y\ z\ \ldots)' = x' + y' + z'$

Simplify: $\overline{(\overline{a} + \overline{b})*(a + \overline{b})*c}$

Simplify: $Y = \overline{AB}*\overline{C}*\overline{D} + A*\overline{B}*\overline{C} + A*\overline{B}*C*\overline{D} + ABD + \overline{AB}*C*\overline{D}$

Truth Tables

Circuit has a 4-bit input $\{A_3, A_2, A_1, A_0\}$, representing a binary number.
Output P = 1 if A is prime or 15; otherwise P = 0.
Note: 0 and 1 are not prime,

a.) Draw the truth table

b.) Write SoP and simplify using multi-level logic
Hint: factorize!

c.) Write the first 4 terms of the PoS expression

d.) Use K-maps to find a simplified expression for P

Karnaugh Maps

Simplify using K-map:

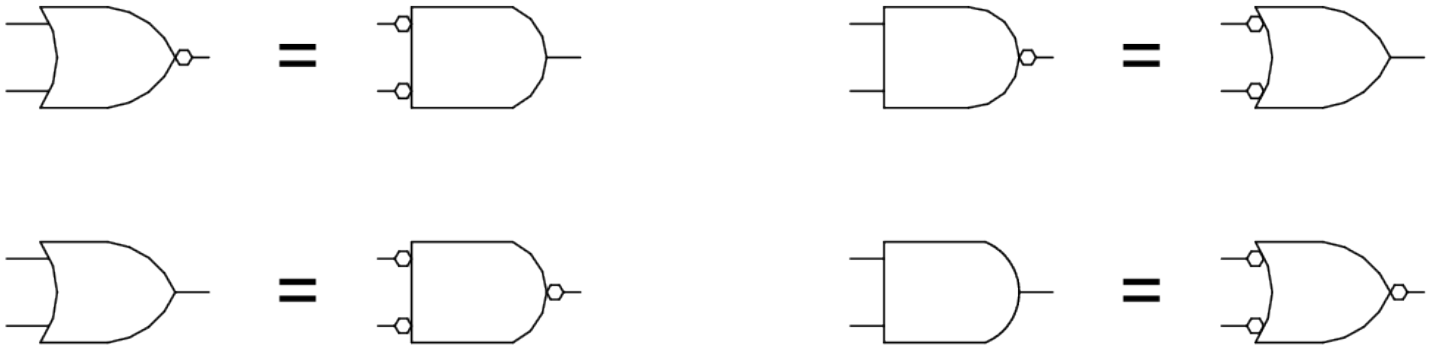
A	B	C	D	Y
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

Using NAND Gates

DeMorgan's Law Review:

$(a + b)' = a' b'$ $\qquad$ $(a\ b)' = a' + b'$

$a + b = (a' b')'$ $\qquad$ $(a\ b) = (a' + b')'$



Convert to NANDs:

$f = a(b + cd) + bc'$