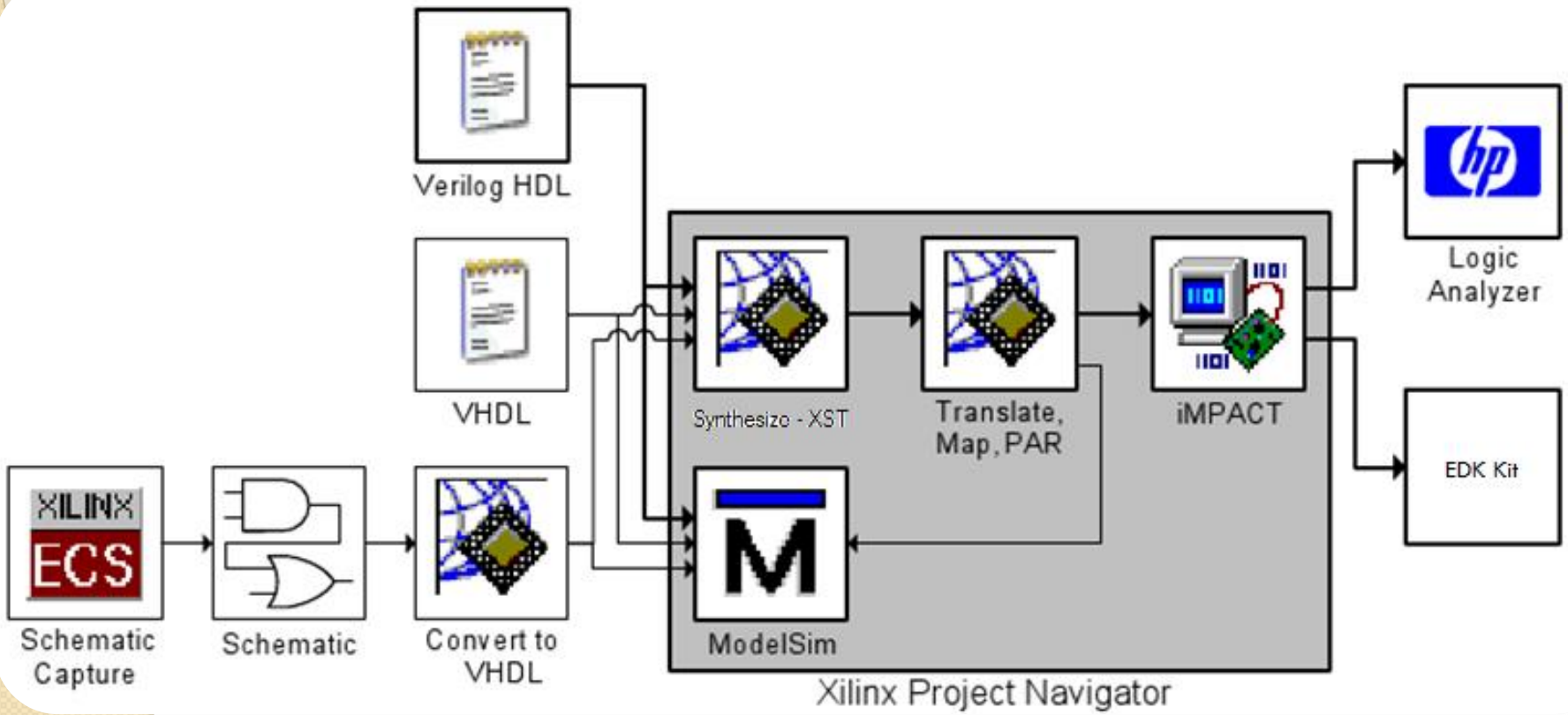
Synthesizers and VHDL Interpreting

By:

Mustafa M. Shiple

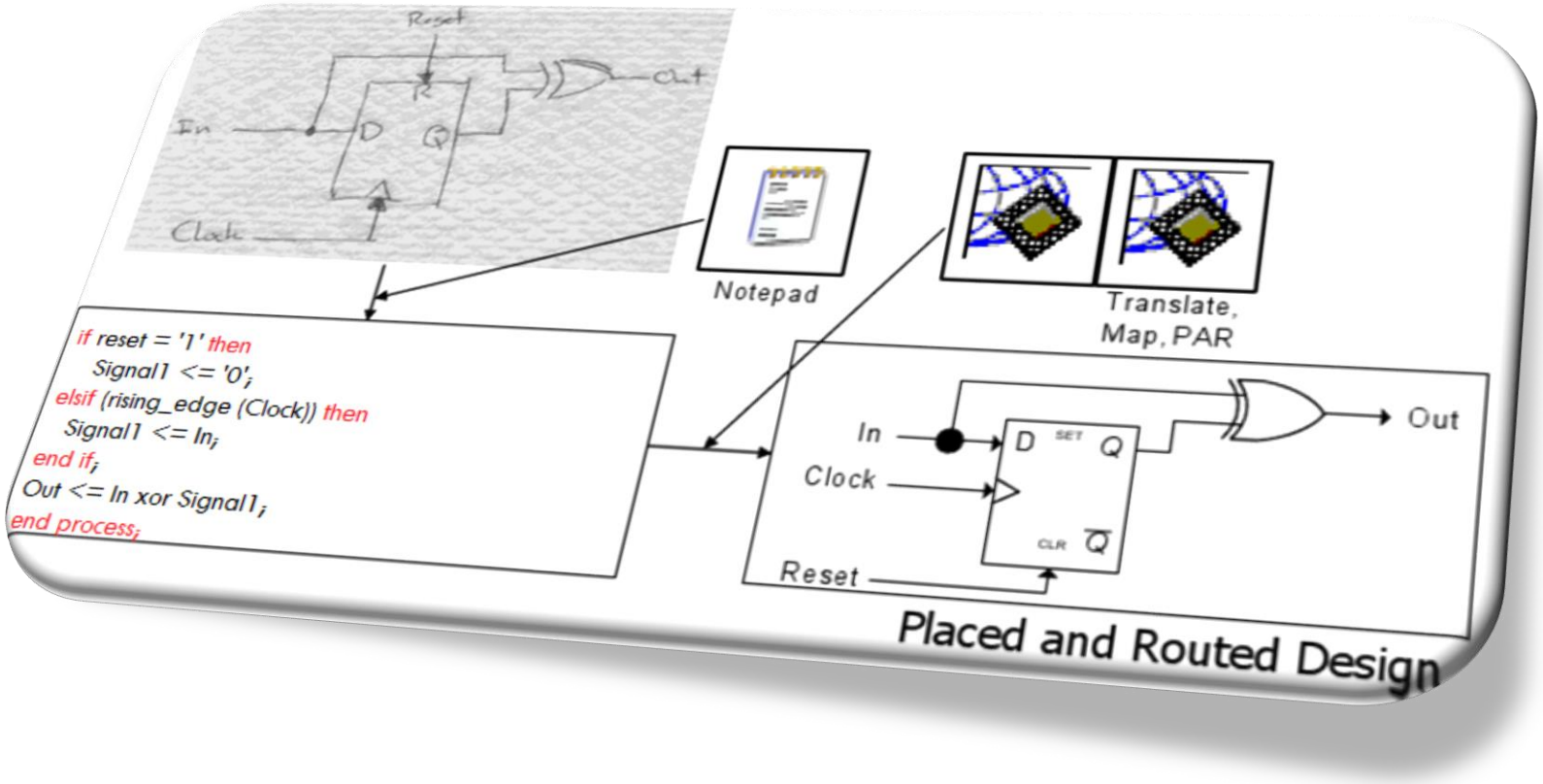
http://Drshiple-courses.weebly.com/

Xilinx Flow

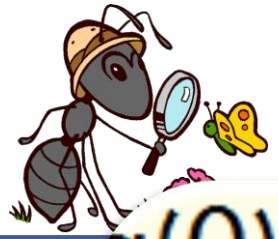


http://Drshiple-courses.weebly.com/

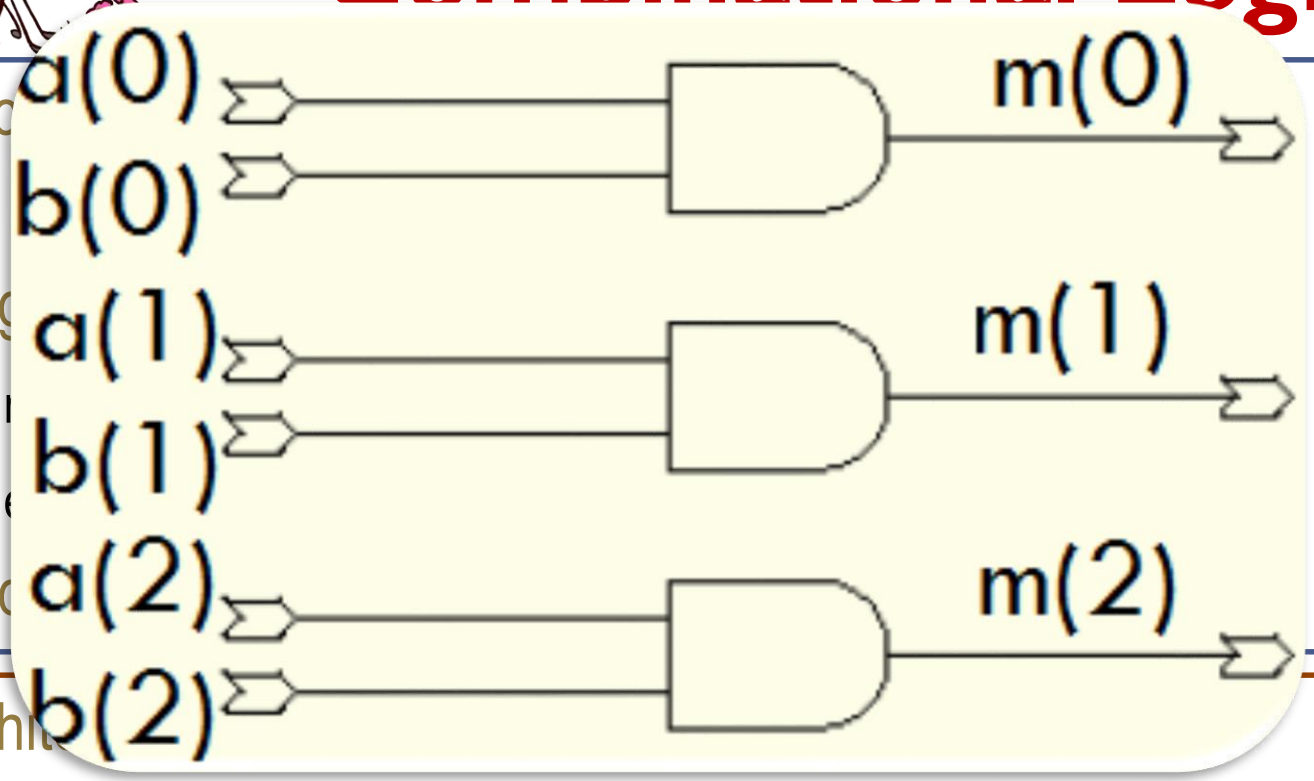
ISE Synthesizer



http://Drshiple-courses.weebly.com/



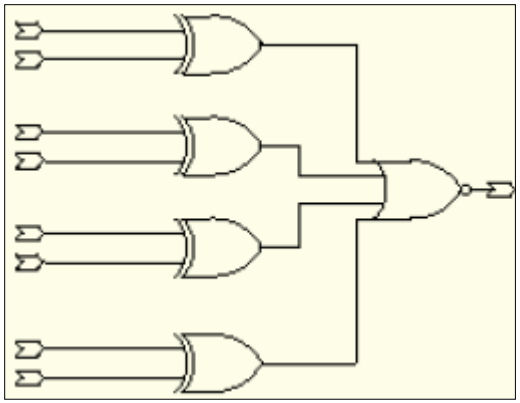
Programming Combinational Logic




```
Arc
beg
end
archi
begin
m <= a and b;     -- a, b: in std_logic_vector (0 to 2);
end example;
```

Relational Operators

Equality : M_out<= Ain = Bin;

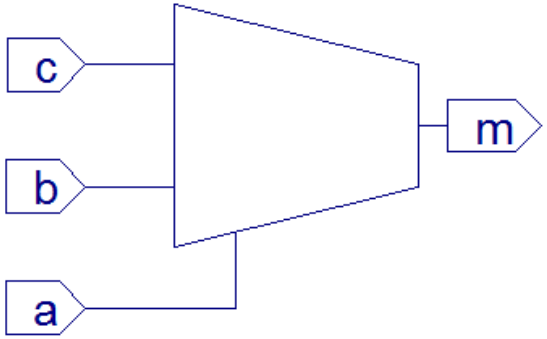


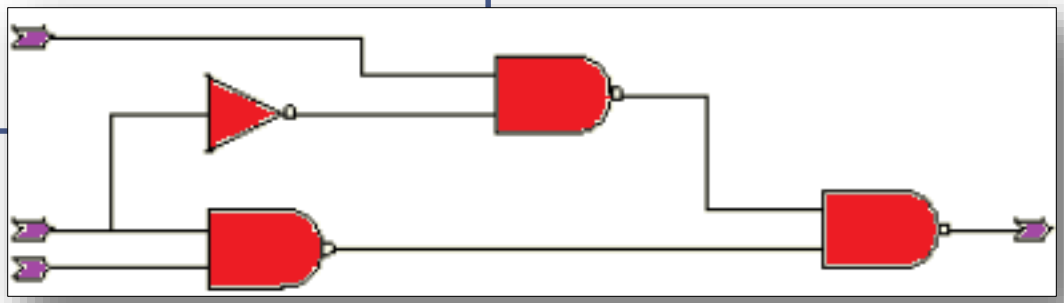
Greater than or equal to : M_out <= Ain>= Bin;

Ain	Bin	M_out
0	0	1
0	1	0
1	0	1
1	1	1

If Statment

```
process (a, b, c)
     variable n: std_logic;
Begin
    if a = '1' then
            n := b;
    else
            n := c;
    end if;
    m <= n;
end process;
```

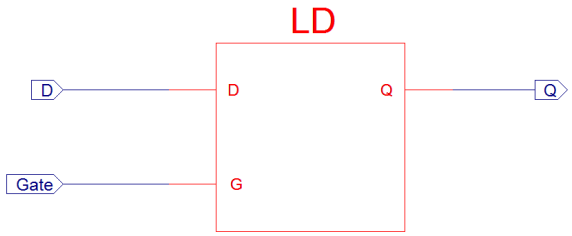


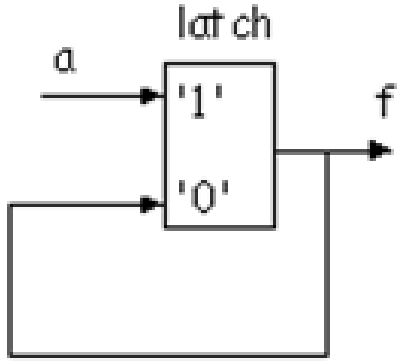


http://Drshiple-courses.weebly.com/

Incomplete *assignment*

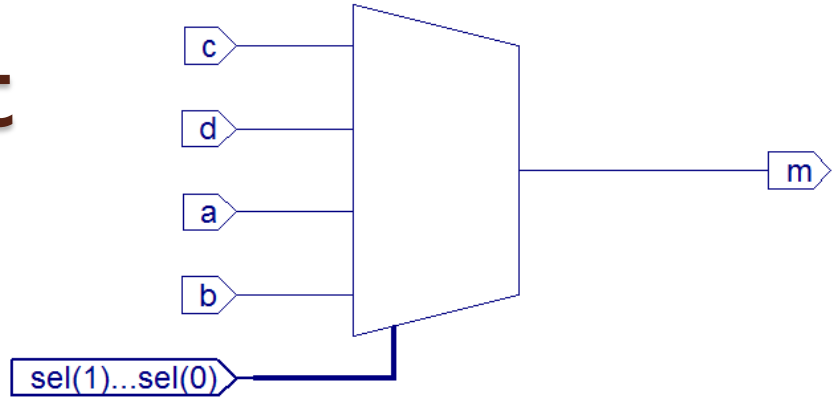
```
 process (a, b, c)
      variable n: std_logic;
begin
      if a = '1' then
            n := b;
      end if;
      m <= n;
end process;
```






```
=========================================================================
*                    HDL Synthesis                                      *
=========================================================================

Synthesizing Unit <testgats>.
    Related source file is D:/PHD/FINAL-VHDL/RIJNDEAL/CSbox_simulation/testgats.vhd
WARNING:Xst:647 - Input <c> is never used.
WARNING:Xst:736 - Found 1-bit latch for signal <$n0000> created at line 23.
```

Case statement

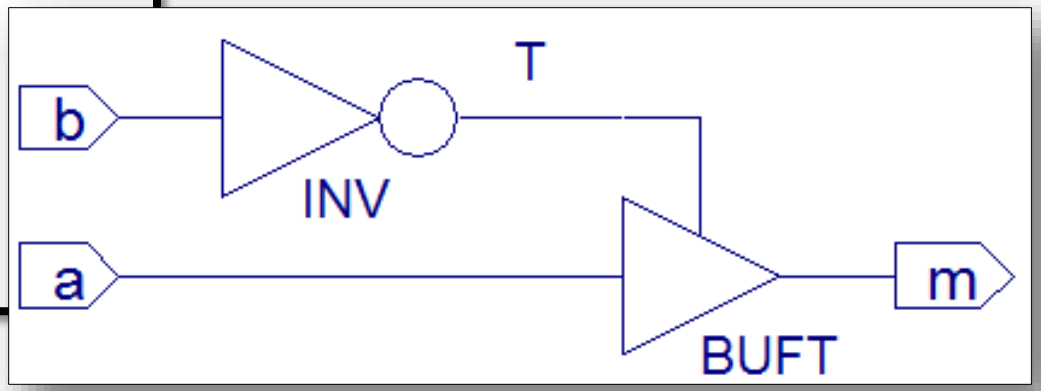



```
process (sel,a,b,c,d)
begin
      case sel is
            when b"00" => m <= c;
            when b"01" => m <= d;
            when b"10" => m <= a;
            when others => m <= b;
      end case ;
end process;
```

ERROR:HDLParsers:812 - D:/testgats.vhd Line 27. A value is missing in case.

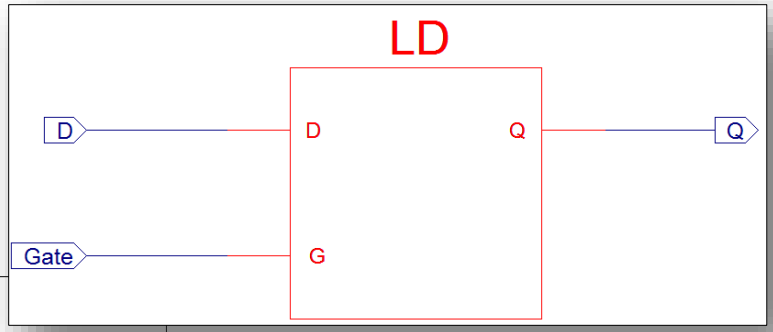
Tri-states

```
process (b, a)
 begin
   if b ='1' then
     m <= a;
   else
     m <= 'Z';
   end if;
end process;
```



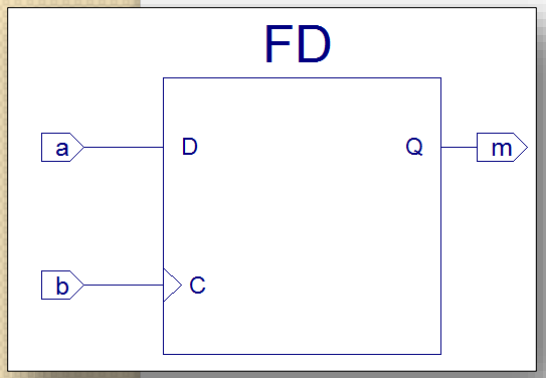
Found 1-bit tristate buffer for signal <m>.
Summary:
inferred 1 Tristate(s).

Latches or FF




```
 process (a, b, c)
      variable n: std_logic;
begin
      if a = '1' then
            n := b;
      end if;
      m <= n;
end process;
```

```
process (b, a)
 begin
  if rising_edge (b ) then
    n <= a;
  end if;
  m<=n;
end process;
```



http://Drshiple-courses.weebly.com/

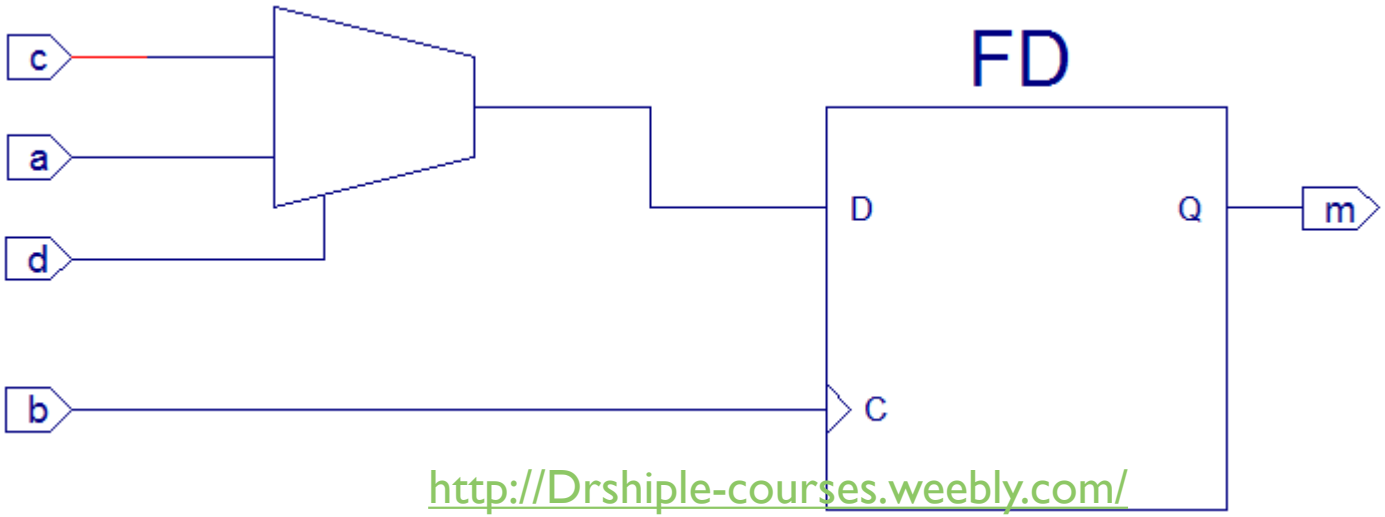
Exercise

```
if rising_edge (b ) then
  if d = '1' then
   m<= a;
   else
   m<= c;
   end if;
end if;
```

?

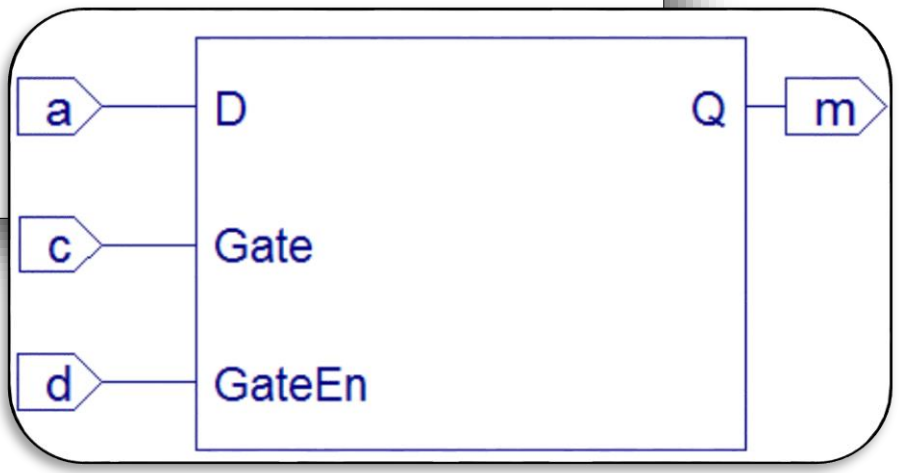
Solution

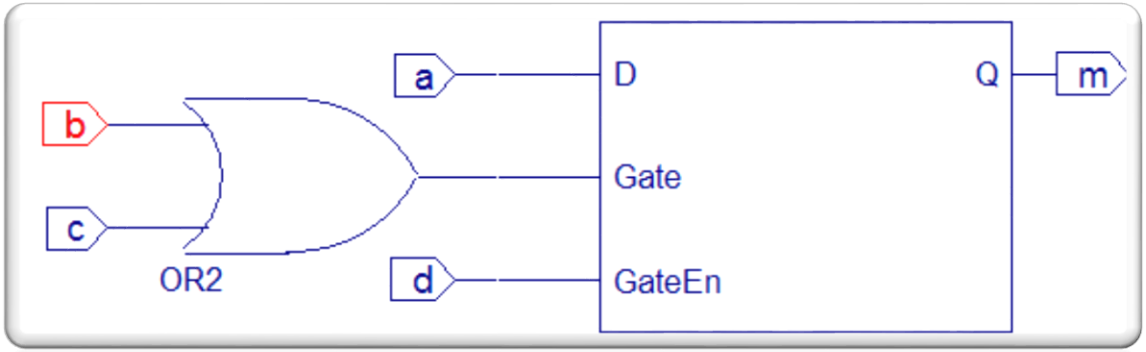
```
if rising_edge (b ) then
 if d = '1' then
  m<= a;
  else
  m<= c;
  end if;
end if;
```



Gate Enable

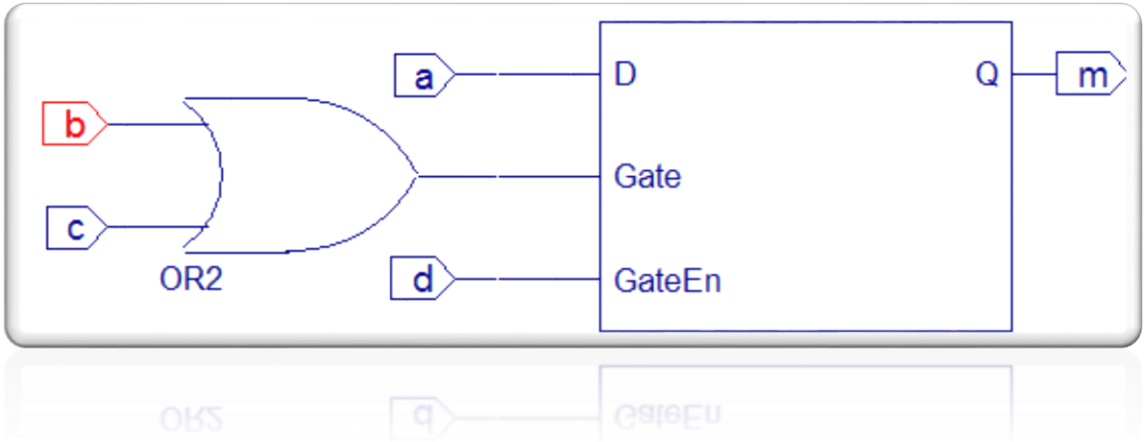
```
if d = '1' and c ='1' then
 m<= a;

 end if;
```





?

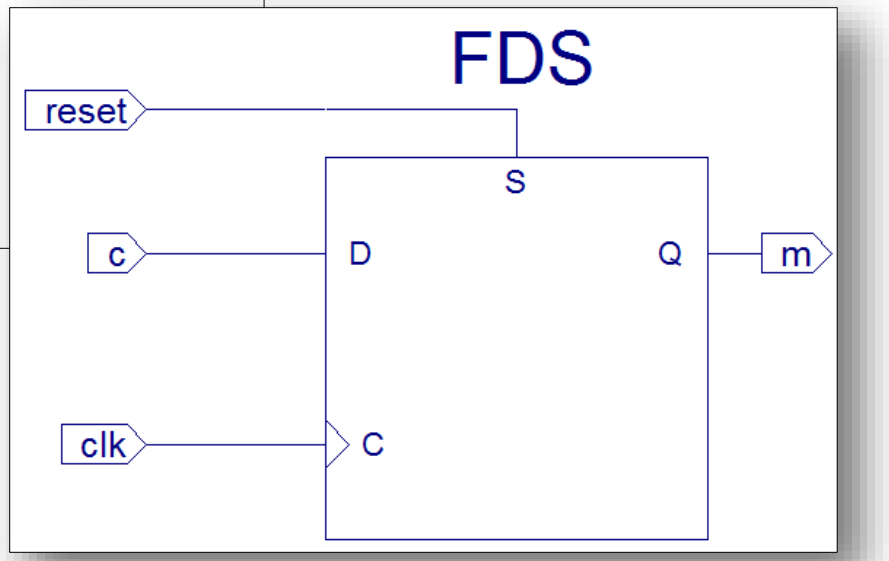
Solution




```
if d = '1' and (c ='1' or b = '1') then
 m<= a;

 end if;
```

Synchronous Set or Reset

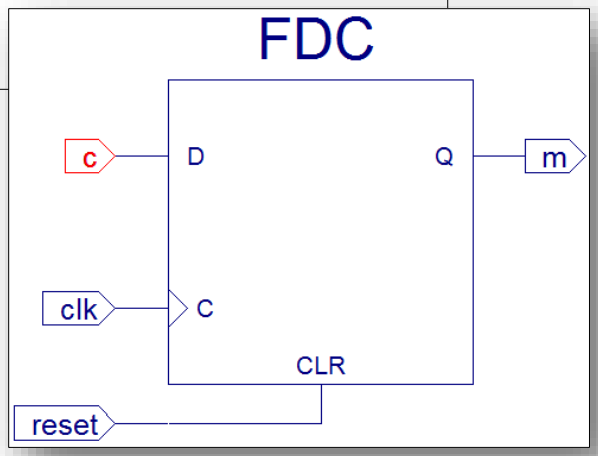
```
if rising_edge (clk) then
    if reset ='1' then
        m<= '1';
    else
        m<= c;
    end if;
end if;
```



Test when m<= ‘0’

Asynchronous Set or Reset

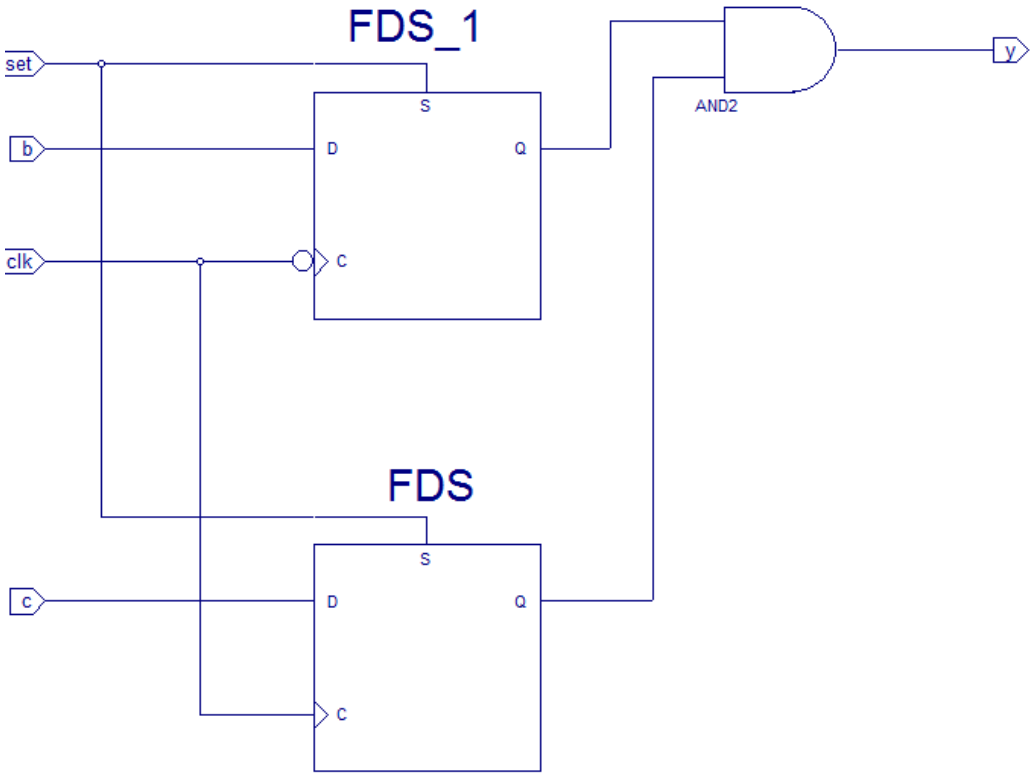
```
if reset ='1' then
 m<= '0';
elsif rising_edge (clk) then
    m<= c;
end if;
```



Test when m<= ‘1’

Double process

```
process (b, a,c)
 begin
  if rising_edge (clk) then
       if reset ='1' then
            m<= '1';
       else
            m<= c;
       end if;
  end if;
end process;
    y <=  m and n;
process (b, a,c)
 begin
  if falling_edge (clk) then
       if reset ='1' then
            n<= '1';
       else
            n<= b;
       end if;
  end if;
end process;
```



http://Drshiple-courses.weebly.com/

Errors

```
if rising_edge (b ) then
   m<= a;
   else
   m<= c;
end if;
```

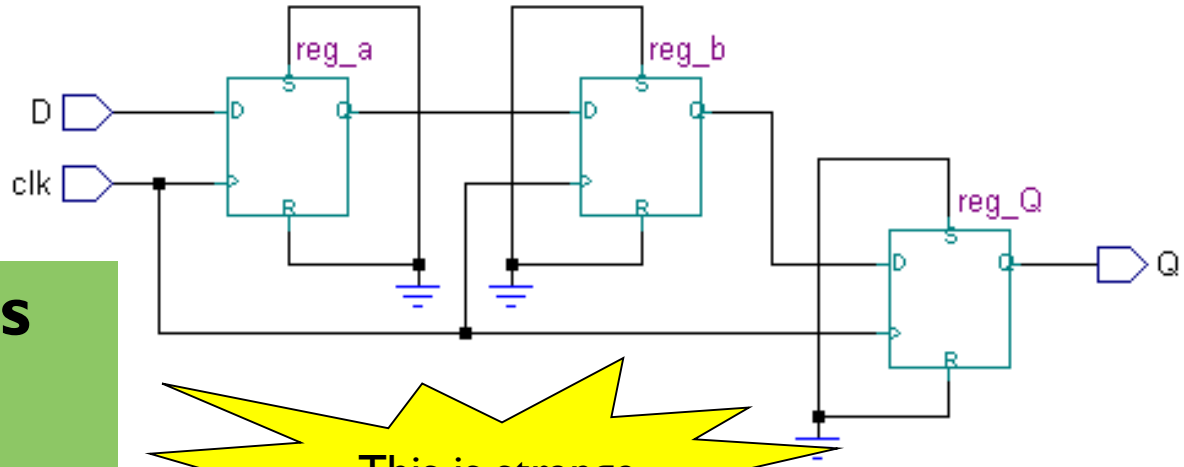
Signal m cannot be synthesized, bad synchronous description.

Flip Flops inferred by Variables

```
architecture EG of FLIPFLOP is
begin
  process (clk)
  variable a,b : std_logic;
  begin
    if (clk'event and clk='1') then
          Q <= b;
          b := a;
          a := D;
    end if;
  end process;
end EG;
```

This assignment of signal needs to have a FF with output a

Here variable b has no value, it is created only after a has value and a only after D has value



- **We obtain a 3 bits shift register**
- **Explain why it is so**

This is strange

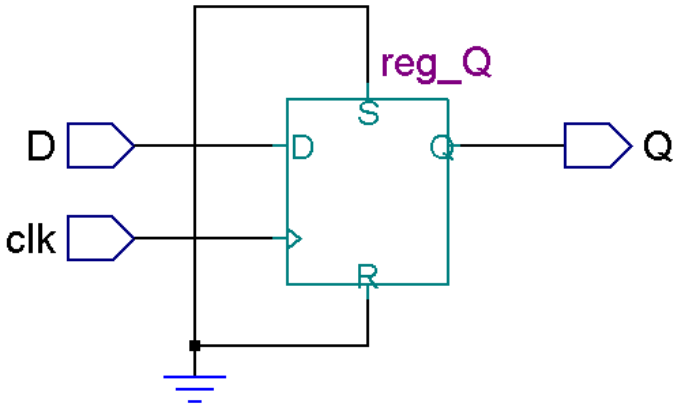
For every variable we have a ff.

Flip Flops inferred by Variables

```
architecture EG of FLIPFLOP is
begin
  process (clk)
  variable a,b : std_logic;
  begin
    if (clk'event and clk='1') then
        a := D;
        b := a;
        Q <= b;
    end if;
  end process;
end EG;
```

Remember that variables are assigned immediately

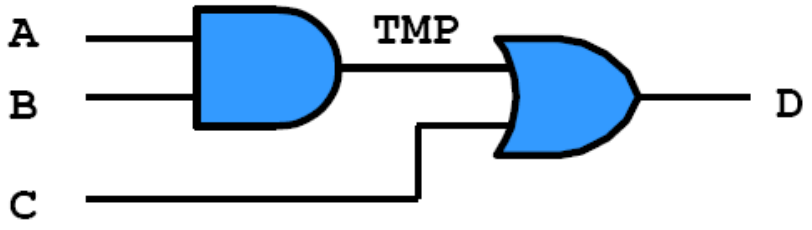
The only difference here is order of assignments in process, here we have variable assignments first.



- Variable assignment order – FF created when assigned before use.

Order of variables again!!!!

```
signal A,B,C,D: bit
...
NO_MEMORY: process (A,B,C)
variable TMP: bit;
begin
        TMP:= A and B;
        D <= TMP or C;
end process;
```



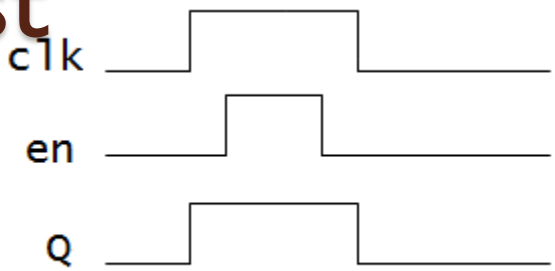
Implementation of TMP : wire

```
signal A,B,C: bit
...
IS_IT_LATCH: process (A,B,C)
variable TMP: bit;
begin
        C <= TMP and B;
        TMP := A or C;
end process;
```

Implementation of TMP : latch. Do you real want it?

Incomplete Sensitivity list

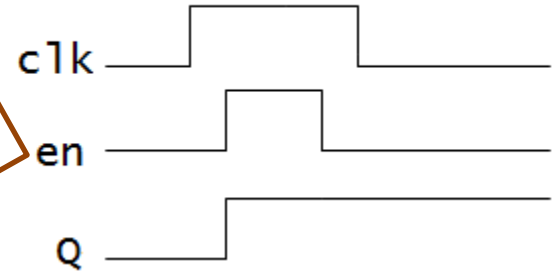
complete




```
process (en)
--variable   tmp: std_logic;
  begin
    Q<= clk and en;
  end process;
```

Synthesize incomplete

Simulation incomplete



One or more signals are missing in the process sensitivity list. To enable synthesis of FPGA/CPLD hardware, XST will assume that all necessary signals are present in the sensitivity list. Please note that the result of the synthesis may differ from the initial design specification. The missing signals are:

<clk>