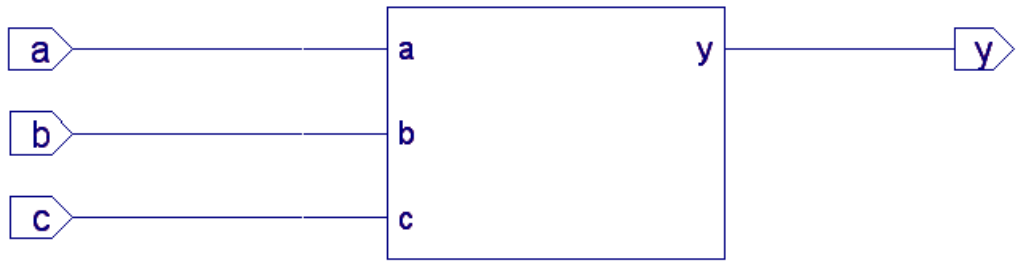
Design Entry

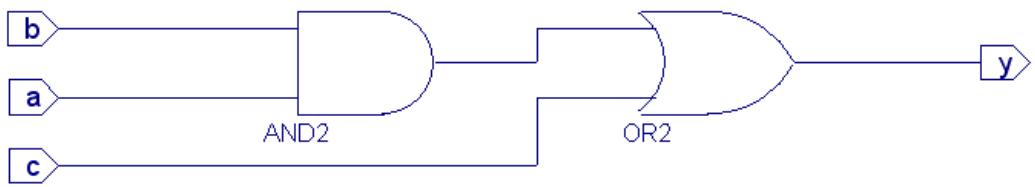
- Hierarchical schematic
- Finite state machine
- Hardware description language (HDL)
  - PALASM
  - Abel
  - AHDL
  - Verilog (IEEE 1364-2001)
  - VHDL (IEEE 1076-2001)
  - SystemC

VHDL - Behavioral Description

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;


entity andor is
port(
  A : in STD_LOGIC;
  B : in STD_LOGIC;
  C : in STD_LOGIC;
  Y : out STD_LOGIC
);
end andor;
```

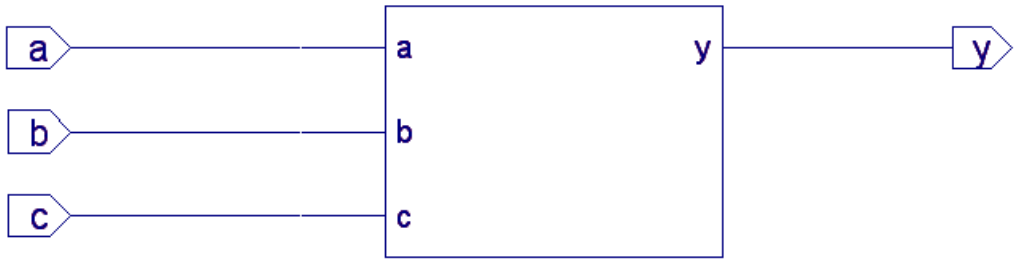


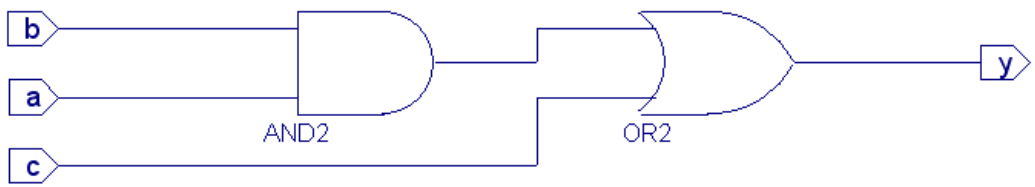

```
architecture behavioral of andor is
begin
  Y <= (A and B) or C;
end behavioral;
```



VHDL - Structural Description

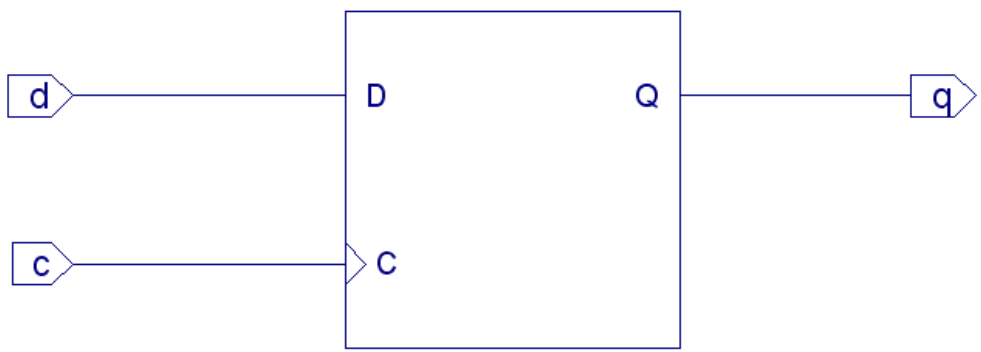
```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity andor is
port(
  A : in STD_LOGIC;
  B : in STD_LOGIC;
  C : in STD_LOGIC;
  Y : out STD_LOGIC
);
end andor;

architecture structural of andor is
signal int : STD_LOGIC;

component AND2
port ( i0 : in STD_LOGIC;
       i1 : in STD_LOGIC;
       O : out STD_LOGIC);
end component;

component OR2
port ( i0 : in STD_LOGIC;
       i1 : in STD_LOGIC;
       O : out STD_LOGIC);
end component;

begin
U1 : AND2
  PORT MAP (I0=>b, I1=>a, O=>int);
U2 : OR2
  PORT MAP (I0=>c, I1=>int, O=>y);
end structural;
```





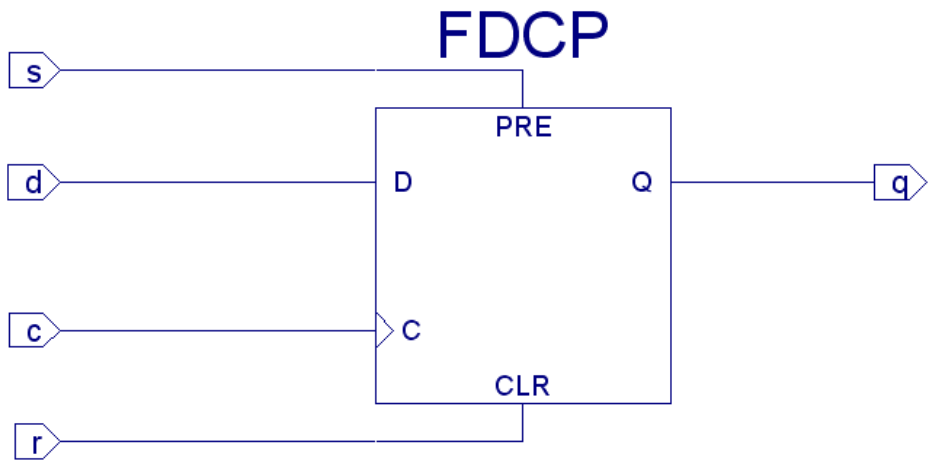
Flip-Flop

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity flipflop is
    Port ( D : in std_logic;
           Q : out std_logic;
           C : in std_logic);
end flipflop;

architecture behavioral of flipflop is
begin
  process (C)
    begin
     if C'event and C='1' then  -- C rising edge
        Q <= D;
     end if;
  end process;
end behavioral;
```



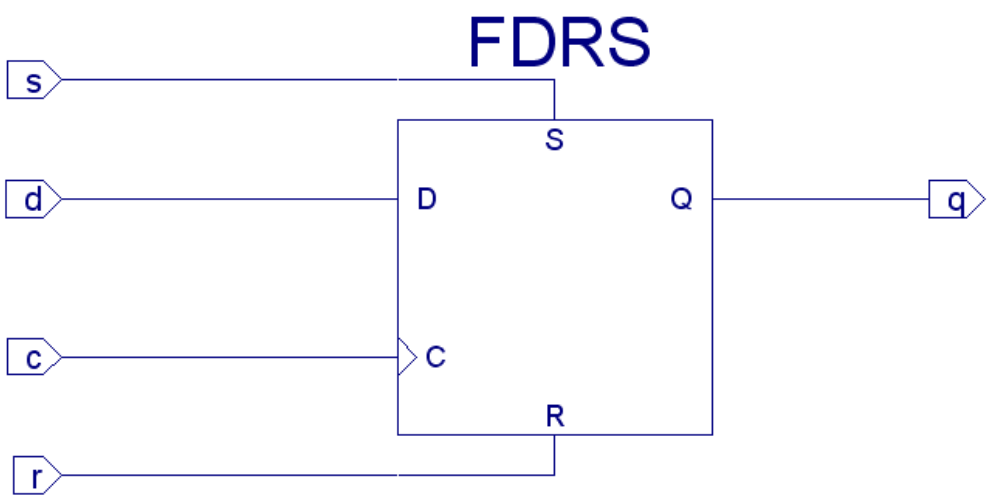
Flip-Flop with Preset and Clear

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity flipflop is
    Port ( D : in std_logic;
           R : in std_logic;
           S : in std_logic;
           Q : out std_logic;
           C : in std_logic);
end flipflop;

architecture behavioral of flipflop is
begin
 process (C,R,S)
  begin
   if R = '1' then
      Q <= '0';
   elsif S = '1' then
      Q <= '1';
   elsif C'event and C='1' then  -- C rising edge
      Q <= D;
   end if;
  end process;
end behavioral;
```



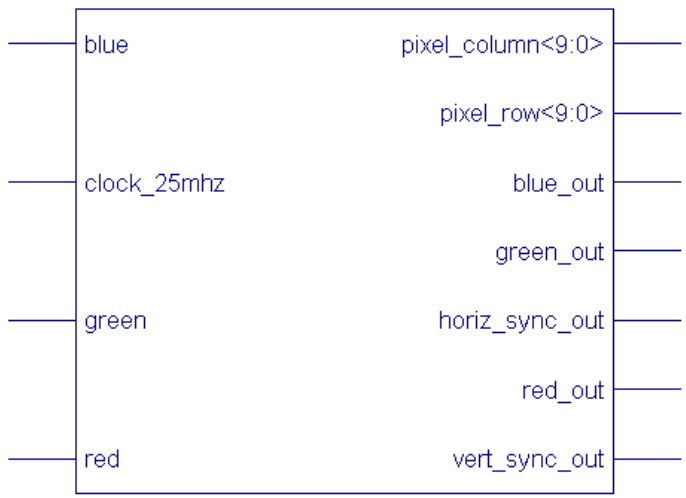
Flip-Flop with Set and Reset

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity flipflop is
    Port ( D : in std_logic;
           R : in std_logic;
           S : in std_logic;
           Q : out std_logic;
           C : in std_logic);
end flipflop;

architecture behavioral of flipflop is
begin
 process (C)
  begin
   if C'event and C='1' then  -- C rising edge
    if  R='1' then
      Q <= '0';
    elsif S='1' then
      Q <= '1';
    else
      Q <= D;
    end if;
   end if;
 end process;
end behavioral;
```



Video Signal Generator

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.all;
USE IEEE.STD_LOGIC_ARITH.all;
USE IEEE.STD_LOGIC_UNSIGNED.all;

-- VGA Video Sync generation

ENTITY VGA_SYNC IS

PORT(clock_25Mhz, red, green, blue: IN  STD_LOGIC;
     red_out, green_out, blue_out,
     horiz_sync_out, vert_sync_out: OUT STD_LOGIC;
     pixel_row, pixel_column : OUT STD_LOGIC_VECTOR(9 DOWNTO 0));

END VGA_SYNC;
```



Video Signal Generator

```
ARCHITECTURE a OF VGA_SYNC IS

 SIGNAL horiz_sync, vert_sync : STD_LOGIC;
 SIGNAL video_on, video_on_v, video_on_h : STD_LOGIC;
 SIGNAL h_count, v_count : STD_LOGIC_VECTOR(9 DOWNTO 0);

BEGIN

-- video_on is high only when RGB data is being displayed

 video_on <= video_on_H AND video_on_V;
 horiz_sync_out     <= horiz_sync;
 vert_sync_out      <= vert_sync;
 pixel_column <= h_count;
 pixel_row <= v_count;
```

Video Signal Generator

```
-- Generate Horizontal and Vertical Timing Signals for Video Signal

 PROCESS

 BEGIN

  WAIT UNTIL (clock_25Mhz'EVENT) AND (clock_25Mhz='1');

   -- H_count counts pixels (640 + extra time for sync signals)
   --
   --  Horiz_sync  ------------------------------------_____________--------
   --  H_count       0                 640             659       755    799
   --

   IF (h_count = 799) THEN
      h_count <= "0000000000";
   ELSE
      h_count <= h_count + 1;
   END IF;

   -- Generate Horizontal Sync Signal using H_count
   if(h_count = 756) then
      horiz_sync <= '1';
   end if;
   if(h_count = 659) then
      horiz_sync <= '0';
   end if;
```

Video Signal Generator

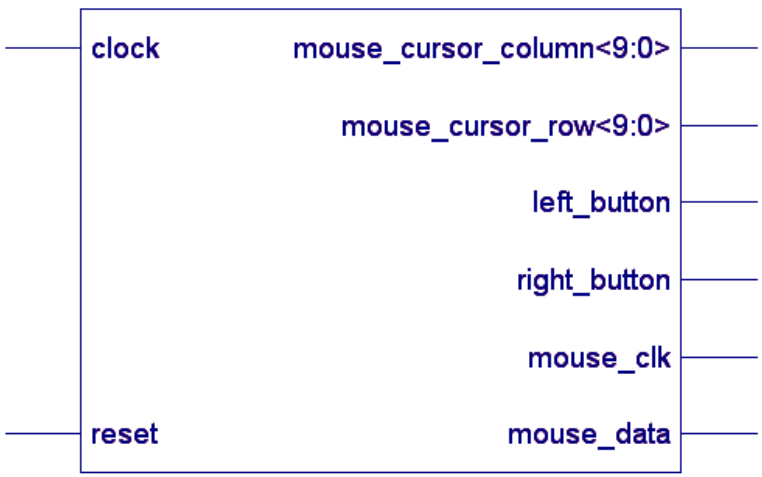
```
--V_count counts rows of pixels (480 + extra time for sync signals)
--
--  Vert_sync      ----------------------------------_______------------
--  V_count         0                             480    493-494          524
--
IF (v_count = 524) AND (h_count = 799) THEN
  v_count <= "0000000000";
ELSIF (h_count = 799) THEN
  v_count <= v_count + 1;
END IF;
-- Generate Vertical Sync Signal using V_count
if (v_count = 493) then
     vert_sync <= '0';
end if;
if (v_count = 495) then
     vert_sync <= '1';
end if;
-- Generate Video on Screen Signals for Pixel Data
if(h_count = 639) then
  video_on_h <= '0';
end if;
if (h_count = 799) then
  video_on_h <= '1';
  if(v_count = 479) then
    video_on_v <= '0';
  end if;
  if(v_count = 524) then
    video_on_v <= '1';
  end if;
end if;
```

Video Signal Generator

```
-- Put all video signals through DFFs to eliminate
-- any logic delays that can cause a blurry image

 red_out  <= red AND video_on;
 green_out <= green AND video_on;
 blue_out <= blue AND video_on;

 END PROCESS;

END a;
```

Mouse Driver

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity mousedriver is
    Generic( timeout: std_logic_vector(19 downto 0) := CONV_STD_LOGIC_VECTOR(100000,20));
    Port ( clock : in std_logic;
           reset : in std_logic;
           mouse_data : inout std_logic;
           mouse_clk : inout std_logic;
           left_button : out std_logic;
           right_button : out std_logic;
           mouse_cursor_row : out std_logic_vector(9 downto 0);
           mouse_cursor_column : out std_logic_vector(9 downto 0));
end mousedriver;
```



Mouse Driver