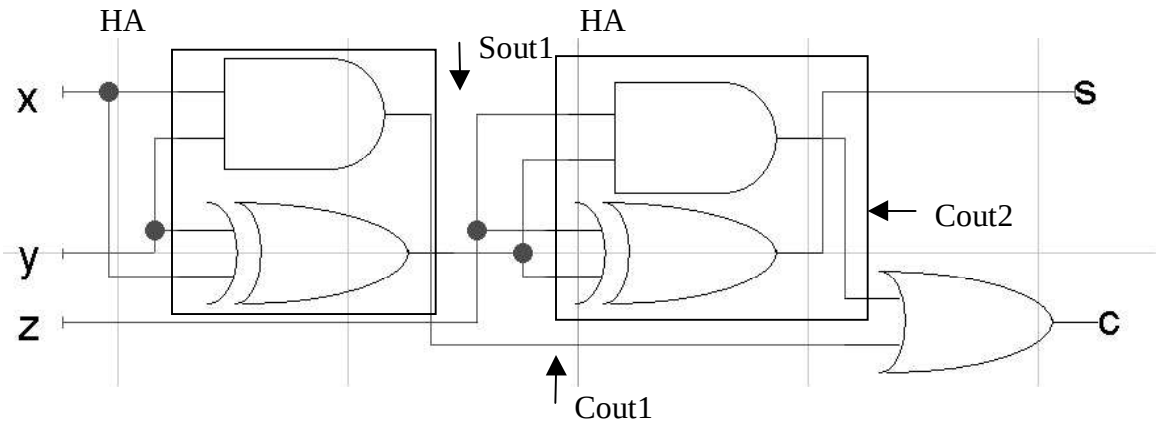
May 14
Lecture #5

Full adder

- structureal description




```
architecture struct of full_adder is
        signal Sout1, Cout1, Cout2: std_logic:
    begin
    HA1: entity work.half_adder
            port map(
                    x => x,
                    y => y,
                    s => Sout1,
                    c => Cout1);
    HA2: entity work.half_adder
            port map(
                    x => Sout1
                    y => Z
                    s => S
                    c Cout2);
    OR1: entity work.or_gate
            portmap (
                    a => Cout2.
                    b => Cout1,
                    c => C);
            end struct;
```

- this circuit needs some connection to hook the entity together
  - signals Cout1, Cout2, Sout1 are defined as std_logic
  - signals and parts are treatred similarly
- One of the reasons to describe circuits in VHDL is simulation
  - we need to connect the ports of a circuit, so we can see it work
  - this is done w/ a “test bench”

- the test bench is self-contained (no ports)
- we simulate the test bench

- our test bench will be called “tb” and have no ports:

```
entity tb is
end tb:
architecture behave of tb is
        signal X, Y, Z, S, C: std_logic;
begin
        TEST: process
        begin
                x <= ‘0’;
                y <= ‘0’;
                z <= ‘0’;
        wait for 100 ns;
        x <= ‘1’l
        wait fr 50 ns;
        y<= ‘1’l
        wait for 50ns;
        z<= ‘1’;
        wait for 50ns;
end process
UUT: entity work.full_adder (struct)
        port map (x=>x y=>y, z=>z, s=>s, c=>c);
end behave;
```

- process always run as often as possible
  - w/ a dependency list
    - process (a, b, c)
  - is only allowed to start when the signals change
- other wise, it starts again immediately
- so, ever process should have either r a dependency list or waits
- when instantiating, we can specify the implementation:

```
        entity work.thing (behave)
```

wait:

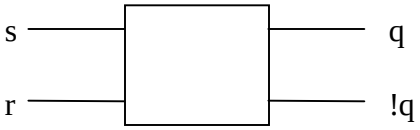
- makes the process pause
- three versions
  1. wait for <time>;
     pause for the given time (simulation)
  2. wait until <condition>;
     pause until the condition is true
     ie. wait until c=’1’;
  3. wait on <signal list>
     wait until one of the signals change
     eg.

```
process (a, b)                  process wait on a, b;
  …..                                 …..
```

end process; end process;

Laches in VHDL

- SR-Latch




```
entity sr_latch is
        port (
                sr: in std_logic;
        q, q_b: out std_logic);
        end sr_latch;
        architectural behave of sr_latch is
        begin
                sr_process: process (s,r)
                begin
                        if      s= “1’ then
                                q <= “1”;
                                q_b <= “0”;
                        elseif  r= “1” then
                                q<= “0”
                                q_b <= “1”;
                        end if
                end process:
        end behav
```