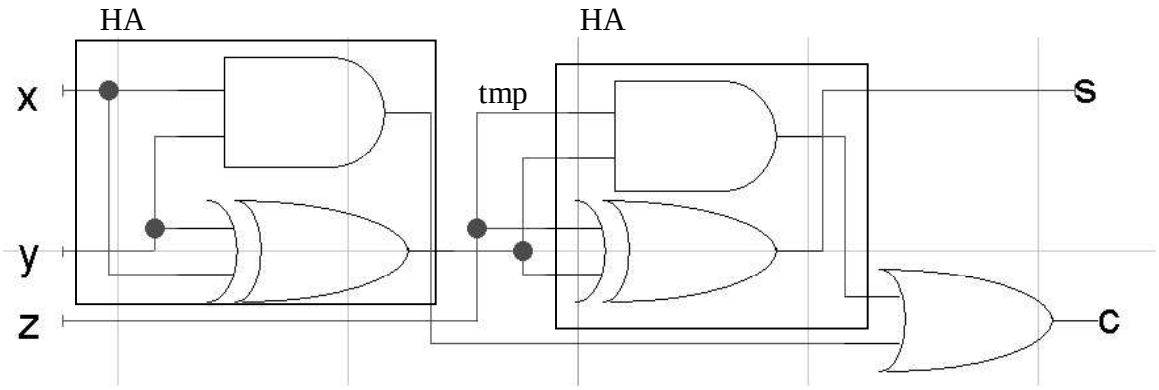
Lecture4
May 12

Building an Adder

- We will build a full adder



- First, a description of the entity:

```
library ieee;
use ieee.std_logic_1164.all;
entity full_adder is
        port (
                x, y, z  : in std_logic;
                s, c     : out std_logic;
        );
end full_adder
```

- Now, a behavioural description

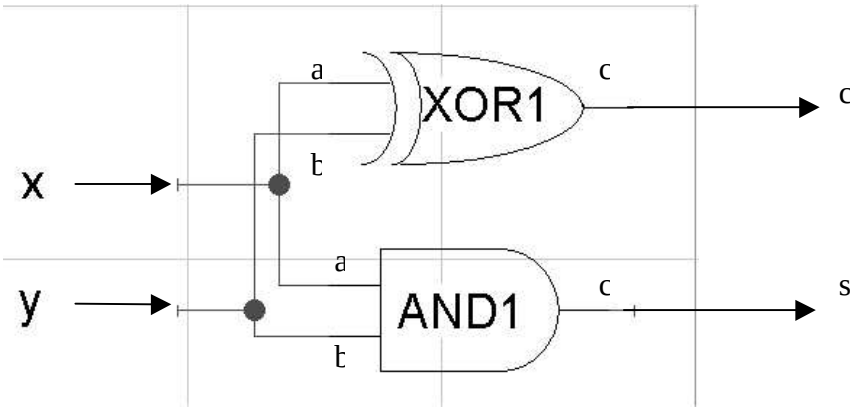
```
Architecture behav of full_adder is
Begin
        FA: process (x, y, z)
                Variable tmp: std_logic;
        Begin
                Tmp:= X xor Y;
                S<= tmp xor Z after 3 ns;
                C<= (X and Y ) or (Z and tmp) after 5 ns;
        End process;
End behave;
```

- Tmp is a variable
  - Variables aren't signals
  - Used like varabiles in a programming language –to hold info
  - Variables have nothing to do w/ structural descriptions or in circuits
  - Assignment to a variable is done with := NOT <=
- The sequential statements execute in order

- The circuits total T_{pol} (propogation delay is 5 ns;
- The assignments execute at the same simulation time
- The “after” just delays the result

- Now, a structural description:
  - First, we need gates:
    - and_gate
    - xor_gate
    - or_gate
  - now, we can build a half adder:

```
entity half_adder is
        port(
                x, y: in std_logic;
                s, c: out std_logic
                );
        end half_adder;
        architecture struct of half_adder is
                begin
                        XOR1: entity work.xor_gate
                                port map (
                                        a=>x,
                                        b=>y,
                                        c=>s);
                        AND1: entity work.adn_gate
                                port map(
                                        a=>x,
                                        b=>y,
                                        c=>s);
        end struct
```



- "struct" is the name of this implementation
- "XOR1" and "AND1" are the labels for the "entity instance"
  - an" instance" is basically a copy of whatever we defined the xor_gate/and_gate to be
- the "port map" connects signals in our entity to ports on the instance
  - x=> y indicates that port x on the instance, y is a signal in the entity we're defining