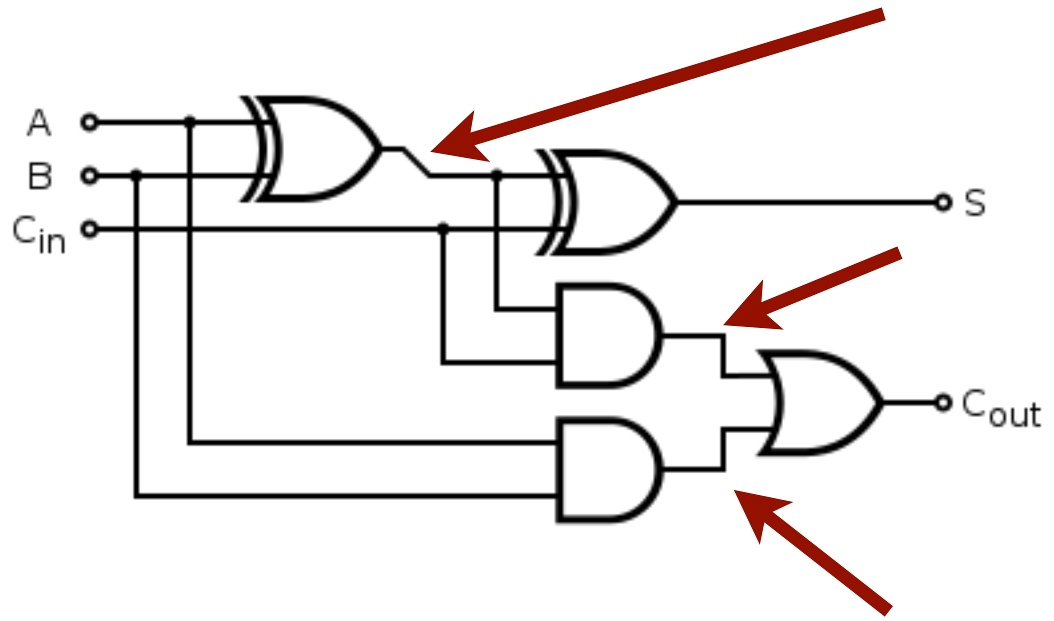
Practice Exercises

Topic #03 - d) Introduction to test-benches

Exercise #1- Implement a full-adder and test it using a complete test-bench



Inputs			Outputs	
A	*B*	*C_{in}*	*C_{out}*	*S*
0	0	0	0	0
1	0	0	0	1
0	1	0	0	1
1	1	0	1	0
0	0	1	0	1
1	0	1	1	0
0	1	1	1	0
1	1	1	1	1

- As a reminder, this is how a full adder looks like
- Use an internal signal at the end of each gate
- Make sure you declare these internal signals

```
ARCHITECTURE architecture_name OF entity_name IS
     [declarations]
BEGIN
     (code)
END architecture_name;
```

Main program

Test-bench

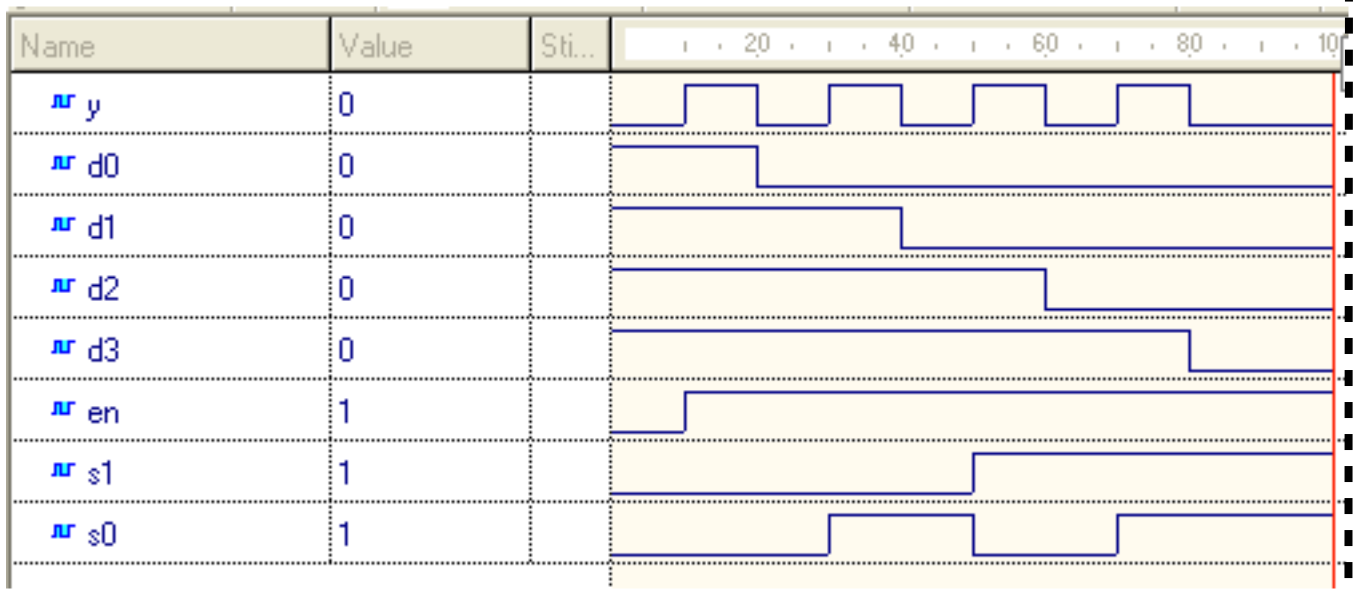
```
entity adder is
    port (
        a,b,cin: in bit;
         s,cout: out bit
         );
end entity adder;

architecture arch of adder is
    signal c,d,e : bit;
begin

    c <= a XOR b;
    s <= c XOR b;
    d <= b AND cin;
    e <= a AND b;
    cout <= d OR e;
end architecture arch;
```

```
entity  test_fa is
end;

architecture bench of test_fa is
  component adder
    port (a, b, cin: in bit;
    s,cout :out bit);
  end component;

  signal a, b, cin, s, cout: bit;

begin
 a   <= '0',
 '1' after 5ns,
 '0' after 10ns,
 '0' after 15 ns;

 b   <= '0',
 '1' after 5ns,
 '1' after 10ns,
 '0' after 15 ns;

 cin <= '0',
 '1' after 5ns,
 '1' after 10ns,
 '1' after 15 ns;

 m: adder port map (a, b, cin, s, cout);
end bench;
```

Exercise #2- Create a test-bench

- What does the code do?

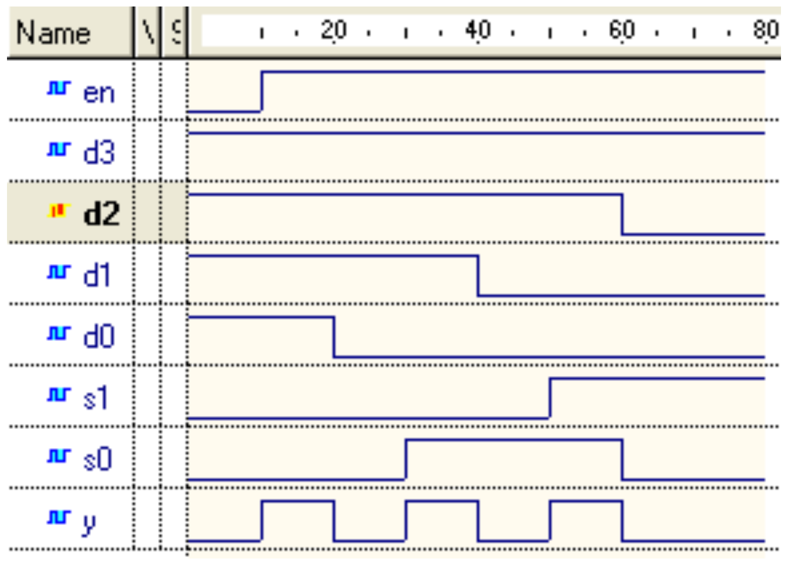
- Create an a test bench that will replicate the following waveforms



Name	Value
y	0
d0	0
d1	0
d2	0
d3	0
en	1
s1	1
s0	1

```
library IEEE;
use IEEE.std_logic_1164.all;

entity ex3 is
    port (
        en,d3,d2,d1,d0,s1,s0: in STD_LOGIC;
         y: out STD_LOGIC
         );
end entity ex3;

architecture arch of ex3 is
begin
y<='0' when (en='0') else
    d0 when (s1='0' and s0='0') else
    d1 when (s1='0' and s0='1') else
    d2 when (s1='1' and s0='0') else
    d3;
end architecture arch;
```

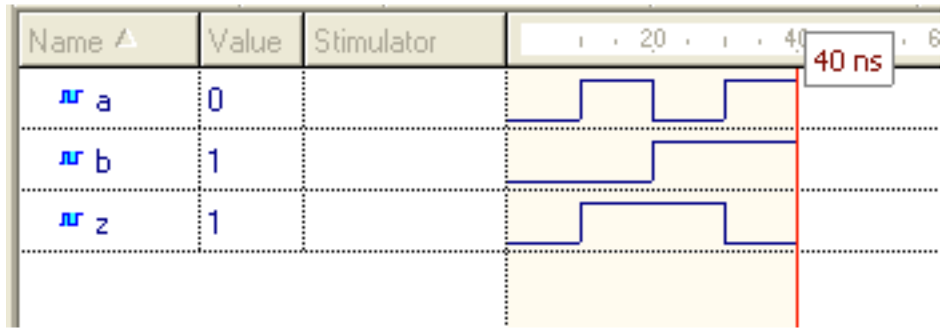
The code is for a slightly modified 4-to-1 MUX



```
library IEEE;
use IEEE.std_logic_1164.all;

entity  test_ex03 is
end;

architecture testbench of test_ex03 is

component ex03
    port (en, d3, d2, d1, d0, s1, s0: in STD_LOGIC;
    y : out STD_LOGIC);
  end component;

signal en, d3, d2, d1, d0,s1, s0, y : STD_LOGIC;

begin
 d0 <= '1', '0' after 20ns;
 d1 <= '1', '0' after 40ns;
 d2 <= '1', '0' after 60ns;
 d3 <= '1', '0' after 80ns;
 en <= '0', '1' after 10ns;
 s1 <= '0', '1' after 50ns;
 s0 <= '0', '1' after 30ns,
 '0' after 60ns,
 '1' after 80ns;

 m: ex03 port map (en, d3, d2, d1, d0,s1, s0, y);

end testbench;
```

Exercise #3 - Write the main VHDL code from the test-bench + waveform

```
entity testmaincode is
end;

architecture bench of testmaincode is
  component maincircuit
    port(a,b: in bit;
   z : out bit);

  end component;

  signal a,b,z: bit;

begin
 a  <= '0', '1' after 10ns, '0' after 20ns, '1' after 30ns, '0' after 40ns;
 b  <= '0', '1' after 20ns;

    m: maincircuit port map (a,b,z);

end bench;
```



Name	Value
a	0
b	1
z	1

Solution #3 - Write the main VHDL code from the test-bench + waveform

This is just an XOR gate

```
entity maincircuit is
   port(a,b: in bit;
   z : out bit);
end entity;

architecture myarch of maincircuit is
begin
   z <= a XOR b;
end architecture;
```