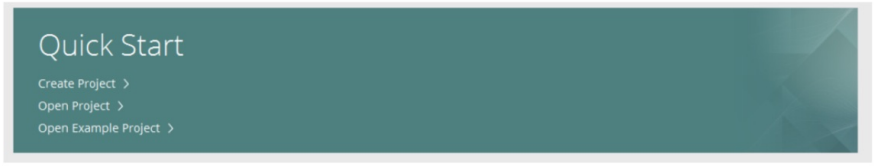
# PROJECT 1:
# 2-Bit Adder

# Xilinx Vivado & XSim

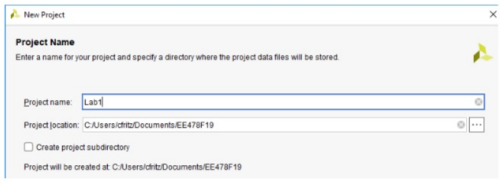
Using Vivado version 2024.1

Create a new project by pressing Create Project under Quick Start.




***Important:*** Do not assume that your project files and code will remain on the lab computers from week to week. After each lab, be sure to back up your code somewhere, or create the project on a flash drive.

Press Next, name the project Lab1, and select any location to save it. Deselect "Create project subdirectory".




Press Next, leave the "RTL Project" setting alone and press Next again.

Press Next on both the Add Sources and Add Constraints pages (we will create these later).

Press Next, then you will be prompted to select the FPGA device we are targeting. Search for "XC7Z010" and select "xc7z010clg400-1."

This depends on the FPGA device being used

Make sure to select this for the Zynq 7010 FPGA

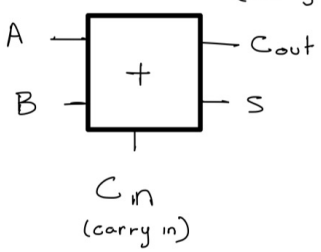
# Creating a VHDL Module (Full Adder)

Logic for full adder

ii. Full Adder

counts/sums 3 input bits

| A | B | Cin | Cout | S |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |




(carry out)

$S = A \oplus B \oplus C_{in}$ ← odd parity (even/address)

$C_{out} = AB + AC_{in} + BC_{in}$ ← majority function

S = 1 only when there's an odd number of 1's in the input (odd parity)

Fill in the missing architecture for the Full Adder module. Here's a start:

To determine the architecture for outputs S and Cout, refer to the truth table

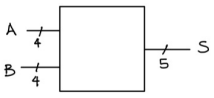
- Determine a logic that describes the relationship between inputs (A, B, Cin) and outputs (S, Cout)

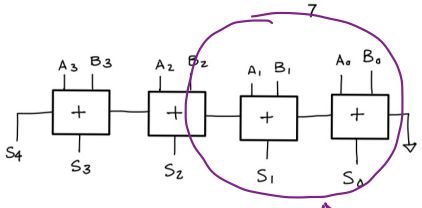
```
entity FullAdder is
    Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           Cin : in STD_LOGIC;
           S : out STD_LOGIC;
           Cout : out STD_LOGIC);
end FullAdder;

architecture Behavioral of FullAdder is

begin

S <=
Cout <=

end Behavioral;
```

$C_{in}$ = 1 only when the majority of the inputs are 1.



c. Example: 4-bit Ripple Carry Adder




4-bit Ripple Carry Adder Example

```
entity adder4 is
    port (
        A0, A1, A2, A3      : in std_logic;
        B0, B1, B2, B3      : in std_logic;
        S0, S1, S2, S3, S4 : out std_logic
    );
end adder4;

architecture arch of adder4 is

signal C0, C1, C2 : std_logic;

component full_adder is
    port (
        A    : in std_logic;
        B    : in std_logic;
        Cin  : in std_logic;
        S    : out std_logic;
        Cout : out std_logic
    );
end component;

begin

adder0 : full_adder port map (A => A0, B => B0, Cin => '0', Cout => C0, S => S0);
adder1 : full_adder port map (A => A1, B => B1, Cin => C0, Cout => C1, S => S1);
adder2 : full_adder port map (A => A2, B => B2, Cin => C1, Cout => C2, S => S2);
adder3 : full_adder port map (A => A3, B => B3, Cin => C2, Cout => S4, S => S3);

end arch;
```

- I need 2 bits so I look at the first 2 bits

# FullAdder Module

Project Summary | Adder2_Tbench.vhd | Adder2.vhd | **FullAdder.vhd**

E:/FPGA/project_1_Lab1/project_1_Lab1.srcs/sources_1/new/FullAdder.vhd

```
22  library IEEE;
23  use IEEE.STD_LOGIC_1164.ALL;
24
25  -- Uncomment the following library declaration if using
26  -- arithmetic functions with Signed or Unsigned values
27  --use IEEE.NUMERIC_STD.ALL;
28
29  -- Uncomment the following library declaration if instantiating
30  -- any Xilinx leaf cells in this code.
31  --library UNISIM;
32  --use UNISIM.VComponents.all;
33
34  entity FullAdder is
35      Port ( A : in STD_LOGIC;
36             B : in STD_LOGIC;
37             Cin : in STD_LOGIC;
38             S : out STD_LOGIC;
39             Cout : out STD_LOGIC);
40  end FullAdder;
41
42  architecture Behavioral of FullAdder is
43
44  begin
45
46  S <= A xor B xor Cin;
47  Cout <= (A and B) or (A and Cin) or (B and Cin);
48
49
50  end Behavioral;
51
```

FullAdders are instantiated into 2-bit Adder Design

(NEXT PAGE)

In VHDL

$S = A \oplus B \oplus C_{in}$ → `S <= A xor B xor Cin;`

$C_{out} = AB + AC_{in} + BC_{in}$ → `Cout <= (A and B) or (A and Cin) or (B and Cin);`

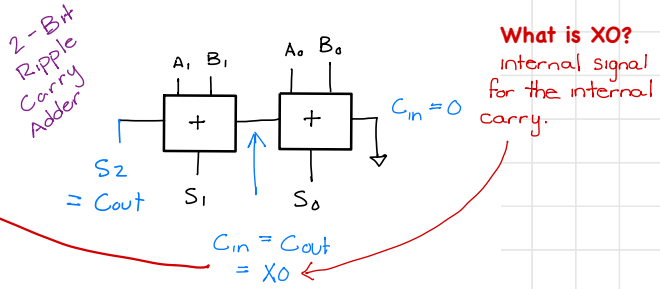
↑ verified by Truth Table (PAGE 3)

# 2-bit Adder Module

Project Summary | Adder2_Tbench.vhd | **Adder2.vhd** | FullAdder.vhd | Lab1.xdc

E:/FPGA/project_1_Lab1/project_1_Lab1.srcs/sources_1/new/Adder2.vhd

```
20
21
22  library IEEE;
23  use IEEE.STD_LOGIC_1164.ALL;
24
25  -- Uncomment the following library declaration if using
26  -- arithmetic functions with Signed or Unsigned values
27  --use IEEE.NUMERIC_STD.ALL;
28
29  -- Uncomment the following library declaration if instantiating
30  -- any Xilinx leaf cells in this code.
31  --library UNISIM;
32  --use UNISIM.VComponents.all;
33
34  entity Adder2 is
35      Port ( A0 : in STD_LOGIC;
36             A1 : in STD_LOGIC;
37             B0 : in STD_LOGIC;
38             B1 : in STD_LOGIC;
39             S0 : out STD_LOGIC;
40             S1 : out STD_LOGIC;
41             S2 : out STD_LOGIC);
42  end Adder2;
43
44  architecture Behavioral of Adder2 is
45
46  signal X0 : std_logic;
47
48  component FullAdder is
49
50      port (
51      A, B, Cin   : in std_logic;
52      S, Cout     : out std_logic
53      );
54
55  end component;
56
57  begin
58
59  fa0: FullAdder port map(A => A0, B => B0, Cin => '0', S => S0, Cout => X0);
60  fa1: FullAdder port map(A => A1, B => B1, Cin => X0, S => S1, Cout => S2);
61
62  end Behavioral;
63
```




2 full adder instantiated here

# Creating a Test bench to simulate my design

ALWAYS SIMULATE DESIGN BEFORE HARDWARE TEST

Project Summary | Adder2_Tbench.vhd | Adder2.vhd | FullAdder.vhd | Lab1.xdc

E:/FPGA/project_1_Lab1/project_1_Lab1.srcs/sim_1/new/Adder2_Tbench.vhd

```
1  ----------------------------------------------------------------------------...
20
21
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24 -- Uncomment the following library declaration if using
25 -- arithmetic functions with Signed or Unsigned values...
33 architecture Behavioral of Adder2_Tbench is
34 component Adder2 is
35     port (
36     A0, A1, B0, B1    : in std_logic;
37     S0, S1, S2        : out std_logic
38     );
39 end component;
40 signal A0, A1, B0, B1, S0, S1, S2    : std_logic;
41 begin
42 uut : Adder2 port map (
43 A0 => A0,
44 A1 => A1,
45 B0 => B0,
46 B1 => B1,
47 S0 => S0,
48 S1 => S1,
49 S2 => S2
50 );
51 stim_proc : process
52 begin
53     wait for 100 ns;
54     A0 <= '0';
55     A1 <= '1';
56     B0 <= '0';
57     B1 <= '1';
58
59     wait for 100 ns;
60     A0 <= '1';
61     A1 <= '1';
62     B0 <= '1';
63     B1 <= '1';
64     wait;
65 end process stim_proc;
66
67 end Behavioral;
68
69
```

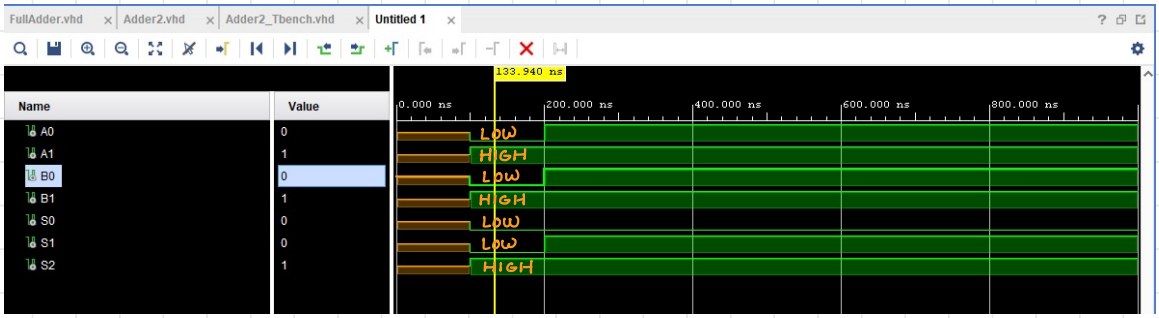
Declare Adder2 module as a component

Internal signals declared inputs / outputs

Unit Under Test (UUT) Adder2 instantiated here

Stimulus process with test cases for simulation

# Simulation

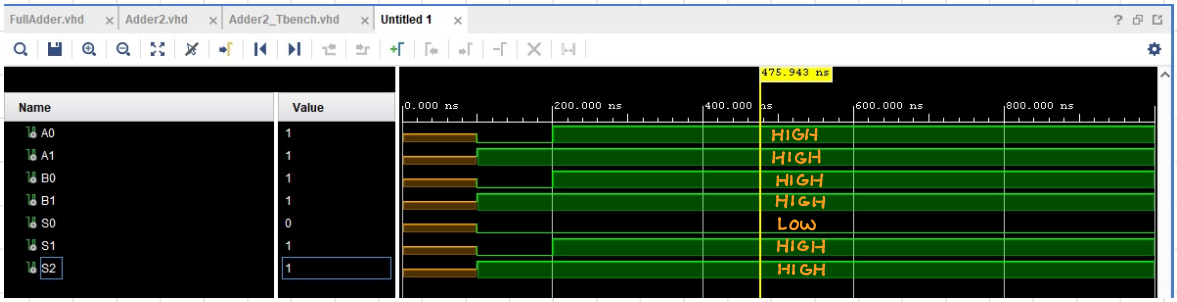



## CASE 1

| A1 | A0 | | B1 | B0 | | $S_2$ | $S_1$ | $S_0$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | + | 1 | 0 | → | 1 | 0 | 0 |
| HIGH | LOW | | HIGH | LOW | | HIGH | LOW | LOW |

Binary Addition

$$\begin{array}{r} 10 \\ +\ 10 \\ \hline 100 \end{array} \quad \rightarrow \quad \begin{array}{r} 2 \\ +\ 2 \\ \hline 4 \end{array}$$




## CASE 2

| A1 | A0 | | B1 | B0 | | $S_2$ | $S_1$ | $S_0$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | + | 1 | 1 | → | 1 | 1 | 0 |
| HIGH | HIGH | | HIGH | HIGH | | HIGH | HIGH | LOW |

Binary Addition

$$\begin{array}{r} 11 \\ +\ 11 \\ \hline 110 \end{array} \quad \rightarrow \quad \begin{array}{r} 3 \\ +\ 3 \\ \hline 6 \end{array}$$

# . xdc file

Here, I'm mapping each I/O in my design to corresponding pins on the board.

- Pins on my Zybo Board
- I/O in my design

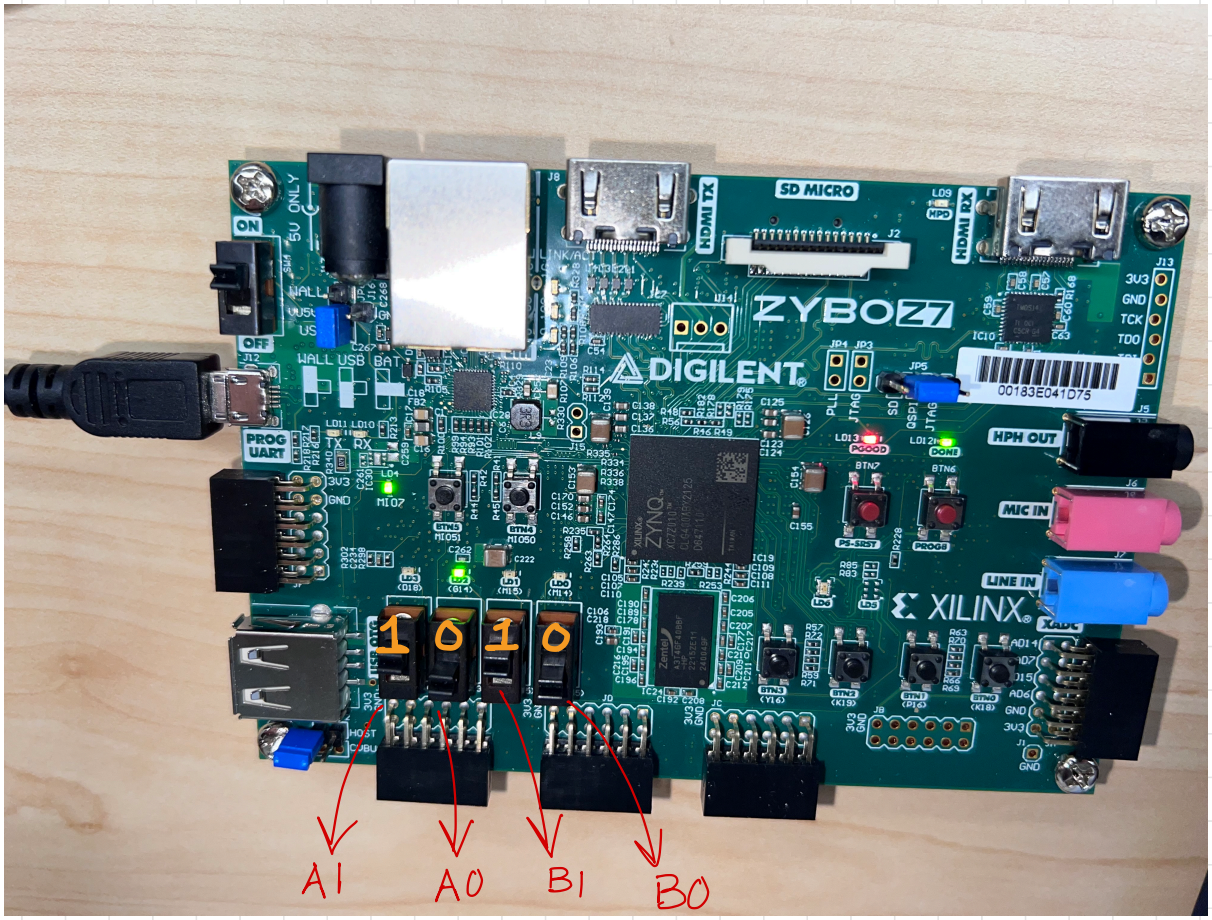
```
1  set_property PACKAGE_PIN T16 [get_ports {A1}]
2  set_property IOSTANDARD LVCMOS33 [get_ports {A1}]
3  set_property PACKAGE_PIN W13 [get_ports {A0}]
4  set_property IOSTANDARD LVCMOS33 [get_ports {A0}]
5  set_property PACKAGE_PIN P15 [get_ports {B1}]
6  set_property IOSTANDARD LVCMOS33 [get_ports {B1}]
7  set_property PACKAGE_PIN G15 [get_ports {B0}]
8  set_property IOSTANDARD LVCMOS33 [get_ports {B0}]
9  set_property PACKAGE_PIN G14 [get_ports {S2}]
10 set_property IOSTANDARD LVCMOS33 [get_ports {S2}]
11 set_property PACKAGE_PIN M15 [get_ports {S1}]
12 set_property IOSTANDARD LVCMOS33 [get_ports {S1}]
13 set_property PACKAGE_PIN  [get_ports {S0}]
14 set_property IOSTANDARD LVCMOS33 [get_ports {S0}]
15
```

M14

This will change depending on the FPGA being used

# Hardware Test

CASE 1




| A1 | A0 | B1 | B0 |
|---|---|---|---|
| 1 | 0 | 1 | 0 |

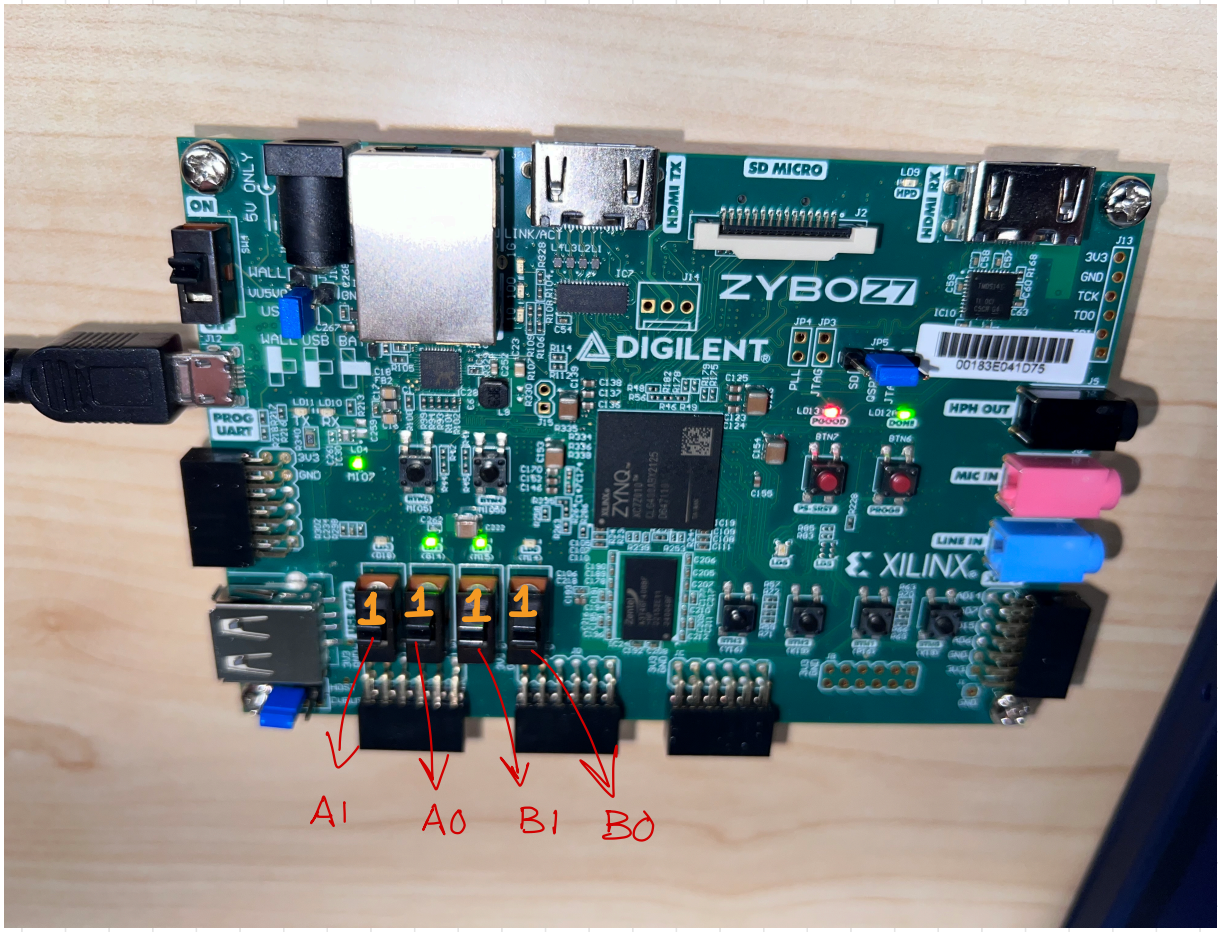
← Switches on Zybo board set to this

LED S2 is ON, S1 and S0 is OFF

| S2 | S1 | S0 |
|---|---|---|
| 1 | 0 | 0 |

← Adds 10 + 10 = 100

## CASE 2




| A1 | A0 | B1 | B0 |
|---|---|---|---|
| 1 | 1 | 1 | 1 |

← Switches on Zybo board set to this

LED S0 is OFF, S1 and S2 is ON

| S2 | S1 | S0 |
|---|---|---|
| 1 | 1 | 0 |

← Adds 11 + 11 = 110

**2-BIT ADDED SUCCESSFUL**

# NOTES

- Be sure to use a USB cable capable of transmitting data. Initially, Vivado Hardware Manager wasn't detecting FPGA because of the USB I was using.

- Get familiar with Vivado Project Manager