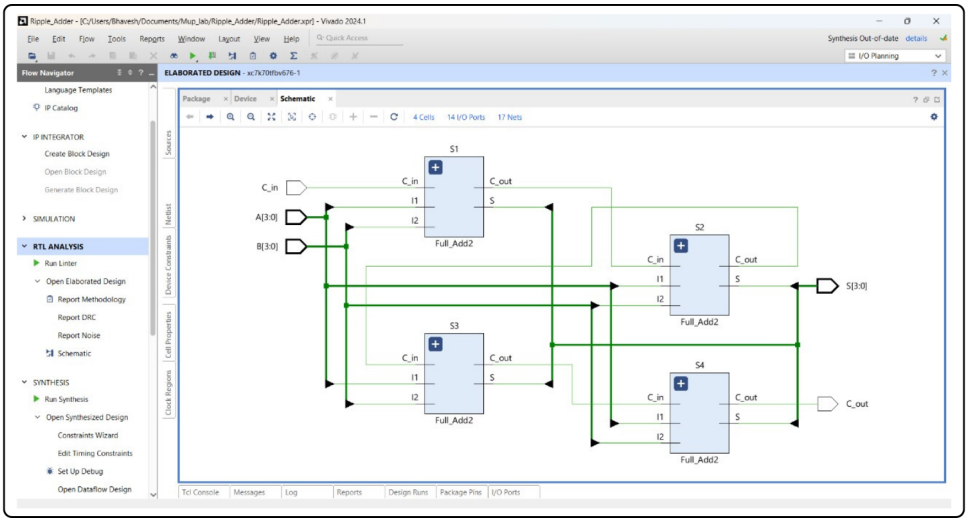

EE2016: Assignment-1

By Siva(EE23B151),
Bhavesh(EE23B017)
and Naveen(EE23B113)

Group 28

Date: 8/12/2024

Indian Institute of Technology Madras

Contents

1 Introduction

1.1 Xilinx Vivado

Xilinx Vivado is an *integrated design environment* (IDE) used for digital design and **hardware description languages** (HDL), specifically targeting Xilinx **Field Programmable Gate Arrays** (FPGAs) and System on Chips (SoCs). Vivado offers a comprehensive suite of tools for designing, simulating, analyzing, and implementing digital systems.

1.2 Hardware Description Language (HDL)

HDLs are used to **model** electronic systems at various levels of *abstraction*, from high-level functional descriptions down to detailed gate-level implementations.

1.3 Field Programmable Gate Arrays (FPGAs)

FPGAs are a type of digital integrated circuit (IC) that can be configured by the user after manufacturing, allowing it to be customized for specific tasks. Unlike fixed-function ICs like **CPUs** or **GPUs**, which have a *predefined* operation, FPGAs offer the flexibility to be **reprogrammed** to perform a wide variety of digital logic tasks.

The FPGA board we are going to use is the *Xilinx Artix-7* board (Fig.1). For this assignment, we will be using the On-Off *Switches* as **inputs** and *LEDs* for **outputs**. There are **16 switches** and **16 LEDs**. Each of these can be **mapped** to the variables we use in our *verilog* code by using the *.xdc* file (**constraint file**).

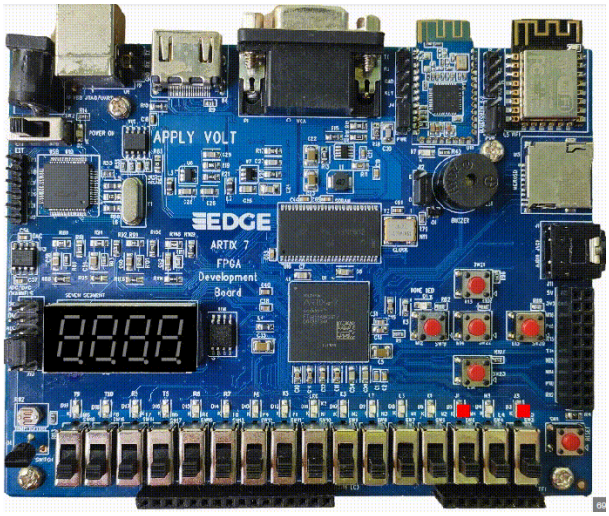

Figure 1: Xilinx Artix-7 FPGA board

1.4 Xilinx Design Constraints (.xdc) files

An XDC file is a *text file* used in FPGA design, **specifically** within the **Xilinx Vivado Design Suite**, to define constraints for your FPGA implementation. Constraints are *rules or guidelines* that direct the synthesis, placement, and routing tools on how to handle your design, ensuring that it meets specific performance criteria and correctly interfaces with external hardware. In this experiment, we will be using it for **I/O pin assignments**.

2 Half-Adder Simulation and Realization on the FPGA Board

2.1 Half-Adder Circuit

A and B are binary inputs and this system gives binary outputs S (sum) and C (carry out) which can be used to find the sum of A and B.

$$S = A \oplus B \quad \& \quad C = A.B$$

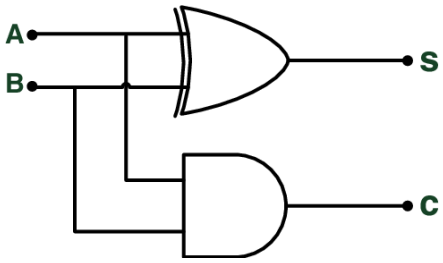

Figure 2: Half-Adder logic circuit

2.2 Verilog Codes and Constraint files

Click the blue hyperlinks:

- **Half-Adder Module**: contains the main module.
- **Testbench**: used for testing the Half-Adder module.
- **Constraint file**: used for mapping the variables in Half-Adder module to respective i/o ports in FGPA board.

2.3 Outputs

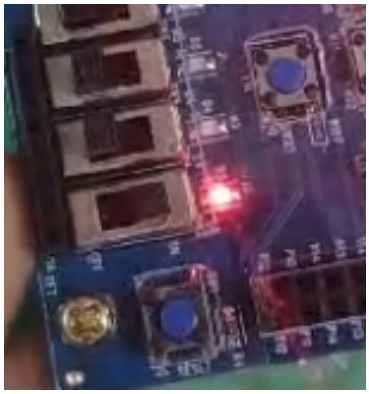

Fig 3(a): $a = 1, b = 0$
$s = 1, c_{out} = 0$

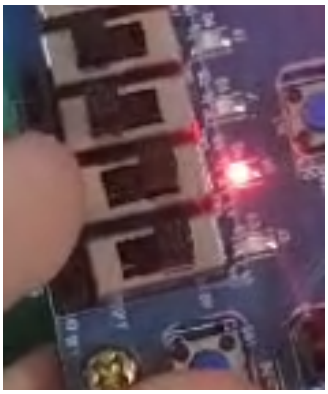

Fig 3(b): $a = 1, b = 1$
$s = 0, c_{out} = 1$

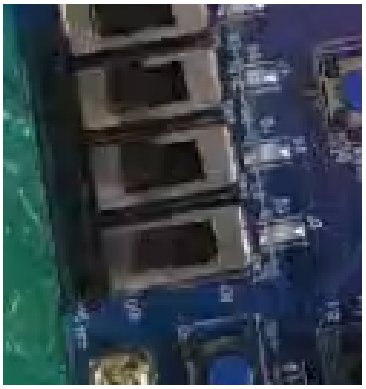

Fig 3(c): $a = 0, b = 0$
$s = 1, c_{out} = 1$

Figure 3: Outputs for different inputs

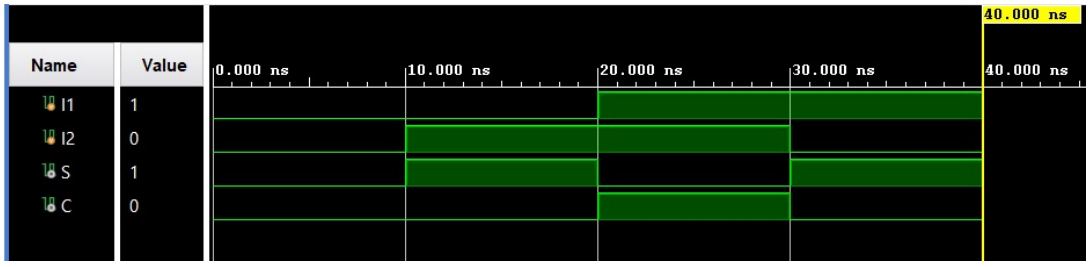

Figure 4: Simulation Graph

CLICK HERE to see full utilization report.

Resource	Utilization	Available	Utilization...
LUT	1	63400	0.01
IO	4	210	1.90

Figure 5: No. of LUTs used

3 Full-Adder Simulation and Realization on the FPGA Board

3.1 Full-Adder Circuit

A, B and C_{in} (carry in) are binary inputs and this system gives binary outputs S (sum) and C_{out} (carry out) which can be used to find the sum of A and B.

$$S = A \oplus B \oplus C_{in}$$

$$C_{out} = A.B + (A \oplus B).C_{in}$$

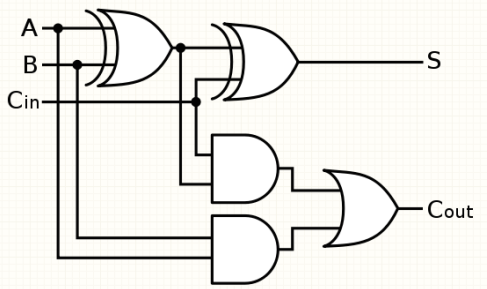

Figure 6: Full-Adder logic circuit

3.2 Verilog Codes and Constraint files

Click the blue hyperlinks:

- **Full-Adder Module**: contains the main module.
- **Testbench**: used for testing the Full-Adder module.
- **Constraint file**: used for mapping the variables in Full-Adder module to respective i/o ports in FGPA board.

3.3 Outputs

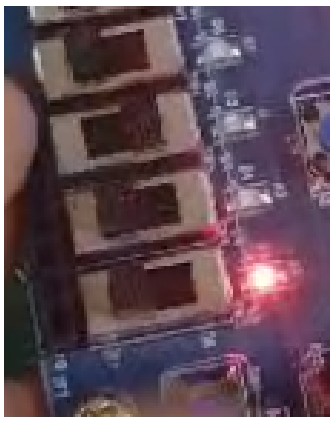

Fig 7(a): $a = 0, b = 0, c_{in} = 1$
$s = 1, c_{out} = 0$

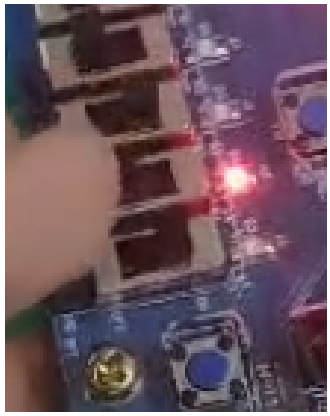

Fig 7(b): $a = 1, b = 0, c_{in} = 1$
$s = 0, c_{out} = 1$

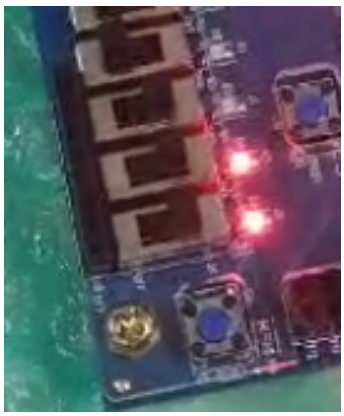

Fig 7(c): $a = 1, b = 1, c_{in} = 1$
$s = 1, c_{out} = 1$

Figure 7: Outputs for different inputs

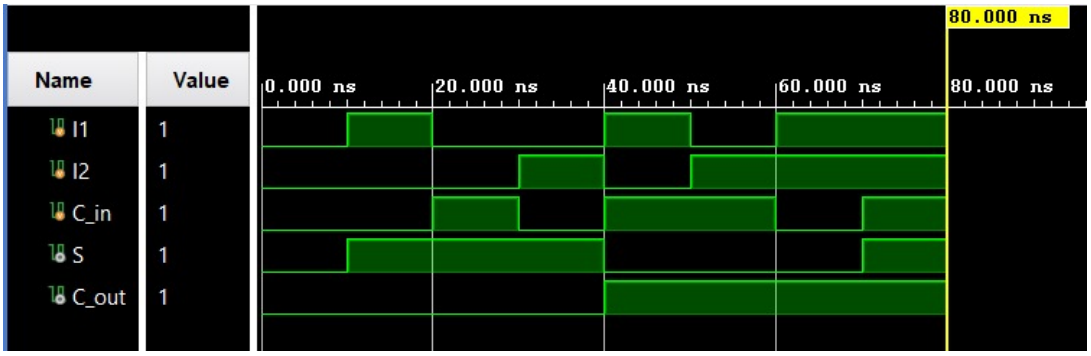

Figure 8: Simulation Graph

Resource	Estimation	Available	Utilization...
LUT	1	1182240	0.01
IO	5	676	0.74

Figure 9: No. of LUTs used

CLICK HERE to see full utilization report.
CLICK HERE to see video of circuit testing.

4 Ripple Carry Adder Simulation and Realization on the FPGA Board

4.1 Ripple Carry Adder Circuit

A and B (carry in) are **4-bit binary** inputs and C_0 is a 1-bit binary input. This system gives 4-bit binary output S (sum) and 1-bit binary output C_{out} (carry out) which can be used to find the sum of the 4-bit numbers, A and B.

$$S_i = A_i \oplus B_i \oplus C_i$$

$$C_{i+1} = A_i.B_i + (A_i \oplus B_i).C_i$$

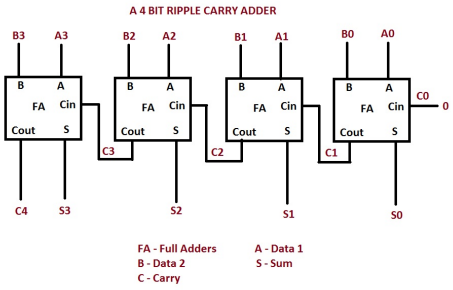

Figure 10: Ripple Carry Adder

4.2 Verilog Codes and Constraint files

Click the blue hyperlinks:

- **Full-Adder Module**: it is called by the main module.
- **Ripple Carry Adder Module**: contains the main module.
- **Testbench**: used for testing the Full-Adder module.
- **Constraint file**: used for mapping the variables in Ripple Carry Adder module to respective i/o ports in FGPA board.

4.3 Outputs

CLICK HERE to see full utilization report.
CLICK HERE to see video of circuit testing.

Resource	Estimation	Available	Utilization...
LUT	4	41000	0.01
IO	14	300	4.67

Figure 11: No. of LUTs used

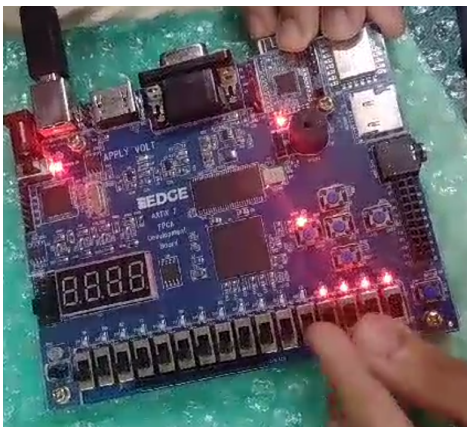

Fig 12(a): $a = (1111)_2, b = (0000)_2, c_{in} = 0$
$s = (1111)_2, c_{out} = 0$

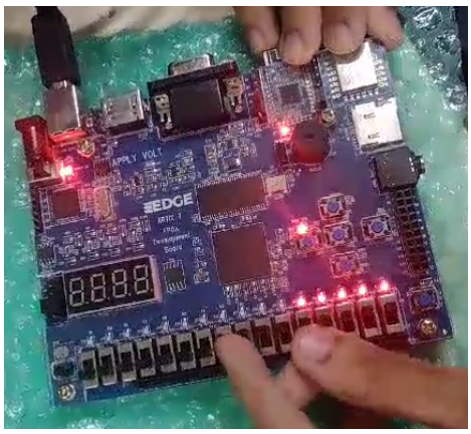

Fig 12(b): $a = (1111)_2, b = (1111)_2, c_{in} = 1$
$s = (1111)_2, c_{out} = 1$

Figure 12: Outputs for different inputs

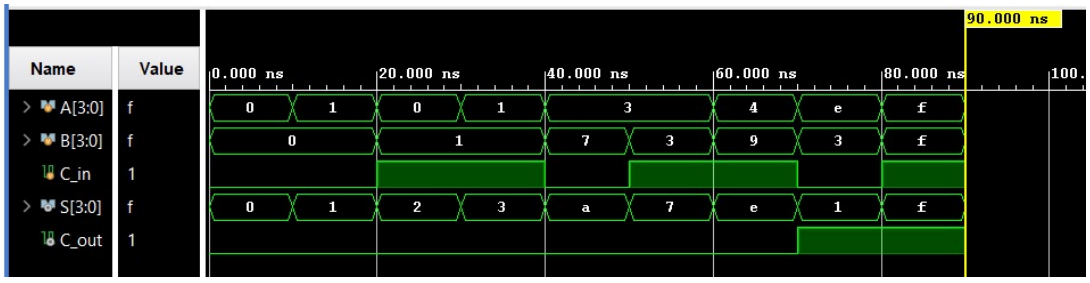

Figure 13: Simulation Graph