MICROPROCESSOR LAB EXPERIMENT 1

Deenabandhan N ee23b021
Sai Harshith Gajendra ee23b069
Krutarth Patel ee23b137

7 August 2024

Introduction

FPGA board : Edge Artix 7

This experiment involves

- Simulating a half-adder using Xilinx Vivado and implementing on the FPGA board.
- Extending the half-adder design to a full-adder, simulating it and implementing on the FPGA board.
- Designing a 4-bit ripple-carry adder and implementing on the FPGA board.

Xilinx Vivado

- We were introduced to Xilinx vivado , a software that is used to synthesize and analyse the hardware description designs.
- The procedures that we followed are ,
  - Create a project with source file as our Verilog code.
  - Add constraints to the ports that we have defined in the Verilog code.
  - Run the synthesis to check whether our Verilog code has any error in it.
  - We can open the Schematics to see the design that we have coded in Verilog.
  - After running synthesis , we can run simulation using our testbench and check for logical errors.
  - Once we confirm there is no logical/syntax error , we can run the implementation which basically implements our hardware description in the board which we have chosen while creating the project.
  - After the implementation is done , we can generate bitstream which is basically a file that contains the configuration information for an FPGA.
  - Once we successfully generate the bitstream , we can connect the target source and program it with the generated bitstream.
- In this report , we have included
  - Verilog code for module instantiation.
  - We have included both data flow as well as gate level modelling here.
  - The Schematics generated from Xilinx Vivado.
  - The Constraints file generated for FPGA.
  - The testbench and simulation generated
  - The analysis of reports obtained from Xilinx Vivado.

Half Adder

Data flow model

```
module half_adder(input bit1, bit2,
                  output cout, sum);

    assign sum = bit1 ^ bit2;
    assign cout = bit1 & bit2;

endmodule
```

Schematics

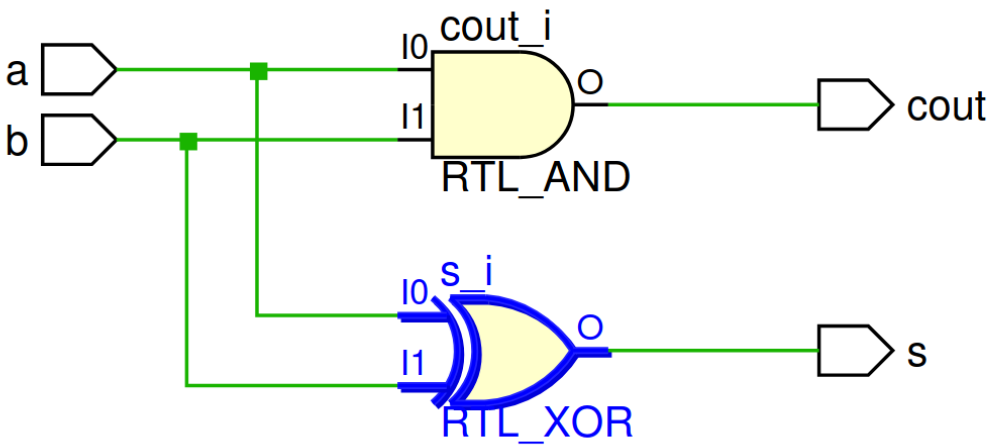


Figure 1: Schematics of the Data flow modelling

Gate level model

```
module half_adder(input bit1, bit2,
                  output cout, sum);

    xor(sum, bit1, bit2);
    and(cout, bit1, bit2);

endmodule
```

Constraints on ports of FPGA

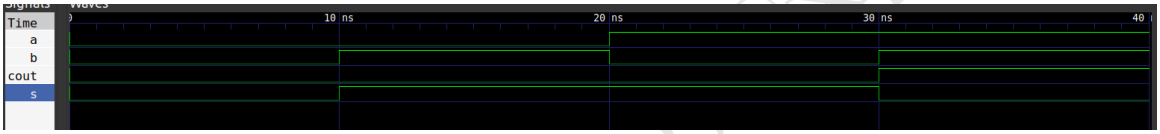
```
    set_property IOSTANDARD LVCMOS33 [get_ports a]
    set_property IOSTANDARD LVCMOS33 [get_ports b]
    set_property IOSTANDARD LVCMOS33 [get_ports cout]
    set_property IOSTANDARD LVCMOS33 [get_ports s]

    set_property PACKAGE_PIN L4 [get_ports a]
    set_property PACKAGE_PIN L5 [get_ports b]
    set_property PACKAGE_PIN J3 [get_ports cout]
    set_property PACKAGE_PIN H3 [get_ports s]
```

Testbench

```
`timescale 1ns/100ps
```

```
module frbit_adder;
        wire s,cout;
        reg a;
        reg b;
        half_adder t(a,b,cout,s);
        initial
        begin
                $dumpfile("test_circuit.vcd");
                $dumpvars(0,frbit_adder);
                a=0;b=0;
                #10 b=1;
                #10 a=1;b=0;
                #10 b=1;

                #10 $finish;

        end
endmodule
```

Simulation



Reports :

Resources utilized :

- For this code, the FPGA has used one **SLICE LUT** where LUT is used as a logic and not memory.
- We can also infer that we have not used any clocking resources or RAM memory.
- As we have not used any clock resources in this experiment, there are no timing constraints.

Time delays

Type of Path	*Path taken*	**Time delay (ns)**
Max Delay path	**a → cout**	8.789
	a → s	8.496
Min Delay path	**b → cout**	2.426
	b → s	2.347

Power consumed

- The power consumed by FPGA is **1.988 W**

Full Adder

Code

Data flow model

```
module full_adder_dataflow(input a,
                           input b,
                           input cin,
                           output s,
                           output cout);

    assign {cout, s} = {a + b + cin};

endmodule
```

Schematics

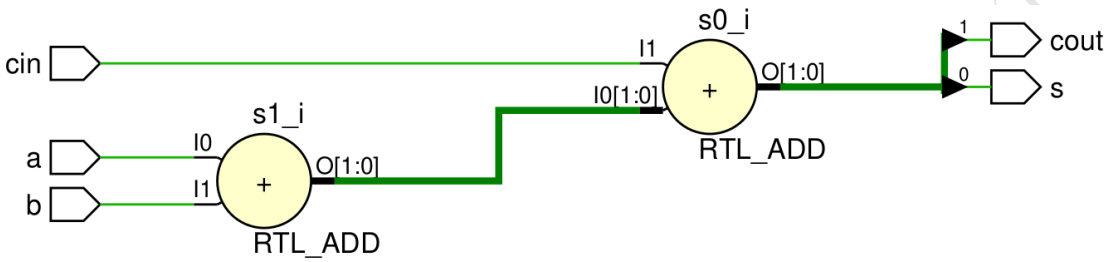


Figure 2: Schematics of the Data flow modelling

Gate level model

```
module full_adder_gate_level(input a,
                             input b,
                             input cin,
                             output s,
                             output cout);

    wire half_adder_sum, half_adder_cout, full_cout;

    xor(half_adder_sum, a, b);
    and(half_adder_cout, a, b);

    xor(sum, half_adder_sum, cin);
    and(full_cout, half_adder_sum, cin);
    xor(cout, half_cout, half_cout);

endmodule
```

Schematics

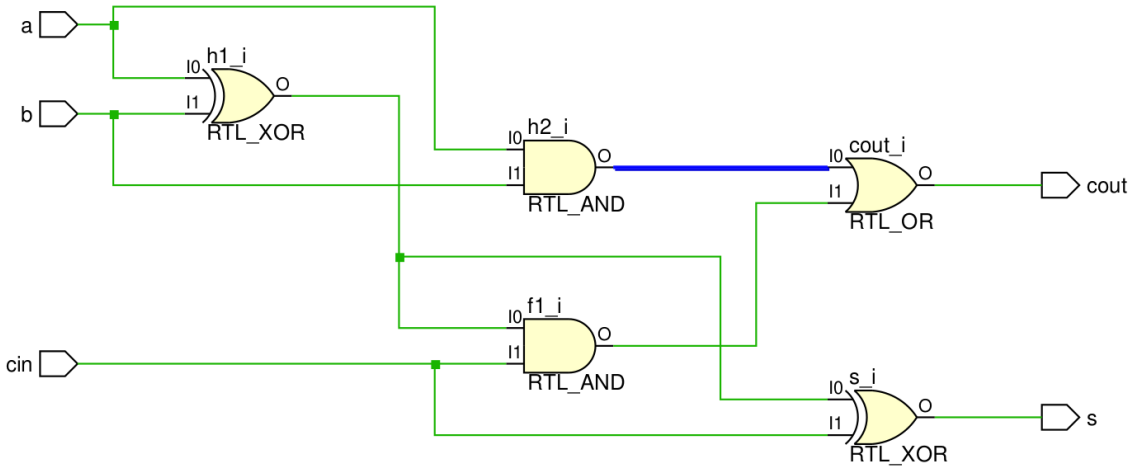


Figure 3: Schematics of the Gate level modelling

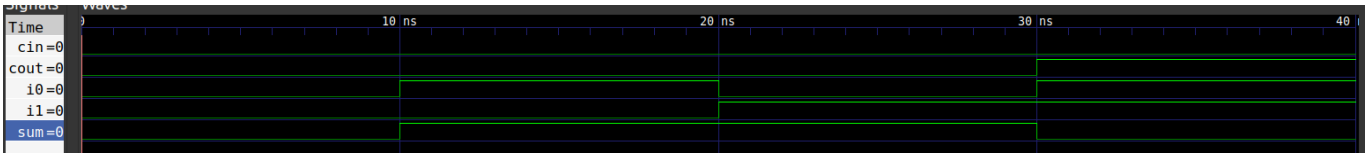
Constraints on ports of FPGA

```
set_property IOSTANDARD LVCMOS33 [get_ports a]
set_property IOSTANDARD LVCMOS33 [get_ports a1]
set_property IOSTANDARD LVCMOS33 [get_ports b]
set_property IOSTANDARD LVCMOS33 [get_ports b1]
set_property IOSTANDARD LVCMOS33 [get_ports cin]
set_property IOSTANDARD LVCMOS33 [get_ports cin1]
set_property IOSTANDARD LVCMOS33 [get_ports cout]
set_property IOSTANDARD LVCMOS33 [get_ports s]
set_property PACKAGE_PIN L5 [get_ports a]
set_property PACKAGE_PIN J3 [get_ports a1]
set_property PACKAGE_PIN L4 [get_ports b]
set_property PACKAGE_PIN H3 [get_ports b1]
set_property PACKAGE_PIN M4 [get_ports cin]
set_property PACKAGE_PIN J1 [get_ports cin1]
set_property PACKAGE_PIN K1 [get_ports cout]
set_property PACKAGE_PIN L3 [get_ports s]
```

Testbench

```
`timescale 1ns/100ps

module test_FA;
        wire sum,cout;
        reg i0,i1,cin;
        full_adder_dataflow DUT(i0,i1,cin,sum,cout);

        initial
        begin
                $dumpfile("test_circuit.vcd");
                $dumpvars(0,test_FA);
                $monitor("At t=%t i0=%b i1=%b cin=%b cout=%b sum=%b ",$time,i0,i1,cin,cout,sum);

                i0=0; i1=0; cin=0;
                #10 i0=1; i1=0;
                #10 i0=0; i1=1;
                #10 i0=1; i1=1;
                #10;
        end
endmodule
```

Simulation :



Reports :

Resources utilized :

- For this code, the FPGA has used one **SLICE LUT** where LUT is used as a logic and not memory.
- We can also infer that we have not used any clocking resources or RAM memory.
- As we have not used any clock resources in this experiment, there are no timing constraints.

Time delays

Type of Path	*Path taken*	**Time delay (ns)**
Max Delay path	**a** $\rightarrow$ **cout**	8.486
	a $\rightarrow$ **s**	8.110
Min Delay path	**b** $\rightarrow$ **cout**	2.446
	b $\rightarrow$ **s**	2.318

Power consumed

- The power consumed by FPGA is **4.882 W**

Ripple Carry Adder

Data flow model

```
module ripple_carry_adder_data_flow  (input cin,
                                      input[3:0] a, b,
                                      output cout,
                                      output[3:0] s);

    assign {cout, s} = {a + b + cin};

endmodule
```

Schematics :

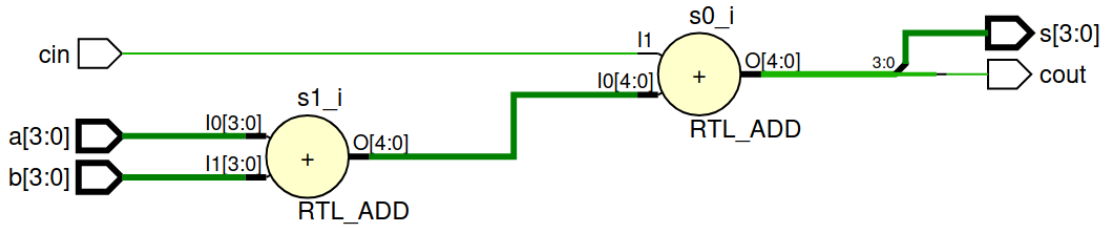


Figure 4: Schematics of the Data flow modelling

Gate level model

```
module HA(sum,cout,a,b);

        input a,b;
        output sum,cout;

        xor x1(sum,a,b);
        and x2(cout,a,b);

endmodule



module FA(sum,cout,a,b,cin);

        input a,b,cin;
        output sum,cout;

        wire s1,s2,s3,s4,s5;

        xor x1(s1,a,b);
        and a1(s2,a,b);
        xor x2(sum,s1,cin);
        and a2(s3,s1,cin);
        or o1(cout,s2,s3);

endmodule


module Frb(out,a,b);

        input [3:0] a;
        input [3:0] b;
        output [4:0] out;
        wire h0,h1,h2;
```

```
        HA h(out[0],h0,a[0],b[0]);
        FA f1(out[1],h1,a[1],b[1],h0);
        FA f2(out[2],h2,a[2],b[2],h1);
        FA f3(out[3],out[4],a[3],b[3],h2);

endmodule
```

Schematics :

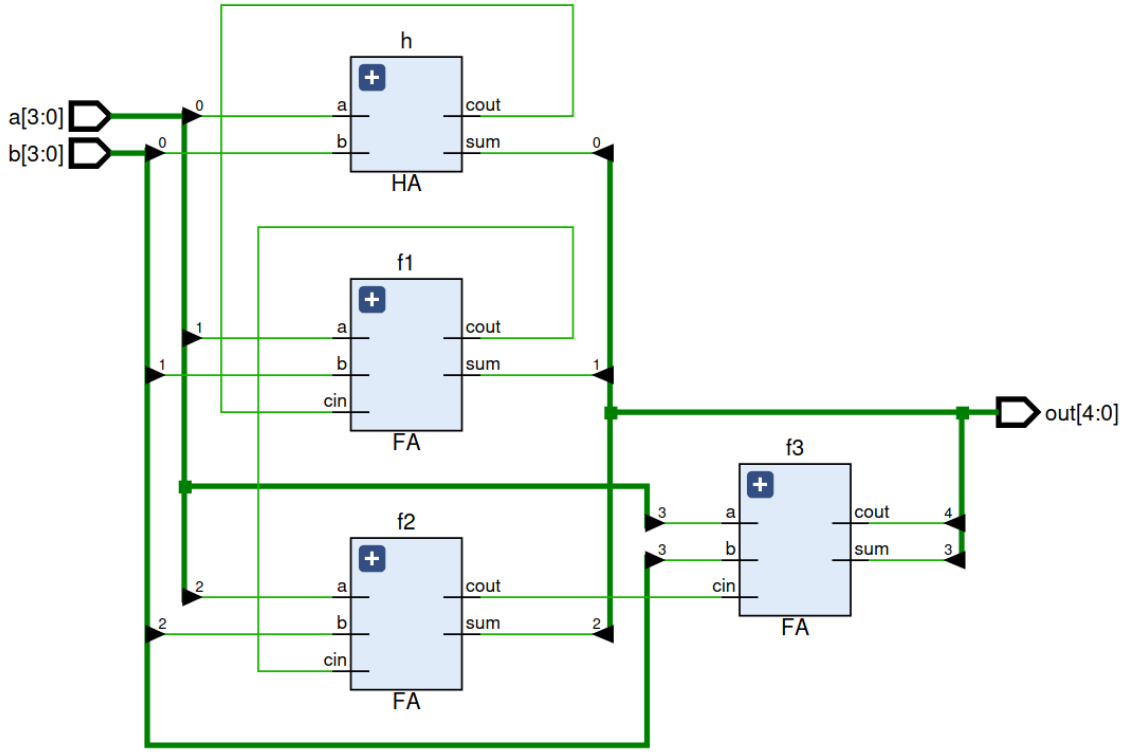


Figure 5: Schematics of the Gate level modelling

Constraints on ports of FPGA

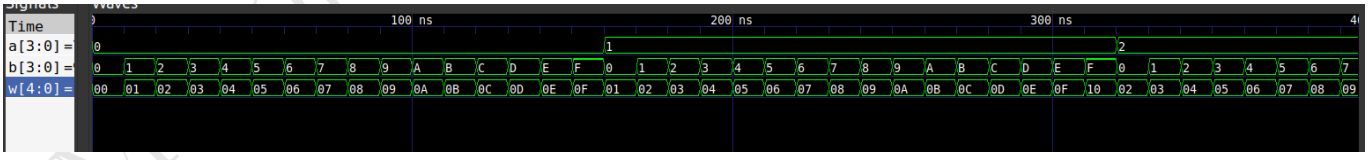
```
set_property IOSTANDARD LVCMOS33 [get_ports {a[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {b[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {b[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {b[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {b[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {s[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {s[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {s[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {s[0]}]
set_property PACKAGE_PIN L5 [get_ports {a[3]}]
set_property PACKAGE_PIN L4 [get_ports {a[2]}]
set_property PACKAGE_PIN M4 [get_ports {a[1]}]
set_property PACKAGE_PIN M2 [get_ports {a[0]}]
set_property PACKAGE_PIN M1 [get_ports {b[3]}]
set_property PACKAGE_PIN N3 [get_ports {b[2]}]
set_property PACKAGE_PIN N2 [get_ports {b[1]}]
set_property PACKAGE_PIN N1 [get_ports {b[0]}]
set_property PACKAGE_PIN J3 [get_ports {s[3]}]
set_property PACKAGE_PIN H3 [get_ports {s[2]}]
```

```
set_property PACKAGE_PIN J1 [get_ports {s[1]}]
set_property PACKAGE_PIN K1 [get_ports {s[0]}]
set_property PACKAGE_PIN L3 [get_ports cout]
set_property PACKAGE_PIN P1 [get_ports cin]
set_property IOSTANDARD LVCMOS33 [get_ports cin]
set_property IOSTANDARD LVCMOS33 [get_ports cout]
```

Testbench

```
`timescale 1ns/100ps

module frbit_adder;
        wire [4:0] w;
        reg [3:0] a;
        reg [3:0] b;

        Frb t(w,a,b);
        initial
        begin
                $dumpfile("test_circuit.vcd");
                $dumpvars(0,frbit_adder);
                a=0;b=0;
                for(integer i=0;i<16;i++)
                begin
                        for(integer j=0;j<16;j++)
                        begin
                                #10 b+=1;
                        end
                        a+=1;b=0;
                end
                $monitor("a = %b b = %b and a+b = %b ",a,b,w);

                #10 $finish;

        end
endmodule
```

Simulation :



Reports :

Resources utilized :

- For this code, the FPGA has used 4 **SLICE LUT** where LUT is used as a logic and not memory.
- We can also infer that we have not used any clocking resources or RAM memory.
- As we have not used any clock resources in this experiment, there are no timing constraints.

Time delays

Type of Path	*Path taken*	**Time delay (ns)**
Max Delay path	**cin** $\rightarrow$ **cout**	10.013
	cin $\rightarrow$ **s[3]**	9.988
	cin $\rightarrow$ **s[2]**	9.240
	cin $\rightarrow$ **s[1]**	8.701
	cin $\rightarrow$ **s[0]**	8.623
Min Delay path	**a[3]** $\rightarrow$ **cout**	2.234
	a[3] $\rightarrow$ **s[3]**	2.259
	a[2] $\rightarrow$ **s[2]**	2.311
	a[0] $\rightarrow$ **s[0]**	2.337
	a[1] $\rightarrow$ **s[1]**	2.352

Power consumed

- The power consumed by FPGA is **8.523 W**