MICROPROCESSOR LAB EXPERIMENT 2

Deenabandhan N ee23b021
Sai Harshith Gajendra ee23b069
Krutarth Patel ee23b137

18 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.

Wallace Multiplier

Data flow model

```
module unsigned_mult(output [7:0] m , input [3:0] a,b);
        wire [3:0]p[0:3];
        genvar i,j;
        generate
        for(i=0;i<4;i=i+1)
        begin
                for(j=0;j<4;j=j+1)
                begin
                        assign p[i][j]=a[j]*b[i];
                end
        end
        endgenerate
        wire s0,s1,s2,s3,s4;
        wire c0,c1,c2,c3,c4;

        wire k1,l1,k2,l2;

        assign m[0]=p[0][0];

        half_adder h0(k1,l1,p[3][0],p[2][1]);

        half_adder h1(k2,l2,p[1][3],p[2][2]);

        half_adder h2(s0,c0,p[0][1],p[1][0]);

        FA_S f1(s1,c1,p[2][0],p[1][1],p[0][2]);

        FA_S f2(s2,c2,p[0][3],p[1][2],k1);

        FA_S f3(s3,c3,p[3][1],k2,l1);

        FA_S f4(s4,c4,p[2][3],p[3][2],l2);

        wire u1,u2,u3,u4;

        assign m[1]=s0;

        half_adder h3(m[2],u1,s1,c0);

        FA_S f5(m[3],u2,u1,s2,c1);

        FA_S f6(m[4],u3,u2,s3,c2);

        FA_S f7(m[5],u4,u3,s4,c3);

        FA_S f8(m[6],m[7],u4,p[3][3],c4);
endmodule
```

Schematics :

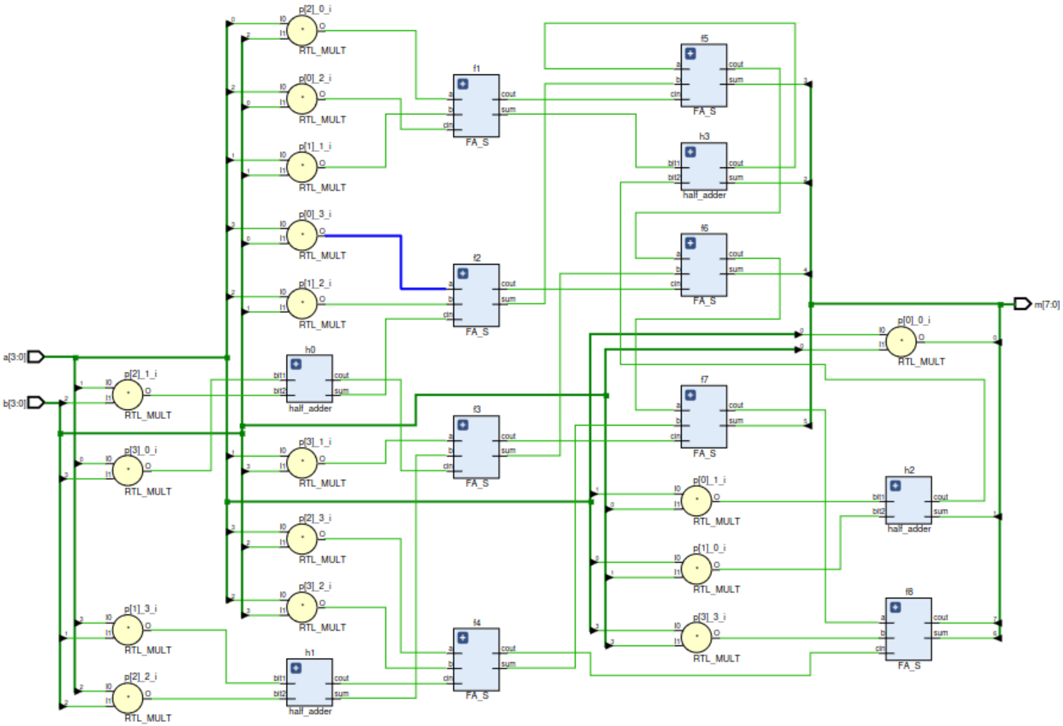

Figure 1: Schematics of the Data flow modelling

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 PACKAGE_PIN L5 [get_ports {a[0]}]
set_property PACKAGE_PIN L4 [get_ports {a[1]}]
set_property PACKAGE_PIN M4 [get_ports {a[2]}]
set_property PACKAGE_PIN M2 [get_ports {a[3]}]
set_property PACKAGE_PIN M1 [get_ports {b[0]}]
set_property PACKAGE_PIN N3 [get_ports {b[1]}]
set_property PACKAGE_PIN N2 [get_ports {b[2]}]
set_property PACKAGE_PIN N1 [get_ports {b[3]}]
set_property PACKAGE_PIN J3 [get_ports {m[0]}]
set_property PACKAGE_PIN H3 [get_ports {m[1]}]
set_property PACKAGE_PIN J1 [get_ports {m[2]}]
set_property PACKAGE_PIN K1 [get_ports {m[3]}]
set_property PACKAGE_PIN L3 [get_ports {m[4]}]
set_property PACKAGE_PIN L2 [get_ports {m[5]}]
set_property PACKAGE_PIN K3 [get_ports {m[6]}]
set_property PACKAGE_PIN K2 [get_ports {m[7]}]
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 {m[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {m[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {m[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {m[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {m[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {m[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {m[1]}]
```

```
set_property IOSTANDARD LVCMOS33 [get_ports {m[0]}]
```

Reports :

Resources utilized :

Ref Name	*Used*	*Functional category*
OBUF	*8*	**IO**
IBUF	*8*	**IO**
LUT2	*2*	**LUT**
LUT3	*2*	**LUT**
LUT4	*3*	**LUT**
LUT5	*7*	**LUT**
LUT6	*6*	**LUT**

Time delays

Type of Path	*Path taken*	**Time delay (ns)**
Max Delay path	**b[6] → m[6]**	**11.016**
	b[3] → m[3]	**10.913**
	b[3] → m[7]	**10.877**
Min Delay path	**a[1] → m[2]**	**2.275**
	b[1] → m[7]	**2.297**
	b[1] → m[6]	**2.332**

Power consumed

- The power consumed by FPGA is **0.325 W**