Field Programmable Gate Arrays & Its Applications

RaktimChakraborty
M.Tech[Part-II]
Department of Computer Science & Engineering
University of Kalyani

1. Introduction

A Field Programmable Gate Array (FPGA) is a reprogrammable chip which contains hundreds of thousands of logic blocks that internally connects together to build complex digital circuitry. It is primarily a semiconductor device that can be configured by the user (customer or designer) after the manufacturing process has been completed. That is why it is also called Field Programmable. The architecture of FPGA which is a collection of interconnected Configurable Logical Blocks (CLB) is depicted in Fig.1. In general, a logic block consists of a few logical cells. The configuration a logical cell is depicted in Fig.2.A typical cell consists of a 4-input LUT (Look-Up Table), a Full adder (FA) and a D-type flip-flop, as shown to the right. The LUTs are in this figure split into two 3-input LUTs. In normal mode those are combined into a 4-input LUT through the left mux.

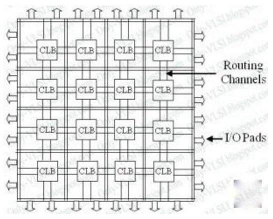


Fig.1

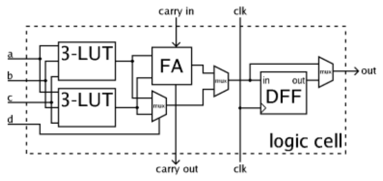
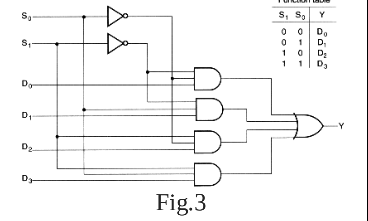


Fig.2

In arithmetic mode, their outputs are fed to the FA. The selection of mode is programmed into the middle multiplexer. The output can be either synchronous or asynchronous, depending on the programming of the mux to the right, in the figure example. In practice, entire or parts of the FA are put as functions into the LUTs in order to save space.
Before FPGAs & programmable logic there exists fixed hardware with their fixed applications. They allow a limited flexibility by adding software support, for example processors. By the early 1980s, Large Scale Integrated Circuits (LSI) widely used to connect the large integrated circuits. These systems typically consisted of few large scale integrated components and large number of Small Scale Integrated Circuits (SSI) & Medium Scale Integrated Circuits (MSI) components.To solve this problem led to development of custom ICs. This was the replacement of large amount interconnections. It reduces the complexity of the circuit & manufacturing cost and provides improved performance. But, the custom ICs have their own disadvantages. The production is very slow of these circuits and the time to come in market is very high as because of increased time to design the circuits. FPGAs are introduced as an alternative of the Custom ICs for implementing entire system in one chip & to provide flexibility of re-programmability to the user/customer/designer.The FPGA industry sprouted from Programmable Read Only Memory (PROM) and Programmable Logic Devices (PLD). PROMs & PLDs both had the options of being programmed in batches in factory or in the field. Rather than Custom IC approach FPGA become very popular as because of its re-programmability, it comes in the market with less time & less amount of inventory cost.
The FPGA configuration is generally specified using a hardware description language (HDL) (ex-Very Hardware Descriptive Language (VHDL), Verilog), similar to that used for an application-specific integrated circuit (ASIC). Here we are going to show an application of FPGA using Xilinx ISE in VHDL.

2. Design of A 4x1 Multiplexer:

The Circuit Diagram of a 4x1 Multiplexer is shown in Fig.3. There are Two Select lines (S_0 and S_1) and four data input lines (D_0-D_3)and a single output line (Y). The truth table of the multiplexer is also shown in Fig.3. From the truth table of the 4x1 multiplexer circuit we can form a logical expression for the output in terms of the data input and after performing the logical OR operation, we get the final expression of the data output.



S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Fig.3

3. Implementation in Xilinx ISE using VHDL

3.1 Behavioral of 4x1 Multiplexer:

```
library IEEE;                          -- Package declaration
use IEEE.STD_LOGIC_1164.ALL;

entity four_input_mux is
    Port ( s0 : in  STD_LOGIC;      -- Input & Output Ports Declaration
           s1 : in  STD_LOGIC;
           d0 : in  STD_LOGIC;
           d1 : in  STD_LOGIC;
           d2 : in  STD_LOGIC;
           d3 : in  STD_LOGIC;
           y : out  STD_LOGIC);
end four_input_mux;

architecture Behavioral of four_input_mux is

begin
-- The Final Expression For the data output
y <= ((d0 and (not s1) and (not s0)) or (d1 and (not s1) and s0) or (d2 and s1 and (not s0)) or (d3 and s1 and s0));
end Behavioral;
```

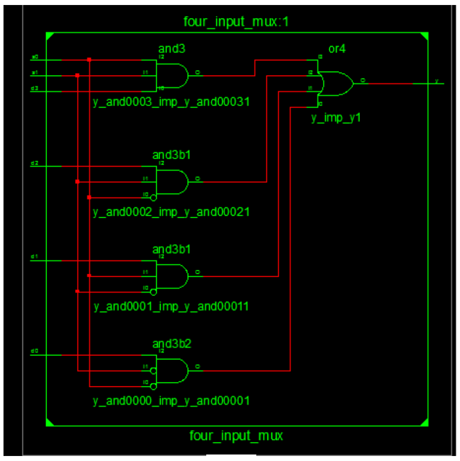


Fig.4

The RTL Schematic Diagram of the 4x1 Multiplexer Circuit is depicted in Fig.4

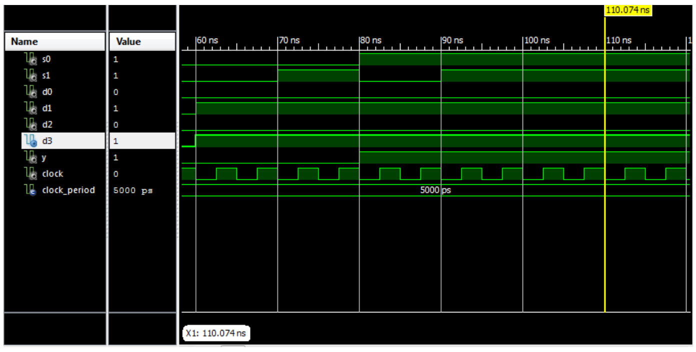
We can analyze this RTL schematic diagram which generated by the behavioral of the circuit by the truth table of that circuit in the test-bench portion. Here, we insert the truth table in the test-bench portion for the testing purpose whether we are getting our desired results or not as per the design and the truth table of the circuit.

3.2 Test Bench of the 4x1 Multiplexer:

```
LIBRARY ieee;               -- Declaration of packages
USE ieee.std_logic_1164.ALL;
ENTITY four_inps_mux IS
END four_inps_mux;

ARCHITECTURE behavior OF four_inps_mux IS
    -- Component Declaration for the Unit Under Test (UUT)
     COMPONENT four_input_mux
    PORT(
         s0 : IN  std_logic;
         s1 : IN  std_logic;

         d0 : IN  std_logic;
         d1 : IN  std_logic;
         d2 : IN  std_logic;
         d3 : IN  std_logic;
         y : OUT  std_logic
        );
    END COMPONENT;
      --Inputs
   signal s0 : std_logic := '0';
   signal s1 : std_logic := '0';
   signal d0 : std_logic := '0';
   signal d1 : std_logic := '0';
   signal d2 : std_logic := '0';
```



Name	Value
s0	1
s1	1
d0	0
d1	1
d2	0
d3	1
y	1
clock	0
clock_period	5000 ps

Fig.5

```
   signal d3 : std_logic := '0';
            --Outputs
   signal y : std_logic;
   constant clock_period : time := 5 ns;
            signal clock : std_logic;

BEGIN

            -- Instantiate the Unit Under Test (UUT)
uut: four_input_mux PORT MAP (
         s0 => s0,
         s1 => s1,
         d0 => d0,
         d1 => d1,
         d2 => d2,
         d3 => d3,
         y => y
       );
   -- Clock process definitions
clock_process :process
   begin
                         clock <= '0';
                         wait for clock_period/2;
                         clock <= '1';
                         wait for clock_period/2;
   end process;
    -- Stimulus process
stim_proc: process
   begin
      -- hold reset state for 100 ns.
      wait for 50 ns;

      wait for clock_period*2;
d0 <= '0';
d1 <= '1';
d2 <= '0';
d3 <= '1';
      -- insert stimulus here
            s0<= '0';
            s1<= '0';
wait for clock_period*2;
            s0<= '0';
            s1<= '1';
wait for clock_period*2;
            s0<= '1';
            s1<= '0';
wait for clock_period*2;
            s0<= '1';
            s1<= '1';
wait for clock_period*2;
      wait;
   end process;
END;
```

The Clock pulse simulation diagram of the circuit as well as the above mentioned code segment is clearly depicted in Fig.5. The snapshot of the clock pulse simulation diagram is taken from 60ns to 120ns for the demonstration.

From the example of a combinational circuit we can say that it is very simple to specify FPGA by a Hardware Descriptive Language (HDL). Due the simplicity the application area of FPGA is still growing. Specific applications of FPGAs include digital signal processing, software-defined radio, ASIC prototyping, medical imaging, computer vision, speech recognition, cryptography, computer hardware emulation, metal detection and a growing range of other areas.Another trend on the usage of FPGAs is hardware acceleration, where one can use the FPGA to accelerate certain parts of an algorithm and share part of the computation between the FPGA and a generic processor.

3.3 FPGA Applications:

Aerospace and Defense, Communications, Missiles & Munitions, Medical Electronics, Cryptography, ASIC Prototyping, Digital Signal Processing (DSP), Encoders, Displays, Switches and Routers, Consumer Electronics, Medical etc.

As FPGA provides many beneficiary applications on the opposite it's also has some drawbacks. As, It takes long compilation, It consumes more power than customized ICs. ASICs are still much faster than the FPGA & it also still can pack more logic into a chip than FPGA.

4. Conclusion

The purpose of considering, analyzing and designing of circuits with the concept of FPGA is to promote its usability, functionality and its flexibility into the fields of research and implementation, so as to increase the scope and application.

5. References:

[1] Engineering Digital Design by Richard F. Tinder.
[2] VHDL Programing by Example by Douglas L. Perry
[3] FPGA Architecture for the Challengefrom toronto.edu.
[4] http://www.altera.com/literature/hb/cyc2/cyc2_cii51002.pdf
[5]Documentation: Stratix IV Devices Altera.com. 2008-06-11.Retrieveds2013-05-01.
[6]http://www.xilinx.com/support/documentation/user_guides/ug070.pdf
[7] Xilinx aims 65-nm FPGAs at DSP applications. EETimes.
[8] Digital Circuits and Design: 4th Edition by S Arivazhagan, S Salivahanan