|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | C | D | **F1** | **F2** |
| 0 | 0 | 0 | 0 | **1** | **0** |
| 0 | 0 | 0 | 1 | **0** | **0** |
| 0 | 0 | 1 | 0 | **1** | **0** |
| 0 | 0 | 1 | 1 | **0** | **1** |
| 0 | 1 | 0 | 0 | **0** | **0** |
| 0 | 1 | 0 | 1 | **0** | **1** |
| 0 | 1 | 1 | 0 | **0** | **1** |
| 0 | 1 | 1 | 1 | **0** | **1** |
| 1 | 0 | 0 | 0 | **0** | **1** |
| 1 | 0 | 0 | 1 | **1** | **0** |
| 1 | 0 | 1 | 0 | **0** | **1** |
| 1 | 0 | 1 | 1 | **1** | **0** |
| 1 | 1 | 0 | 0 | **0** | **0** |
| 1 | 1 | 0 | 1 | **1** | **1** |
| 1 | 1 | 1 | 0 | **0** | **1** |
| 1 | 1 | 1 | 1 | **1** | **0** |

Compact Minterm form:

F1: m(0,2,9,11,13,15)

F2: m(3,5,6,7,8,10,13,14)

Reduced SOP Form:

F1:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AB\CD | 00 | 01 | 11 | 10 |
| 00 | 1 | 0 | 0 | 1 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 1 | 1 | 0 |
| 10 | 0 | 1 | 1 | 0 |

SOP: (A’ \* B’ \* D’) + (A \* D)

F2:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AB\CD | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 1 | 0 |
| 01 | 0 | 1 | 1 | 1 |
| 11 | 0 | 1 | 0 | 1 |
| 10 | 1 | 0 | 0 | 1 |

SOP: (A’ \* C \* D) + (B \* C \* D’) + (B \* C’ \* D) + (A \* B’ \* D’)

Source Code:

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity exp5 is

Port ( A : in STD\_LOGIC;

B : in STD\_LOGIC;

C : in STD\_LOGIC;

D : in STD\_LOGIC;

F1 : out STD\_LOGIC;

F2 : out STD\_LOGIC);

end exp5;

architecture Behavioral of exp5 is

begin

F1 <= (((NOT A) AND (NOT B) AND (NOT D)) OR (A AND D));

F2 <= (((NOT A) AND C AND D) OR (B AND C AND (NOT D)) OR (B AND (NOT C) AND D) OR (A AND (NOT B) AND (NOT D)));

end Behavioral;

Questions:

1) 6 Gates for F1, 10 Gates for F2, assuming 4-input NAND gates work in the same way as the 2-input NAND gates.

2) The Logic Analyzer fully reuiqres the NEXYS2 board’s clock to store data because it uses the board’s clock for the timing waves. If the NEXYS2 board stops giving output, the logic analyzer loses its timing.

3) It can store less with a higher sampling frequency because it stores more data for that one sample, meaning it stores the same amount of info but it is just more detailed for a shorter amount of time.

4) You can use the Time/Div selection in the Timing Waveforms viewer to lengthen or shorten the amount of time between each marker. That way you can have more, less accurate data on the screen at one time or less samplings but with higher detail at one time.

5) Instead of using the Logic Analyzer’s clock like in Procedure 1, We use the NEXYS2 board’s system clock in Procedure 2. If we hooked up our procedure 2 solution to the analyzer, the timing would be a little bit off because of the differences in clocks.

6) Not neccesarily, the timing could just be off between the board and the logic analyzer. There could also be more propogation delays when viewed on the analyzer.

7) It has up to 33,192 logic cells

Conclusions:

Tyler: In this lab I learned how to use the logic analyzer to look at how the FPGA on the NEXYS board was programmed. I then made a truth table based on the inputs and outputs, and looked at the results. From there I was able to create Minterm lists for each of the two outputs and then make SOPs from them. I then used the Xilinx tools to re-create the circuitry that was in the original .bit file for experiment 5. After using multiple parts of the Xilinx tools package, I was finally able to correctly re-create the original logic.

Susan: During this lab, I familiarized myself with the basic functions of the the Logic Analyzer. We were able to use it to take in 4 inputs and display all the possible outputs for the 2 functions we were given. Using this we were able to write SOPs for the 2 functions and then write VHDL to duplcate the output. This was a lot cleaner and simpler than physically creating the circuits as in expertiment 3. I happy to use it in the future to make creating complex circuits more easily.