Christopher Badolato

Random Access Memory

EEE 3342 LAB 0014

11/11/2018

**Objective:**

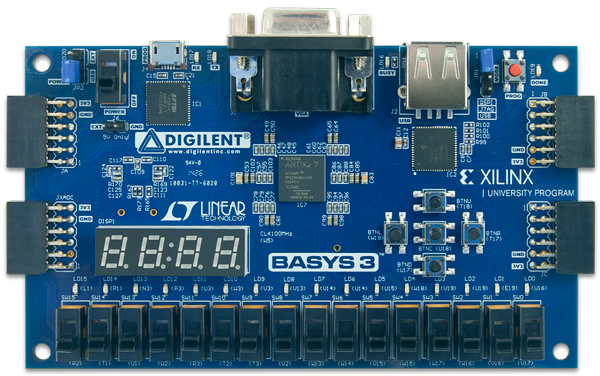
* To examine the use of RAM (and ROM) as means of realizing combinational logic circuits.

Apparatus List:

Xilinix’s FPGA VIVADO HLX edition

VIVDAO 2017.4

Basby Atrix-7



Procedure and/or Design Methodology:

F1(w,x,y,z) = wy'z + wxyz + w'x'y + wx'z '+ w'x'y'z'

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| W | X | Y | Z | Output |
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

**Full Adder – F2**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| W | X | Y | Z | Cout | S1 | S0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 |

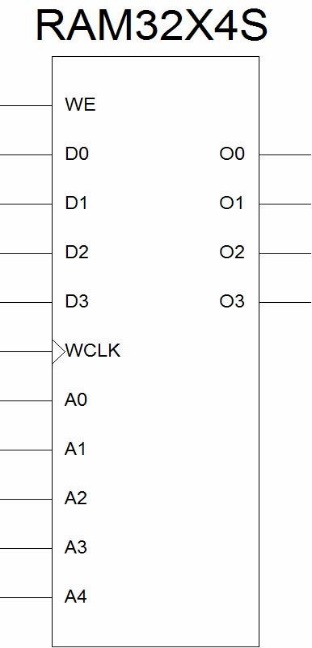
Adress data (what we want to store at the selected switches)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address[4] | Address[3] | Address[2] | Address[1] | Address[0] | Data\_in[3] | Data\_in[2] | Data\_in[1] | Data\_in[0] |
| Off | Off | Off | Off | Off | 0 | 0 | 0 | 1 |
| Off | Off | Off | Off | On | 0 | 0 | 1 | 0 |
| Off | Off | Off | On | Off | 0 | 1 | 0 | 0 |
| Off | Off | Off | On | On | 0 | 1 | 1 | 0 |
| Off | Off | On | Off | Off | 0 | 0 | 1 | 0 |
| Off | Off | On | Off | On | 0 | 1 | 0 | 0 |
| Off | Off | On | On | Off | 0 | 1 | 1 | 0 |
| Off | Off | On | On | On | 1 | 0 | 0 | 0 |
| Off | On | Off | Off | Off | 0 | 1 | 0 | 1 |
| Off | On | Off | Off | On | 0 | 1 | 1 | 1 |
| Off | On | Off | On | Off | 1 | 0 | 0 | 1 |
| Off | On | Off | On | On | 1 | 0 | 1 | 0 |
| Off | On | On | Off | Off | 0 | 1 | 1 | 0 |
| Off | On | On | Off | On | 1 | 0 | 0 | 1 |
| Off | On | On | On | Off | 1 | 0 | 1 | 0 |
| Off | On | On | On | On | 1 | 1 | 0 | 1 |

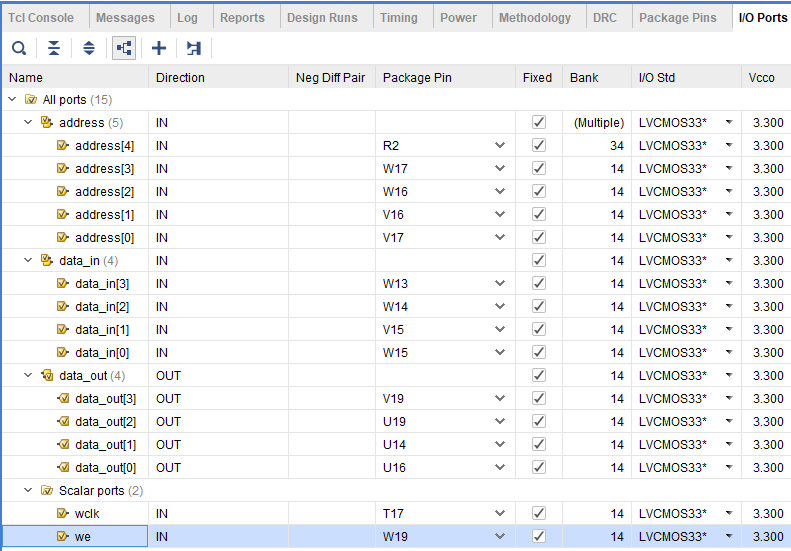
F1 and F2

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| W | X | Y | Z | Cout(data\_out[3]) | S1(data\_out[2]) | S0(data\_out[1]) | Output(data\_out[0]) |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |

Block Diagram:



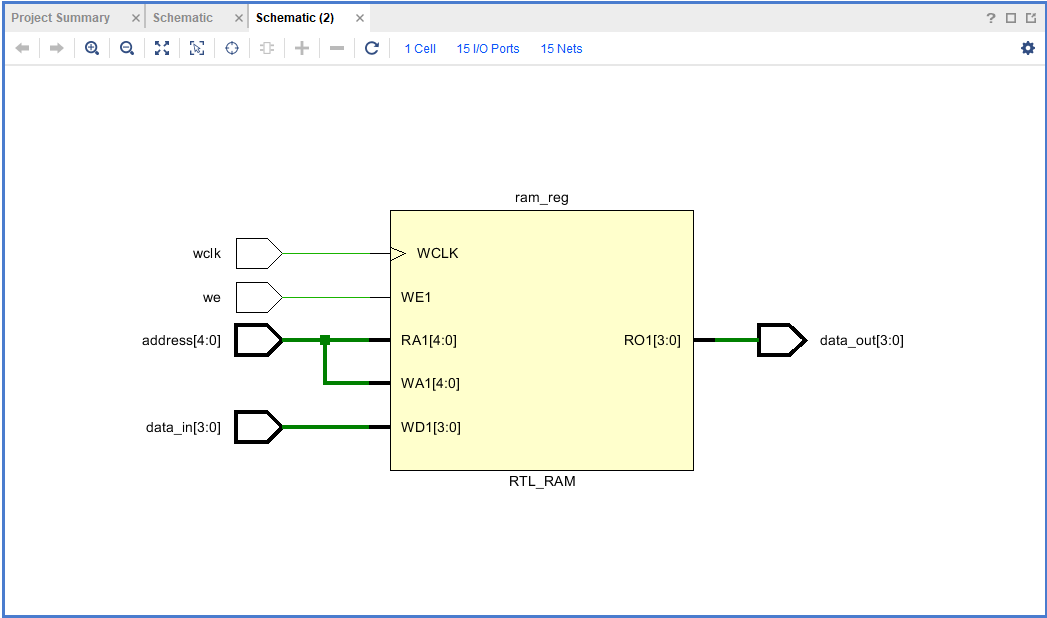
Design Specification Plan:



S\_Ram.v

|  |  |
| --- | --- |
| `timescale 1ns / 1ps  module S\_Ram(wclk, we, address, data\_in, data\_out);  input wclk;  input we;  input [4:0] address;  input [3:0] data\_in;  output [3:0] data\_out;    reg [3:0] ram [0:32];    always@(posedge wclk)  begin  if(we)  ram[address] <= data\_in;  end  assign data\_out = ram[address];  endmodule |  |

Detailed Schematic Diagram:



Test Plan:

After the pins are set, and the code is uploaded we can write to our bit file and start testing our board. We can follow our Address and data\_in (or f2 and f1) truth tables, to store each value of our data\_in at the corresponding address selected by the switches. We want to store by first clicking the W19 WE button while holding this button we then click T17 button representing the clock, and on the post edge of the click the values will be store on the board at the “switch selected” address .

Results:

The results are each value stored at the corresponding address,we can show what is store at each address just by flipping the switches associated with them, but only after we’ve gone through and stored them

We can also overwrite values at the address by repeating the process used to enter each value.

Conclusion

Our Ram will store values at the corresponding address following our truth table.

**Questions:**

1. Discuss the advantages of using RAM for implementing combinational functions.

Seeing as ram is read and write memory it is more flexible than combination gate arrays. We do not have to worry about if there is a large function and we only need a few cases, we can find the minterms associated with the function and a small ram would be a more efficient way than programming the gates.

1. Which one is more practical, read/write memory or read only memory? Explain.

A read/write memory is more practical depending on the situation. In the case we want our memory to never be overwritten then using a ROM may be a better choice.

1. Some memory devices have only one set of data lines for both input and output. What is the advantage and disadvantage of having the input and output lines separately or shared?

The advantage of having the input and output line separate would obviously be lower cost, though we may need to send information both directions through the line.

4. Have you met all the requirements of this lab (Design Specification Plan)?

5. What is the advantage of using RAM or ROM memory to implement combinational logic?

Ram is read and write. We again would not need to set up the function. Just find the minterms and use a small ram to program the gates.

6. How can two 32x4s devices be combined together to form one 64 by 4 bit memory system? (Hint: 8 – 2 to 1 multiplexers can be used in the design).

