****

**A REPORT TO ASSESS THE IMPLEMENTATION OF TWO’S COMPLEMENT CHECKSUM FOR MEMORY**

**Hassam Abdullah**

**Department of Computer Engineering**

**Bilkent University**

**Digital Design 223-02**

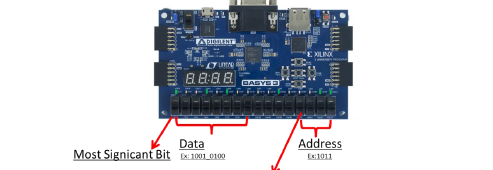
**May 18, 2020**

**Introduction: -**

A Basys-3 board was used to implement the two’s complement checksum for memory. Its buttons were assigned to several operations outlined below and its switches were assigned to input the data and address variables, respectively.

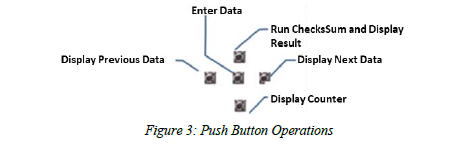
**Physical Modules used: -**

As shown in figure.1, below, switches were used to input the data and address variables. The right most switches were assigned to contain the address of the data and the left most switches were assigned to contain the data itself. With the most significant bits, starting from the left as shown in the figure below.



**Figure.1**

As shown in figure. 2, the buttons were assigned to specific operations, that will be described in detail below. The up button, left button, center button, right button and down button were assigned to “Run Checksum and Display Result”, “Display previous data”, “Enter data”, “Display Next Data” and “Display Counter” respectively.



The enter data operation enters the data when pressed and the address pertaining to it. The Run Checksum and Display Result operation converts the data into checksum and displays the result on the seven segment display. The display next and previous data operations display the next data and previous data in the list, respectively. While the display counter displays the total number of clock cycles during CheckSum operation.

**Modules Used:**

**Ten\_sec: -** This module is used to affect the clock cycle of the top module every ten seconds so the seven segment displays the results every ten seconds.

**Memory: -** The memory module assigns the data and address variables to arrays by taking 8 bit inputs from data switches and 4 bit data from address switches assigned as shown in figure.1.

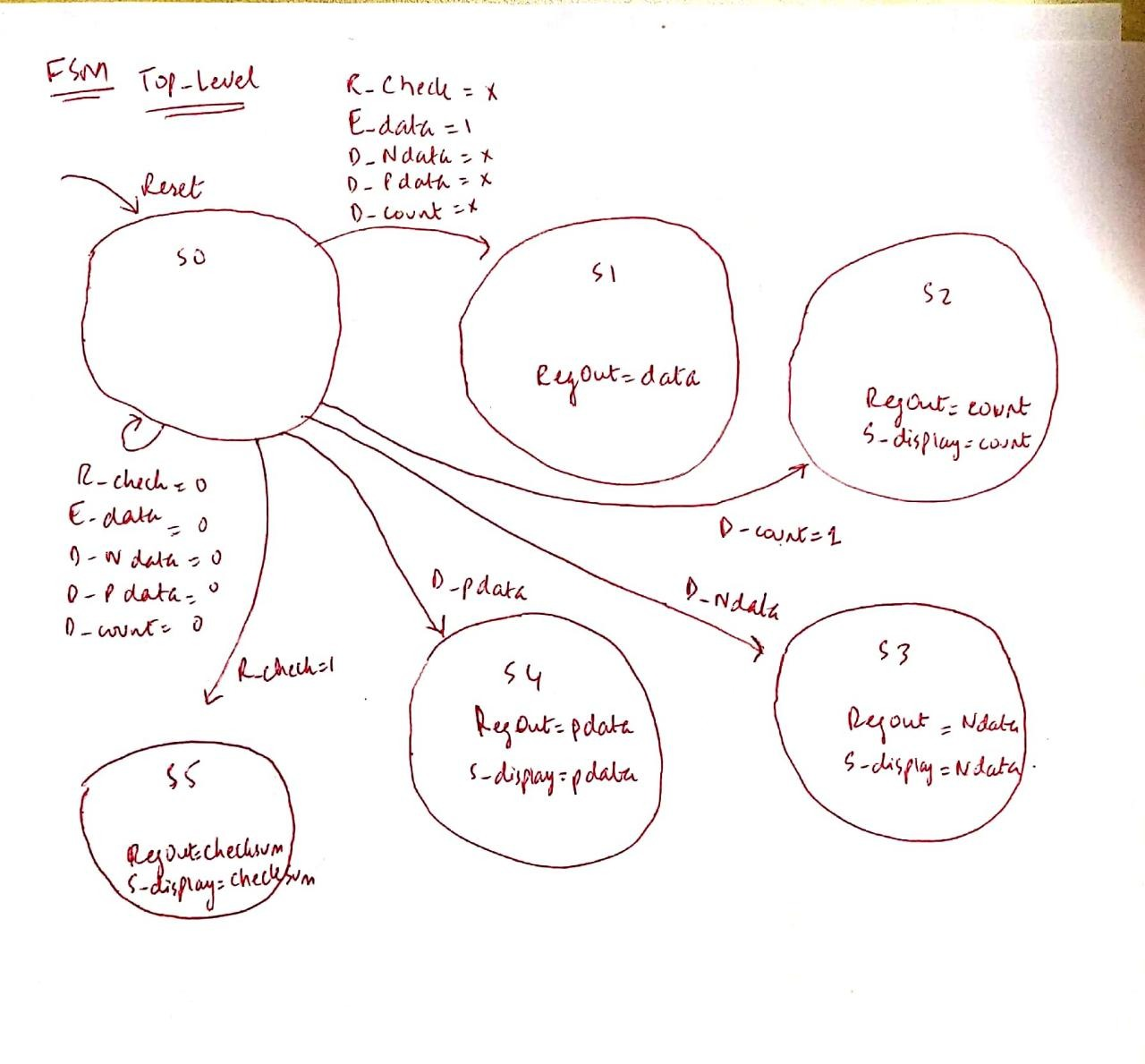
**Seven\_segment: -** The seven segment module is used to alter the seven segment display and show the appropriate result depending on the operation/button pressed.

**Checksum: -** The checksum module calculates the checksum of data entered via the switches. It takes the 8 bit input and computers the checksum of it by inverting the bits and adding one. It outputs the clock counter value and the checksum value as a result.

**Debouncer:-** The debouncer module is used to take care of instances where the user might press the button and due to pressure may trigger unwanted instances of posedge and negedge.

**Top\_level:-** The top level module integrates all the above mentioned modules and depending on the button pressed, it will perform the corresponding operation on each clock cycle.

**FSM: -** The Finite state machine below represents the inner working of the top-level module.



There were problems in implementing the debouncer module to work properly and the coding of the seven\_segment display module was exteremely challenging since depending on the state the segment had to display different results including equals and dash sign, depending on the state.