**COL 215 – Mini Project**

Yash Malviya 2016CS50403

Hardik Khichi 2016CS50404

16 bit CRC Computation

Note: The problem statement for mini project was changed after submission of lab report 3. In accordance to that we had to make some changes to our implementation starting from the scratch. So, many points in this report may not match with the previously submitted reports.

1. Specifications:

The user will enter n (=10) 16-bit binary numbers using the switches on the board. One input is registered after the user presses button no.1. These numbers will be stored one by one in a B-ram cell that can store 10 16-bit vectors.

If the user presses button no.3, a number form B-ram cell will be input to the CRC calculation entity, which will further display the hex value of the calculated checksum.

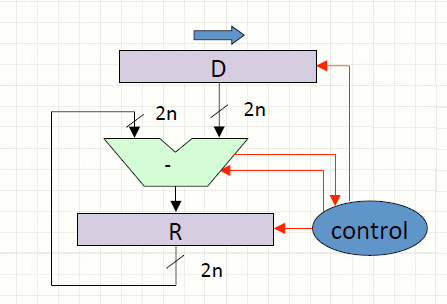
If the user presses button no.4, a new number will be input to the CRC calculation entity but its 7th-bit will be inverted. Similarly, if user presses button no.5, the number’s 12th-bit will be inverted. Finally, its checksum is displayed.

If the user presses button no.2 (RESET Button), all the numbers in the B-ram cell will be removed. The SSD will start displaying the seed value (i.e. ffff).

1. Overall approach:

We will make a finite state machine similar to divider circuit when load button is pressed 1111111111111111 (initial value) and 0000000000000000 (augmented value) prefixed and suffixed to 16bit message respectively. Then this value is divided by pre-decided CRC polynomial using division algorithm but because quotient is not used it is not even stored. When calculation is complete a done signal is turned on. When reset is pressed 0xFFFF is displayed

1. Block diagram:



1. Test and demonstration plan:

We will run random test case manually on the board and check their corresponding CRC with online CRC calculator

COL215 Mini-Project Report-3

CRC-CCITT Computation

Yash Malviya 2016CS50403

Hardik Khichi 2016CS50404

**Detailed description of the overall design**

The BRAM unit stores new inputs every time push button 1 button is pressed. Push button 2 is used for reseting. Calculation is triggered when either of push button 3, push button 4, push button 5 is pressed. Push button 4 & push button 5 modify input before calculation to showcase CRC of erroneous input. The CRC calculation is done in the same way as defined previously. Addresses of input present in BRAM cell corresponding to the calculation of push button 3, push button 4 and push button 5 are stored in separate registers and will be displayed when calculation is done. After FSM sends done signal CRC is displayed on the SSD.

**Description of the major subsystems/components and interconnecting signals**

Since the problem statement changed signals added

**BRAM component related signals**

din – input to write in BRAM

dout – output from BRAM

enable – enable signal for BRAM

reset – reset register in BRAM to 0

wen - “1111” to use BRAM in write mode and “0000” for read

**FSM unit**

pb1 = To load the 16-bit inputs in a BRAM (for n = 10).

pb2 = To reset the CRC computation and reset the checksum to the seed value.

pb3 = To start the CRC computation with no error introduced.

pb4 = To start the CRC computation with error introduced in the 5th word at

7th bit.

pb5 = To start the CRC computation with error introduced in the 3th word at

12th bit.

counter = 0 to 31 to count xor operations

done = 0 or 1 indicating computation done or not

polynomial = crc polynomial

state = 1 or 2 or 3

err1 = input with error of type 1

err2 = input with error of type 1

**Status of coding**

After the modification in problem statement we are using BRAM with our previously written code which was for 1 input. Now BRAM stores multiple inputs for which CRC is calculated

**Status of testing**

We haven’t tested FSM combined with BRAM unit yet as something are yet unclear regarding new problem statement.