Report of CS220 Assignment 4

Anushka Panda Harshit Raj Shubhan R

March 6, 2022

1 About

- About Assignment

  Name: Assignment 4
  Due on: March 6, 2022

- Team

Name: Anushka Panda
Email: apanda20@iitk.ac.in
Roll: 200174

Name: Harshit Raj
Email: harshitr20@iitk.ac.in
Roll: 200433

Name: Shubhan R
Email: shubhanr20@iitk.ac.in
Roll: 200971

- About Course

  Course Code: CS220
  Name of course: Computer Organization
  Instructor: Dr. Urbi Chatterjee
  Semester: 2021-22-II

2 8-bit Adder and Subtracter circuit

2.1 Question

Write a detailed description of eight-bit adder/subtracter to add/subtract two eight-bit two's complement numbers. and it's working with the proper circuit diagram in a PDF file. Then write the Verilog code module to implement an eight-bit adder/subtracter. It will be implemented in two modules. First, module implements a one-bit adder/subtracter with four inputs a, b, cin, and opcode, and two outputs sum and carry. For the addition operation the input opcode will be 0 and 1 for subtraction operation. The top module implements the eight-bit adder/subtracter using the one-bit adder/subtracter module. There will be two inputs for this module the two input numbers and the opcode and produces the sum and whether there is an overflow as the outputs. Now, add a test bench to test the eight-bit adder/subtracter. Your test bench must have fifteen different inputs. Put five-time unit delay between consecutive inputs.

2.2 Basic Definition

The eight bit adder/subtracter circuit takes is a special circuit that takes in two eight-bit numbers, and performs and outputs their addition/subtraction with correct overflow reading, based on the operation inputted (0 for addition, 1 for subtraction).

2.3 The two's complement

For an N-bit number, the two's complement is the complement of a number with respect to 2^N, i.e., the sum of a number and its two's complement is 2^N. So, the two's complement of a number is its subtraction from 2^N. It can also be called the one's complement of a number $+1$.

The two's complement is a signed representation in binary; the two's complement of a number other than 0 is considered as the negative of that number. As a result, we get one extra negative number, that being -2^{N-1}.

According to the above convention, the MSB is used to determine the sign of the number, with 1 being a negative number and 0 being a positive number.

2.4 Our approach to the circuit

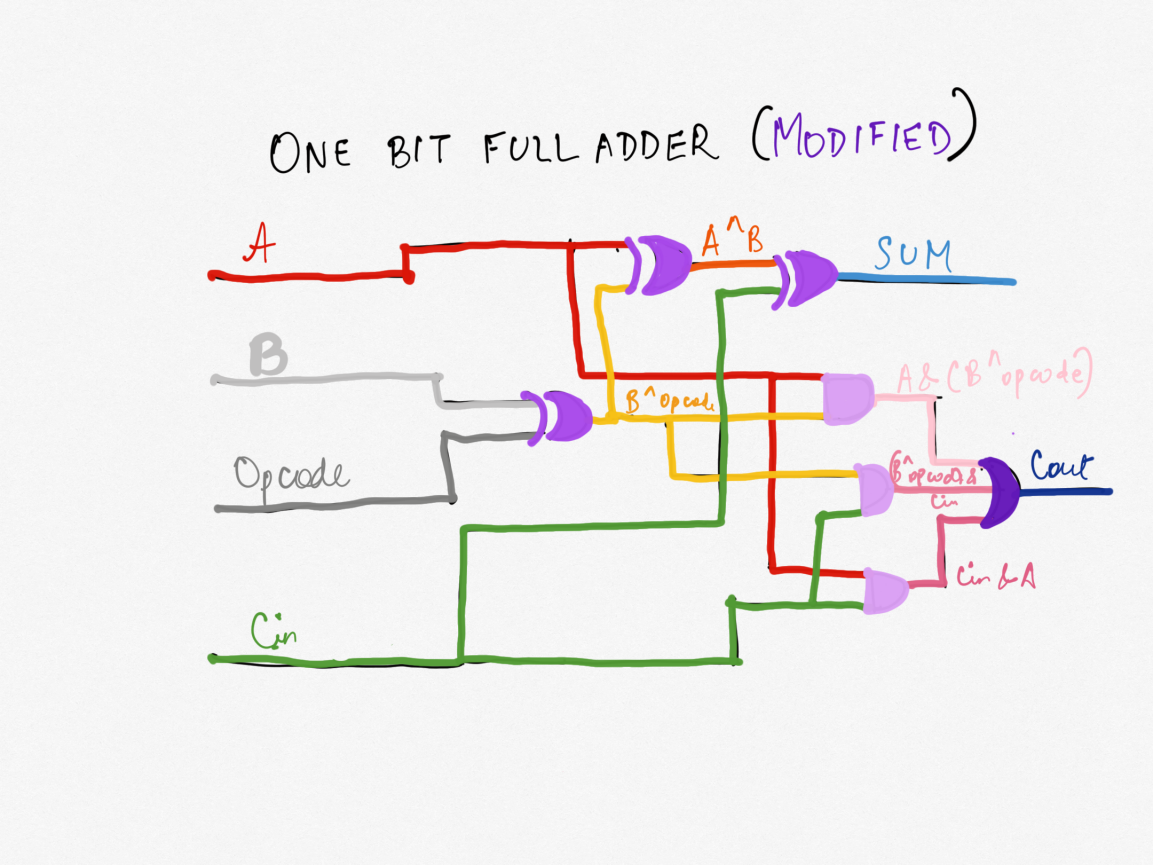
We have taken the inputs to be according to the above two's complement convention. The opcode shall define whether addition or subtraction should be carried out. In case subtraction

is carried out, we convert the second input to it's two's complement and then carry out eight bit addition the way a full adder does. Please note that:

- The above mechanism carries out subtraction with the assumption that the second input is subtracted from the first.
- Our circuit is designed in such a way that it automatically converts the second input to it's two's complement, when the opcode is 1 (subtraction).

2.5 Circuit Diagram, Design, and Working

2.5.1 The one-bit adder



The one-bit full adder has no change as such in the structure. Two input bits are taken along with the carry-in, their XOR giving the sum, and the OR of the pairwise AND of the three, giving the value of carry-out.
The only change is that for the second input B, we take it's XOR with the opcode, since it helps in converting B to it's one's complement.

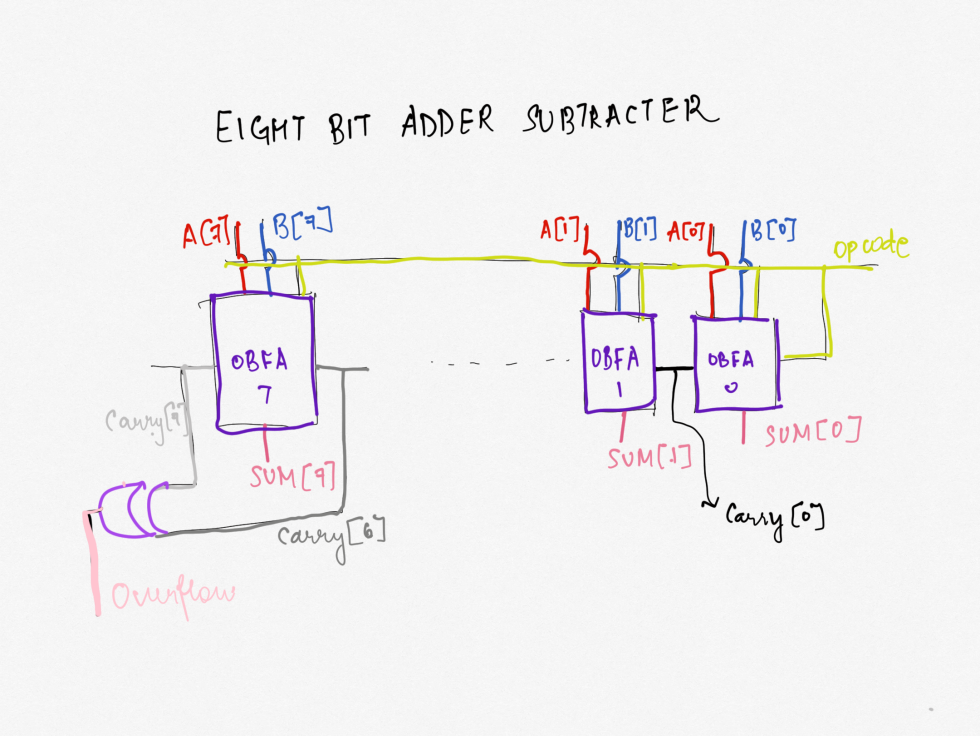
- **Case I: Opcode** $= 1$**:**

- If the value of B is 1, then the XOR gives the output 0, which is correct, since Opcode $= 1$ means B must be converted to the two's complement. This step only converts it to one's complement, and 0's complement is 1.
- If $B = 0$, then the XOR gives 1, which is correct, since 1's complement is 0.

- **Case II: Opcode $= 0$:**
  - If the value of B is 1, then the XOR gives the output 1, which is correct, since Opcode $= 0$ means B must remain as it is, i.e. $B = 1$.
  - If $B = 0$, then the XOR gives 0, which is correct, since Opcode $= 0$ means B must remain as it is, i.e. $B = 0$.

2.5.2 The eight-bit adder-subtracter

For the eight-bit adder-subtracter, we take opcode as the first adder's carry-in for the simple reason that the sum of the one's complement of a number and 1 gives its two's complement. Our one-bit adder already converts the second input to its one's complement, hence adding opcode $= 1$ as carry-in not only gives the command to subtract, but also efficiently carries out the two's complement conversion, basically killing two birds with a stone.



We take eight one-bit adders as described above, take opcode as the first adder's carry-in, and take it's carry-out as the second adder's carry-in and so on.

For overflow, we take the XOR of the last and second last carry, since that is the overflow condition, that the last and second last carry shouldn't be equal.

3 3-bit Gray code counter

3.1 Question

Construct a finite state machine for the 3-bit Gray code counter. The counter is to be designed with one input terminal (which receives pulse signals) and one output terminal. It should be capable of counting in the Gray system up to 7 and producing an output pulse for every 8 input pulses. After the count 7 is reached, the next pulse will reset the counter to its initial state, i.e., to a count of zero.

The state transitions will be: S0: 000, S1:001, S2:011, S3:010, S4:110, S5:111, S6:101, S7:100 $\rightarrow$ S0

Output will be high only from S7 $\rightarrow$ S0.

1. Construct the state assignment, the state diagram, state table, transition and output table, excitation table.
2. Then build the K-map for the same and design the circuitry or logic diagram.
3. Write the Verilog code module to implement the 3-bit Gray code counter.
4. Now, add a test bench to test the 3-bit Gray code counter.

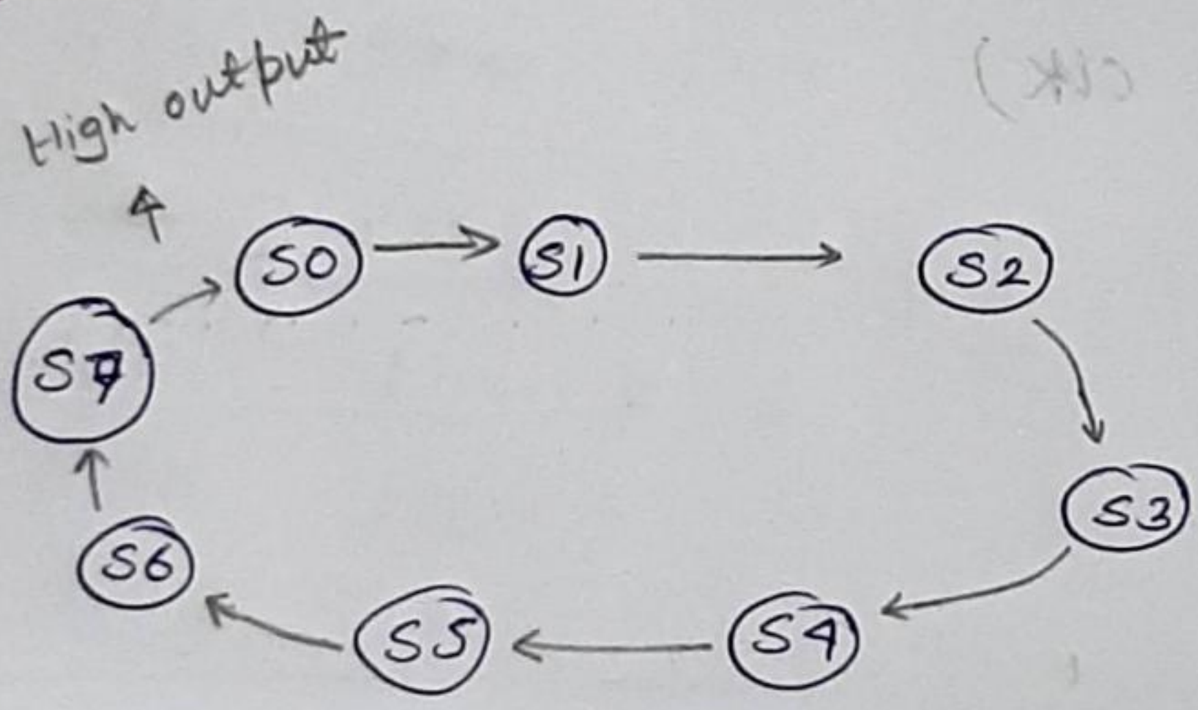
3.2 Solution

Assignment 4 / Q1.

State assignment:

S0:	000	S5	111
S1	001	S6	101
S2	011	S7	100
S3	010		
S4	110		

State diagram:



State table:

On positive impulse

State	Q_2	Q_1	Q_0	Q_2^*	Q_1^*	Q_0^*	Next State
S0	0	0	0	0	0	1	S1
S1	0	0	1	0	1	1	S2
S2	0	1	1	0	1	0	S3
S3	0	1	0	1	1	0	S4
S4	1	1	0	1	1	1	S5
S5	1	1	1	1	0	1	S6
S6	1	0	1	1	0	0	S7
S7	1	0	0	0	0	0	S0

Transition & output table

on positive impulse:

Current state	Next state	output
S0	S1	0
S1	S2	0
S2	S3	0
S3	S4	0
S4	S5	0
S5	S6	0
S6	S7	0
S7	S0	1

Exitation table:

Current state	Input	Next state
S0	0	S0
S0	1	S1
S1	0	S1
S1	1	S2
S2	0	S2
S2	1	S3
S3	0	S3
S3	1	S4
S4	0	S4
S4	1	S5
S5	0	S5
S5	1	S6
S6	0	S6
S6	1	S7
S7	0	S7
S7	1	S0

K- map

⊙ K map for Q_0^* (Next state of Q_0)

Q_2 \ Q_1Q_0	00	01	11	10
0	1	1	0	0
1	0	0	1	1

$$\left[Q_0^* = \bar{Q}_2\bar{Q}_1 + Q_2Q_1 = \text{NOT}(\text{XOR}(Q_1, Q_2))\right]$$

⊙ K map for Q_1^* (Next state of Q_1)

Q_2 \ Q_1Q_0	00	01	11	10
0	0	1	1	1
1	0	0	0	1

$$\left[Q_1^* = \bar{Q}_2Q_0 + Q_1\bar{Q}_0 = \text{XOR}(Q_0, Q_2)\right]$$

⊙ K map for Q_2^* (Next state of Q_2)

Q_2 \ Q_1Q_0	00	01	11	10
0	0	0	0	1
1	0	1	1	1

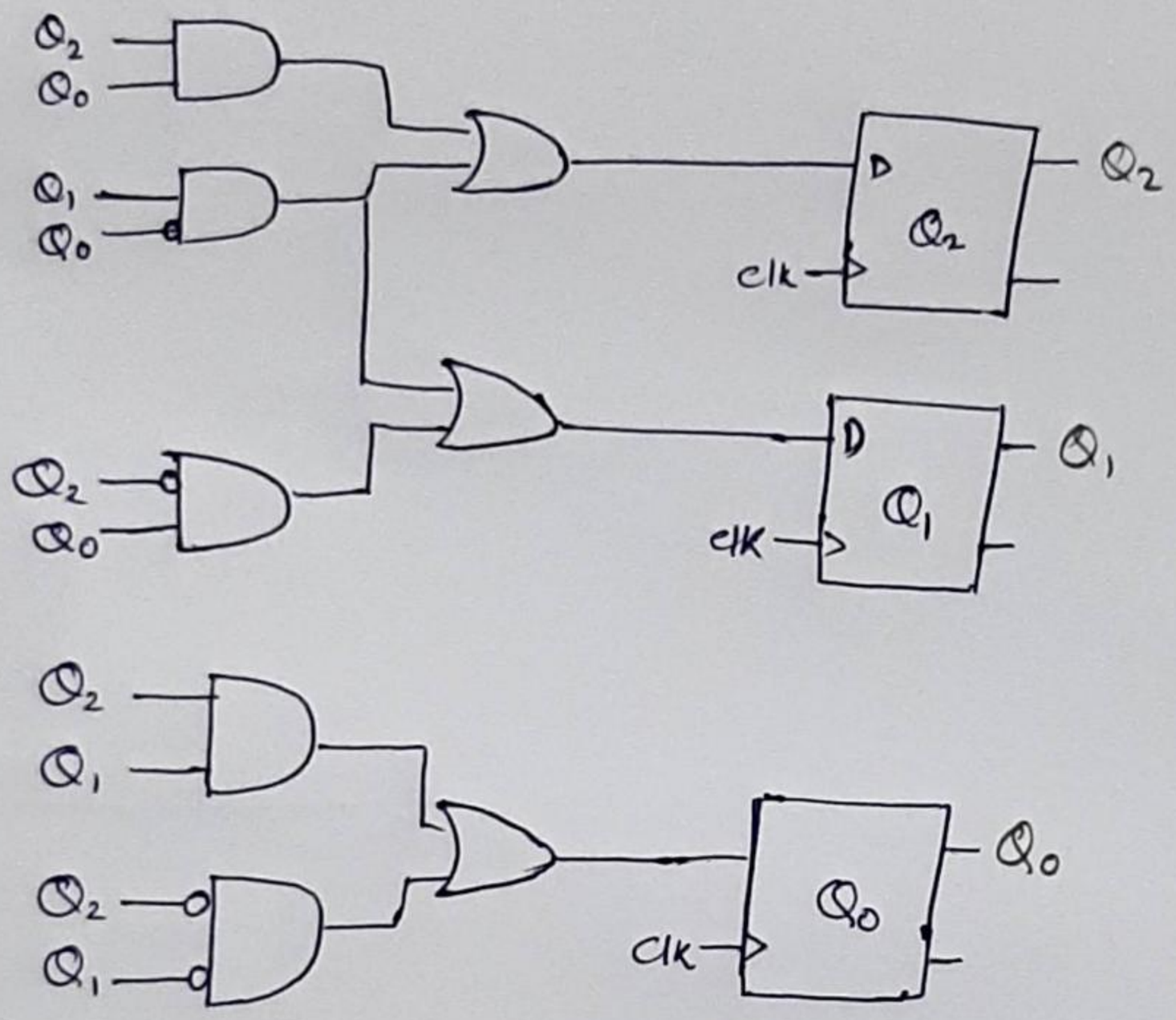
$$\left[Q_2^* = Q_2Q_0 + Q_1\bar{Q}_0\right]$$

- K map for output:

Q_2 \ Q_1Q_0	00	01	11	10
0	0	0	0	0
1	1	0	0	0

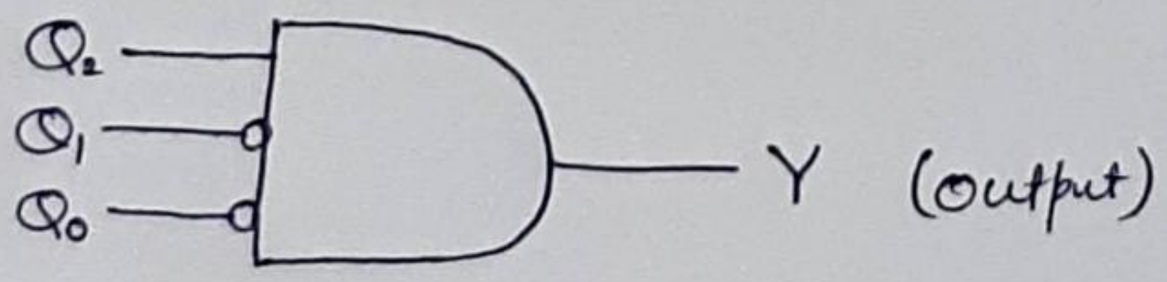
$$Y = Q_2\bar{Q}_1\bar{Q}_0$$

logic diagram:



gray code

when triggred. (through pulse)



output of ~~eow~~ pulse counter