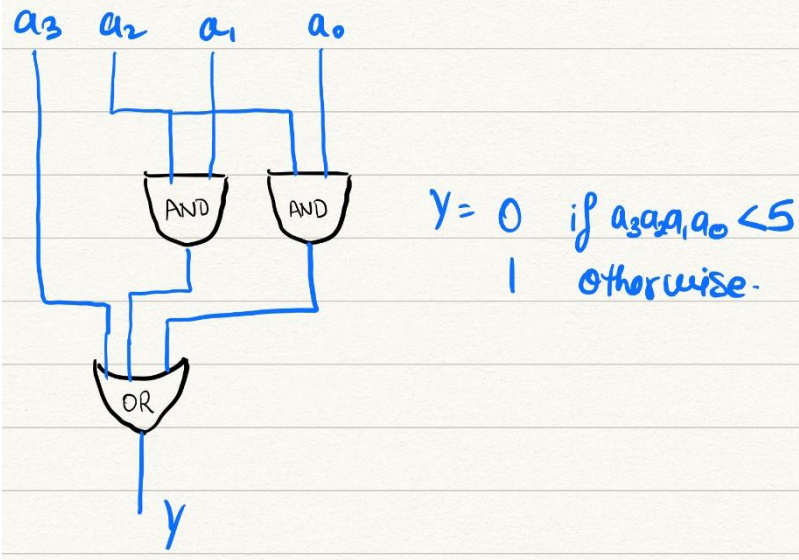
EXPERIMENT O2

COMBINATIONAL DOUBLE – DABBLE

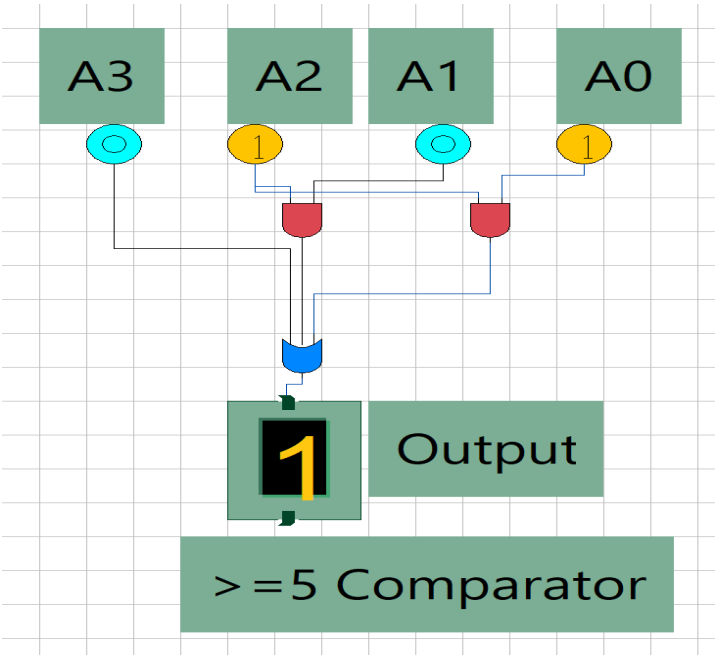
≥ 5 COMPARATOR



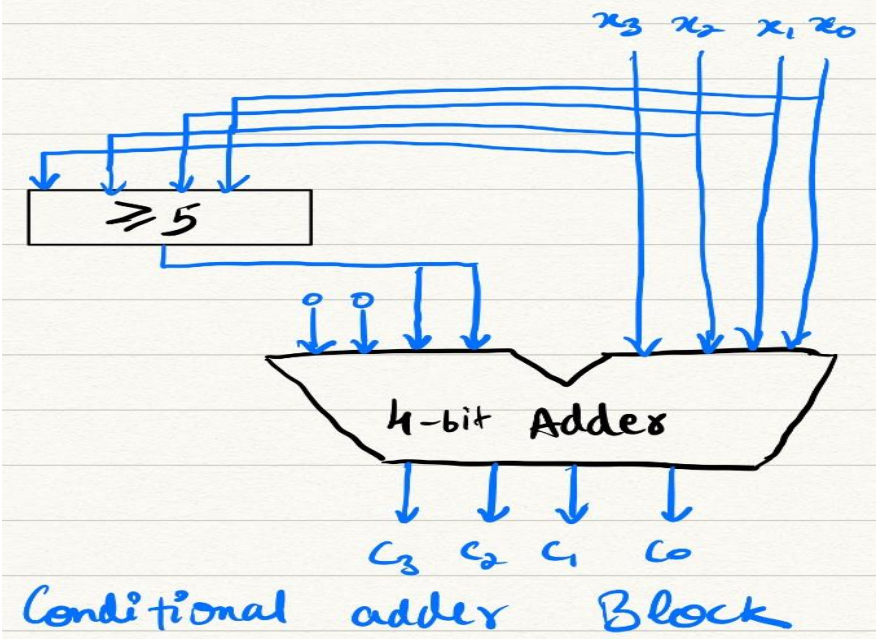
The circuit above is a *comparator* that outputs 1 if the decimal value of the 4-bit input binary number is greater than or equal to 5 and 0 otherwise. The following table checks the accuracy of the circuit by exhaustively calculating the output for each possible 4-bit binary input.

a_3	a_2	a_1	a_0	$b = a_2 \,.\, a_1$	$c = a_2 \,.\, a_0$	$b + c + a_3 = Y$
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	0	1	0	1	1
0	1	1	0	1	0	1
0	1	1	1	1	1	1
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	0	1

1	0	1	1	0	0	1
1	1	0	0	0	0	1
1	1	0	1	0	1	1
1	1	1	0	1	0	1
1	1	1	1	1	1	1

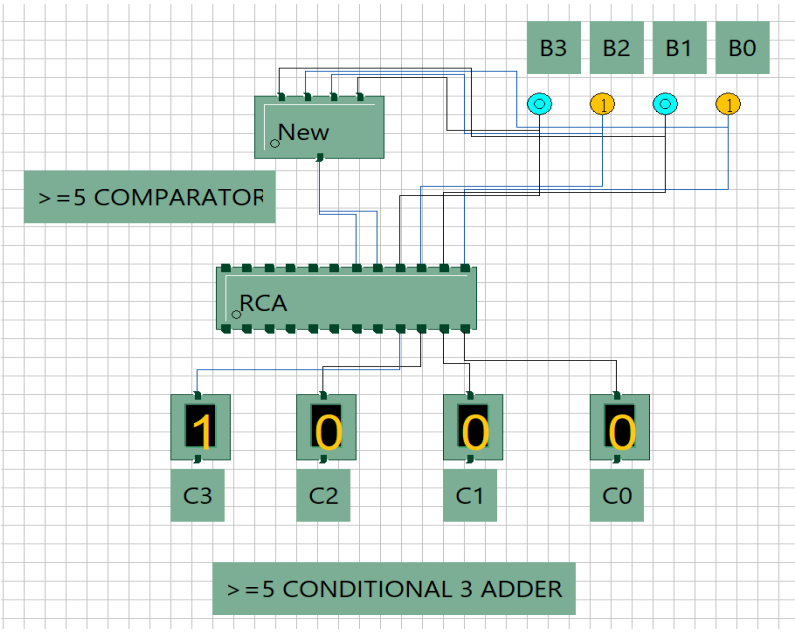


CONDITIONAL ADDER BLOCK



The *comparator* discussed in the last section is used to construct the above *conditional adder block*. It adds 3_{10} or $(0011)_2$ to the 4-bit binary input if its decimal value is greater than or equal to 5. Another characteristic of this circuit, that can be considered as a shortcoming, is that it performs the 4-bit addition operation even if the 4-bit binary input is less than 5, but only this time it is added with $(0000)_2$. Therefore, the circuit is unable to save the computation cost when it is certain that the output is same as the input.

This *≥5 conditional 3 adder block* is needed to construct the binary to BCD converters. It particularly performs the *add-3* operation.



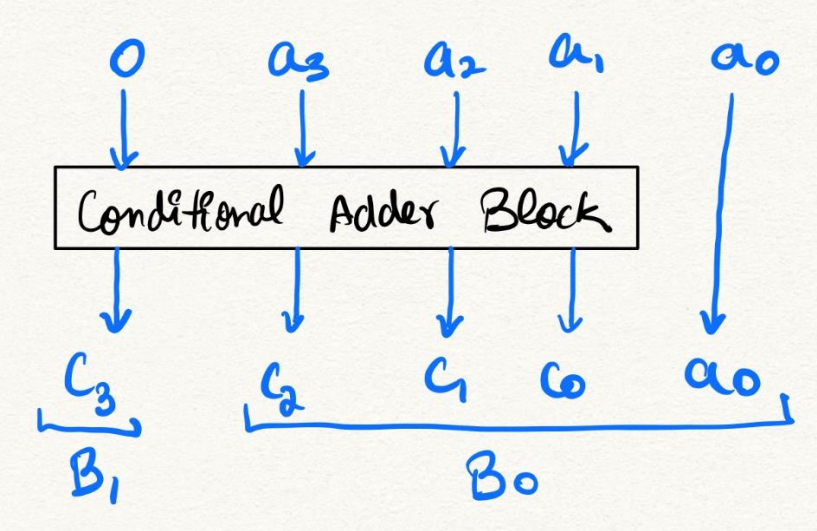
4-BIT BINARY TO BCD CONVERTOR

Step No.	Operation	B_1	B_0	
01	INITIALIZATION	0000	0000	$a_3a_2a_1a_0$
02	L-SHIFT	0000	$000a_3$	$a_3a_2a_1a_0$
03	L-SHIFT	0000	$00a_3a_2$	$a_3a_2a_1a_0$
04	L-SHIFT	0000	$0a_3a_2a_1$	$a_3a_2a_1a_0$
05	ADD-3	0000	$b_3b_2b_1b_0$	$a_3a_2a_1a_0$
06	L-SHIFT	$000b_3$	$b_2b_1b_0a_0$	$a_3a_2a_1a_0$

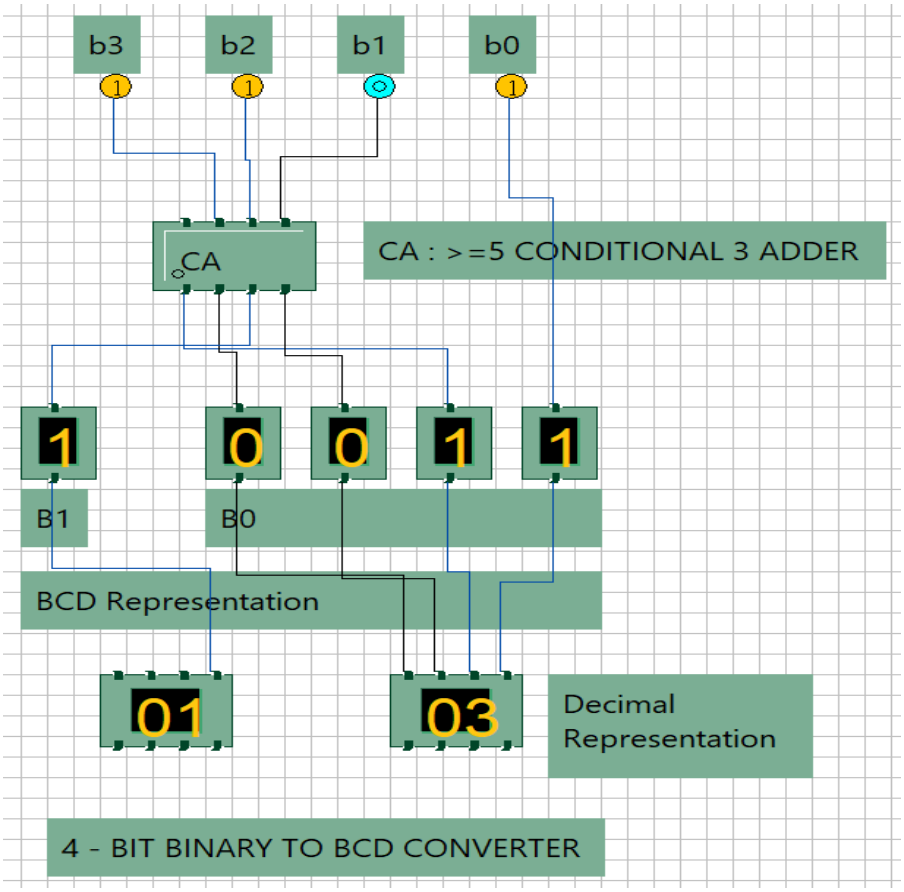
Consider the above implementation of *double dabble algorithm* on a general 4-bit binary vector. The following observations can be made based on the above implementation that might help in figuring out the circuital implementation for this algorithm.

- The three most significant bits in the binary number are *left-shifted* without any *add-3* operation because the decimal value of the zeroth nibble would certainly be less than 5.
- *Add-3* operation is performed at most once and that too only on the zeroth nibble. No adjustment or *add-3* operation is ever performed on the first nibble.

The first observation hints that the first three *left-shifts* can be done consecutively without any checking for $(B_0)_{10} < 5$ post each shift. The second observation hints on the need of only one *conditional adder block* in the circuit. In the circuital implementation below, by the virtue of the first observation we can directly start from the 4[th] step. Therefore, the first step in the circuital implementation would be to input $0a_3a_2a_1$ into the *conditional adder* and get the output $b_3b_2b_1b_0$. The least significant bit a_0 is then simply concatenated to this binary output vector to get the final BCD conversion.



Consider the above circuit diagram. Let $a_3a_2a_1a_0$ be the 4-bit binary vector that is to be converted to its corresponding *binary coded decimal* representation. The *conditional adder block* above consists of a *comparator* that compares the decimal value of the 4-bit binary input with 5, and returns 0 if its less than 5 and 1 otherwise. The *conditional adder block* decides to add 3_{10} to the 4-bit binary input if and only if the comparator embedded in it returns 1 for that input. The circuit is the exact reflection of the iterative algorithm implemented earlier.



6-BIT BINARY TO BCD CONVERTOR

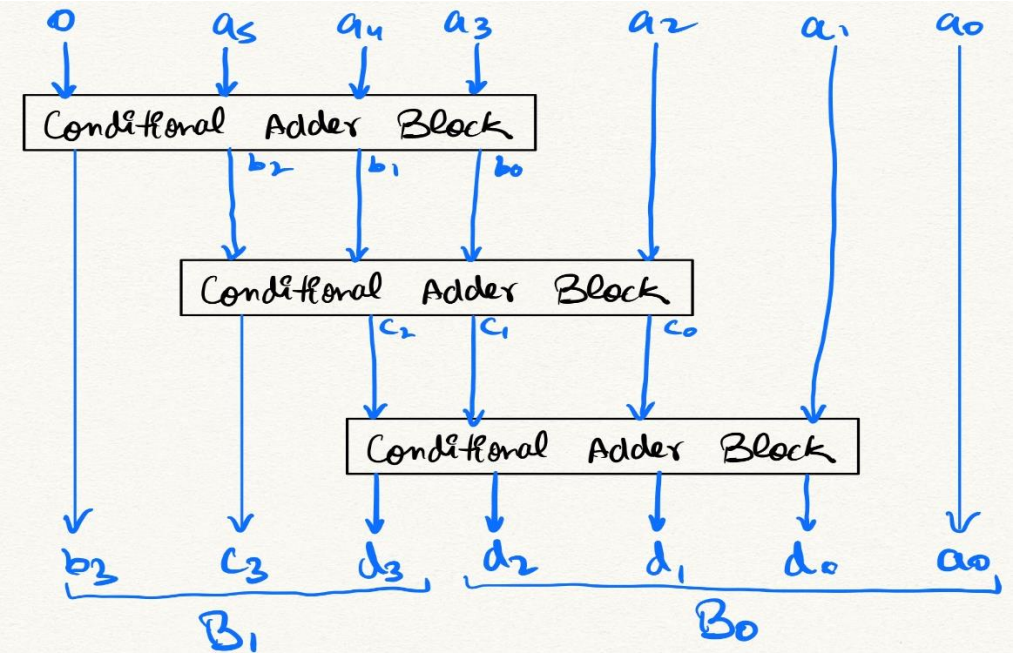
Step No.	Operation	B_1	B_0	
01	INITIALIZATION	0000	0000	$a_5a_4a_3a_2a_1a_0$
02	L-SHIFT	0000	$000a_5$	$a_5a_4a_3a_2a_1a_0$
03	L-SHIFT	0000	$00a_5a_4$	$a_5a_4a_3a_2a_1a_0$
04	L-SHIFT	0000	$0a_5a_4a_3$	$a_5a_4a_3a_2a_1a_0$
05	ADD-3	0000	$b_3b_2b_1b_0$	$a_5a_4a_3a_2a_1a_0$
06	L-SHIFT	$000b_3$	$b_2b_1b_0a_2$	$a_5a_4a_3a_2a_1a_0$
07	ADD-3	$000b_3$	$c_3c_2c_1c_0$	$a_5a_4a_3a_2a_1a_0$
08	L-SHIFT	$00b_3c_3$	$c_2c_1c_0a_1$	$a_5a_4a_3a_2a_1a_0$
09	ADD-3	$00b_3c_3$	$d_3d_2d_1d_0$	$a_5a_4a_3a_2a_1a_0$
10	L-SHIFT	$0b_3c_3d_3$	$d_2d_1d_0a_0$	$a_5a_4a_3a_2a_1a_0$

Consider the above implementation of *double dabble algorithm* on a general 6-bit binary vector. The following observations can be made based on the above

implementation that might help in figuring out the circuital implementation for this algorithm.

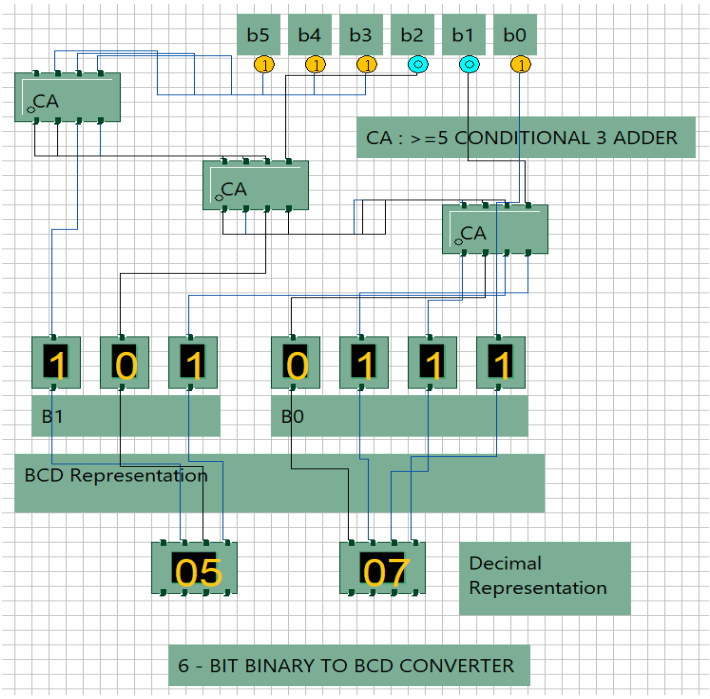
- The three most significant bits in the binary input are *left-shifted* without any *add-3* operation because the decimal value of the zeroth nibble would certainly be less than 5.
- *Add-3* operation is performed at most *thrice* and that too only on the zeroth nibble. No adjustment or *add-3* operation is performed on the first nibble.

The first observation hints that the first three *left-shifts* can be done consecutively without any checking for $(B_0)_{10} < 5$ post each shift. The second observation hints on the need of only three *conditional adder blocks* in the circuit. The following is the circuital implementation of the double-dabble algorithm for a 6-bit binary number that aims at following the same steps as in the above table.



- The circuit starts directly with the *fourth* step of the algorithm where B_1 is 0000 and B_0 is $0a_5a_4a_3$. The binary number $B_0 = 0a_5a_4a_3$ is passed through a *conditional adder block* that gives $b_3b_2b_1b_0$ as the 4-bit binary output.
- *Left-shift* operation is performed and a_2 is concatenated with the three rightmost bits $b_2b_1b_0$. At this step, B_1 is $000b_3$ and B_0 is $b_2b_1b_0a_2$.
- The zeroth nibble $b_2b_1b_0a_2$ is passed through another *conditional adder block* that gives $c_3c_2c_1c_0$ as the 4-bit binary output.

- *Left-shift* operation is performed again and a_1 is concatenated with the three rightmost bits $c_2c_1c_0$. At this step, B_1 is $00b_3c_3$ and B_0 is $c_2c_1c_0a_1$.
- The zeroth nibble $c_2c_1c_0a_1$ is passed through the final *conditional adder block* that gives $d_3d_2d_1d_0$ as the 4-bit binary output.
- Now the least significant bit a_0 is simply concatenated to the result of the previous step. Groups of 4 bits are made from the rightmost bit and B_1 is finally $0b_3c_3d_3$ and B_0 is $d_2d_1d_0a_0$.



7-BIT BINARY TO BCD CONVERTOR

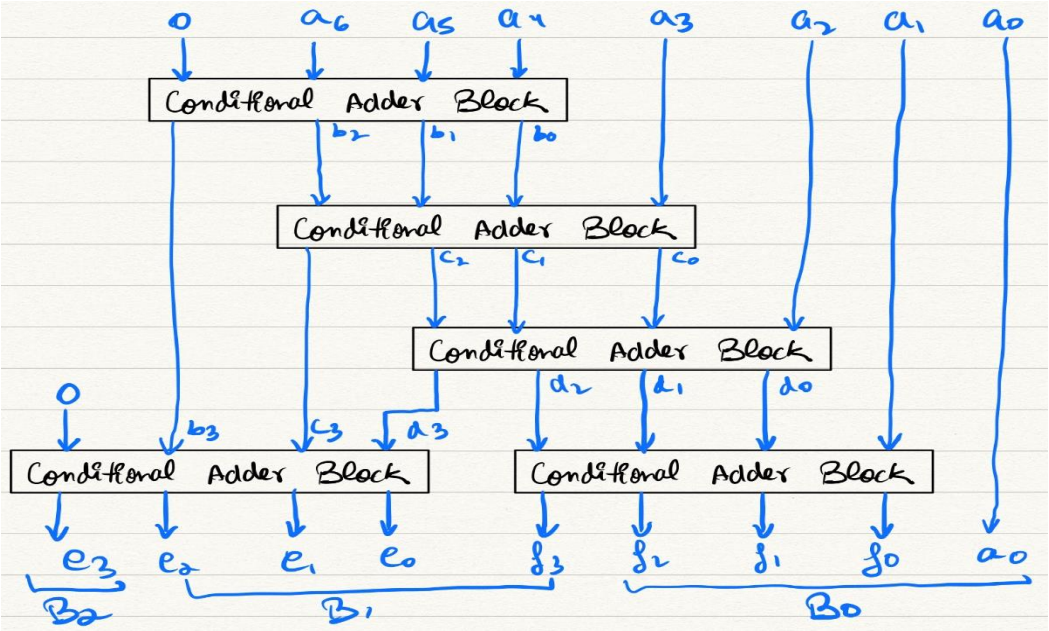
Step No.	Operation	B_2	B_1	B_0	
01	INITIALIZATION	0000	0000	0000	$a_6a_5a_4a_3a_2a_1a_0$
02	L-SHIFT	0000	0000	$000a_6$	$a_6a_5a_4a_3a_2a_1a_0$
03	L-SHIFT	0000	0000	$00a_6a_5$	$a_6a_5a_4a_3a_2a_1a_0$
04	L-SHIFT	0000	0000	$0a_6a_5a_4$	$a_6a_5a_4a_3a_2a_1a_0$
05	ADD-3	0000	0000	$b_3b_2b_1b_0$	$a_6a_5a_4a_3a_2a_1a_0$
06	L-SHIFT	0000	$000b_3$	$b_2b_1b_0a_3$	$a_6a_5a_4a_3a_2a_1a_0$
07	ADD-3	0000	$000b_3$	$c_3c_2c_1c_0$	$a_6a_5a_4a_3a_2a_1a_0$
08	L-SHIFT	0000	$00b_3c_3$	$c_2c_1c_0a_2$	$a_6a_5a_4a_3a_2a_1a_0$
09	ADD-3	0000	$00b_3c_3$	$d_3d_2d_1d_0$	$a_6a_5a_4a_3a_2a_1a_0$
10	L-SHIFT	0000	$0b_3c_3d_3$	$d_2d_1d_0a_1$	$a_6a_5a_4a_3a_2a_1a_0$

11	ADD-3	0000	$e_3e_2e_1e_0$	$f_3f_2f_1f_0$	$a_6a_5a_4a_3a_2a_1a_0$
12	L-SHIFT	$000e_3$	$e_2e_1e_0f_3$	$f_2f_1f_0a_0$	$a_6a_5a_4a_3a_2a_1a_0$

Consider the above implementation of *double dabble algorithm* on a general 7-bit binary vector. The following observations can be made based on the above implementation that might help in figuring out the circuital implementation for this algorithm.

- The three most significant bits in the binary input are *left-shifted* without any *add-3* operation because the decimal value of the zeroth nibble would certainly be less than 5.
- *Add-3* operation is performed at most *five times*. Out of maximum of five times, at most *four add-3* operations would be needed on the zeroth nibble and at most *one* on the first nibble. No adjustment or *add-3* operation is performed on the second nibble.

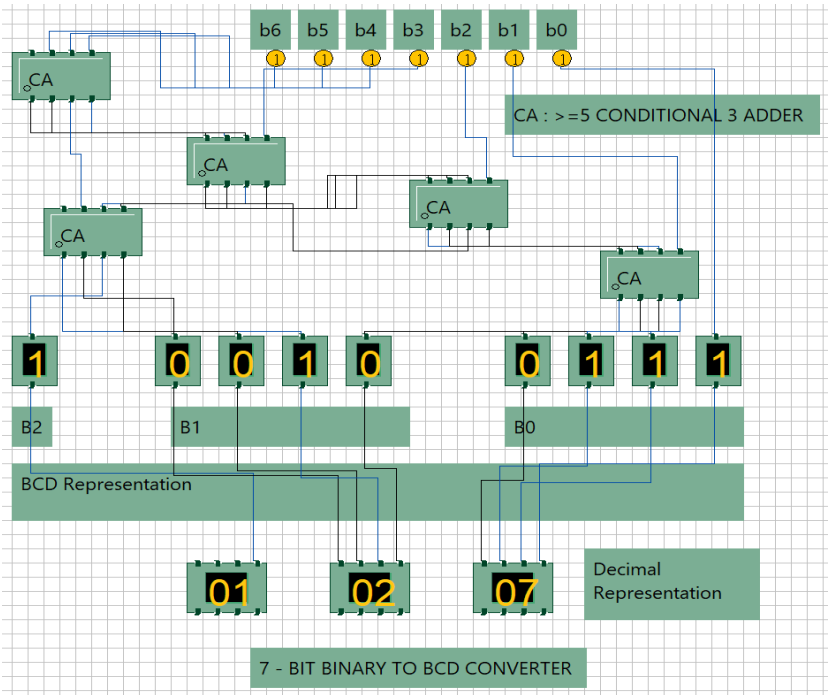
The first observation hints that the first three *left-shifts* can be done consecutively without any checking for $(B_0)_{10} < 5$ post each shift. The second observation hints on the need of exactly five *conditional adder blocks* in the circuit. The following is the circuital implementation of the double-dabble algorithm for a 7-bit binary number that aims at following the same steps as in the above table.



- The circuit starts directly with the *fourth* step of the algorithm where B_1 is 0000 and B_0 is $0a_6a_5a_4$ (B_2 is anyway 0000 till the last step). The

binary number $B_0 = 0a_6a_5a_4$ is passed through a *conditional adder block* that gives $b_3b_2b_1b_0$ as the 4-bit binary output.

- *Left-shift* operation is performed and a_3 is concatenated with the three rightmost bits $b_2b_1b_0$. At this step, B_1 is $000b_3$ and B_0 is $b_2b_1b_0a_3$.
- The zeroth nibble $b_2b_1b_0a_3$ is passed through another *conditional adder block* that gives $c_3c_2c_1c_0$ as the 4-bit binary output.
- *Left-shift* operation is performed again and a_2 is concatenated with the three rightmost bits $c_2c_1c_0$. At this step, B_1 is $00b_3c_3$ and B_0 is $c_2c_1c_0a_2$.
- The zeroth nibble $c_2c_1c_0a_2$ is passed through the third *conditional adder block* that gives $d_3d_2d_1d_0$ as the 4-bit binary output.
- *Left-shift* operation is performed and a_1 is concatenated with the three rightmost bits $d_2d_1d_0$. At this step, B_1 is $0b_3c_3d_3$ and B_0 is $d_2d_1d_0a_1$.
- *Now, this step is different than the ones in the previous part*. B_1 is $0b_3c_3d_3$ which means its decimal value can be greater than or equal to 5. Therefore, in this step both the zeroth nibble $d_2d_1d_0a_1$ and the first nibble $0b_3c_3d_3$ are passed through separate *conditional adder blocks* that give $f_3f_2f_1f_0$ and $e_3e_2e_1e_0$ respectively as the 4-bit binary outputs.
- Now the least significant bit a_0 is finally concatenated to the result of the previous step. Groups of 4 bits are made from the rightmost bit and B_2 is finally $000e_3$, B_1 is $e_2e_1e_0f_3$ and B_0 is $f_2f_1f_0a_0$.



GROUP MEMBERS

Group ID	Roll No.	Name
09	19CS10011	Anish Sofat
09	19CS10031	Abhishek Gandhi
09	19CS10044	Nakul Aggarwal
09	19CS10051	Sajal Chhamunya
09	19CS10053	Satvik Bansal