Bank Queue System:

Team members:

1. Ahmed Nasser Emam
2. Ahmed Mohamed Mahmoud Taha
3. Adham Ibrahim

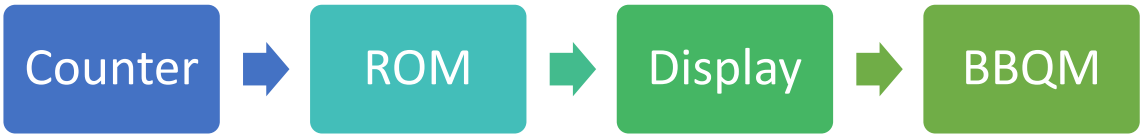
Supervisors:

Dr. Ahmed shalaby

Eng. Mostafa samy

Content:

The hierarchy of the modules or blocks:



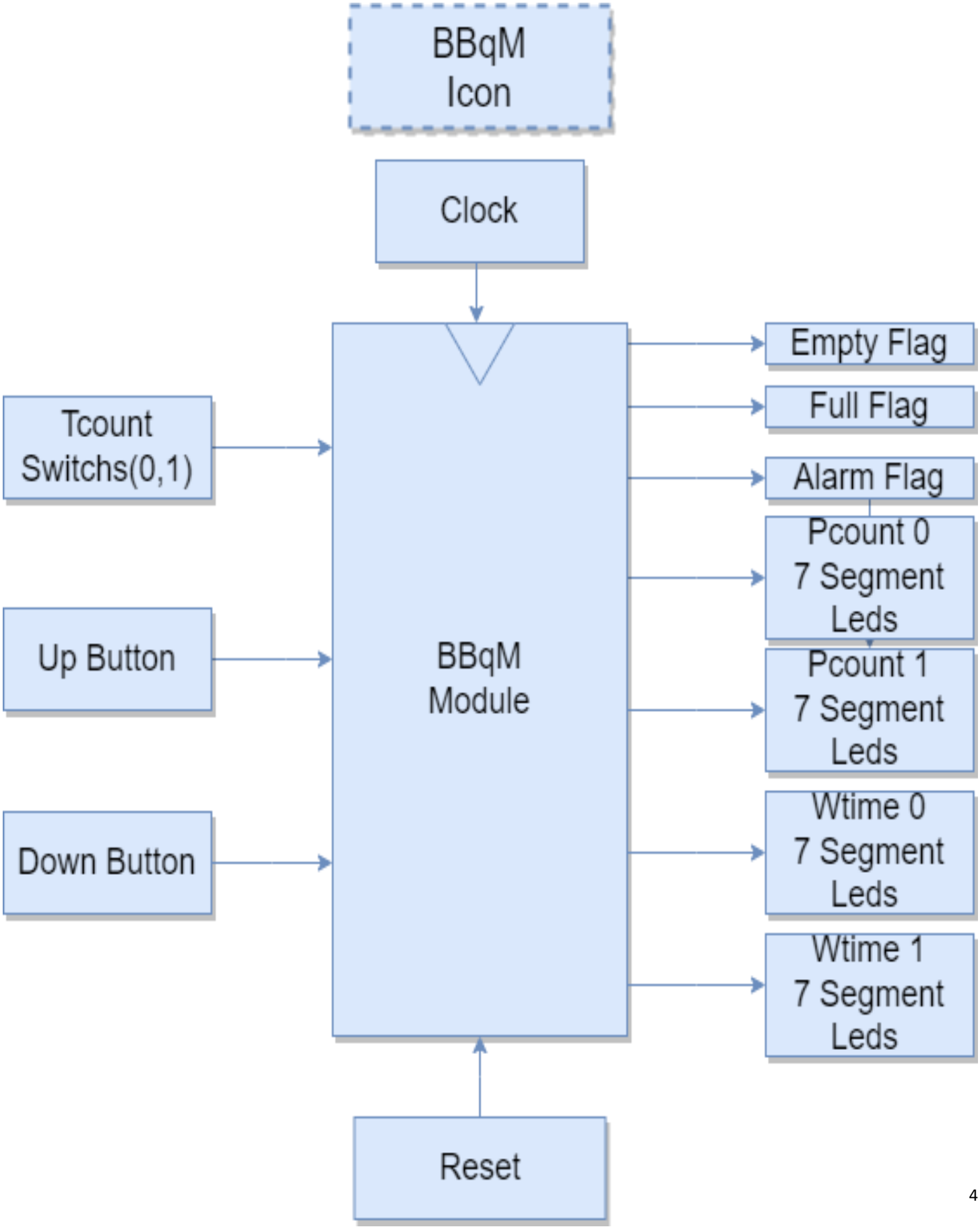
Implementation

It's a Bank system which based on a queue structure. The queue structure is the main part that arrange the flow of clients on the queue. The System consist of a lot of parts or modules that every single one of them has a specific job. The counter that control, increasing and decreasing the number of clients. The counter has some submodules in it. First, the clock divider that control the time of increasing or the time of decreasing on the machine using the push buttons, we made it appropriate for the system of the (FBGA). Second , a flip flops that we used to solve the problem of denouncing (considering one click as a many of clicks) so that usage of counter will not have any problems .Third , the counter itself that are responsible for increasing , decreasing to control the number of clients are in the queue , increasing (using push button up) from zero to seven (the maximum value) then cannot add more , decreasing (using push button down) from seven to zero (the minimum value) then cannot less again . forth, flags that show if the queue is empty or full to know if we enable to add or decrease. It used also in displaying function to be shown. The Finite State Machine which collects the number of available tellers with the number of people are asking for service (in the queue) and calculate the waiting time of every single client using a specific equation then we store all the probabilities possible of the waiting time. Tellers and client's data and the clients waiting time stored in a special place called (ROM). The Read Only Memory (ROM) in which we store the values of the finite state machine results. we used five bits for every state and assigned it's waiting time (that been calculated in the FSM before). It helps in arranging the sequence of states and aid in the performance of the system. the display function that is responsible for showing all of the results, use two equations to show the waiting time in a decimal number in which the human can understand, assigning all the input and the output produced and arrange it's work on the FPGA. the top-level module (BBQM) which responsible for calling all the modules and arrange them work on the FPGA. It's contained all of modules on it is the main module that directly interact with the machine. The system is installed in the machine directly without any problems after the assignment of the controller of input and the parts in which the output will produce in.

Abstract Of Project

System used to show the number of people in the queue of Banha Bank branch (**BBqM™**) and how much waiting time they will spend in the queue

BBqM icon

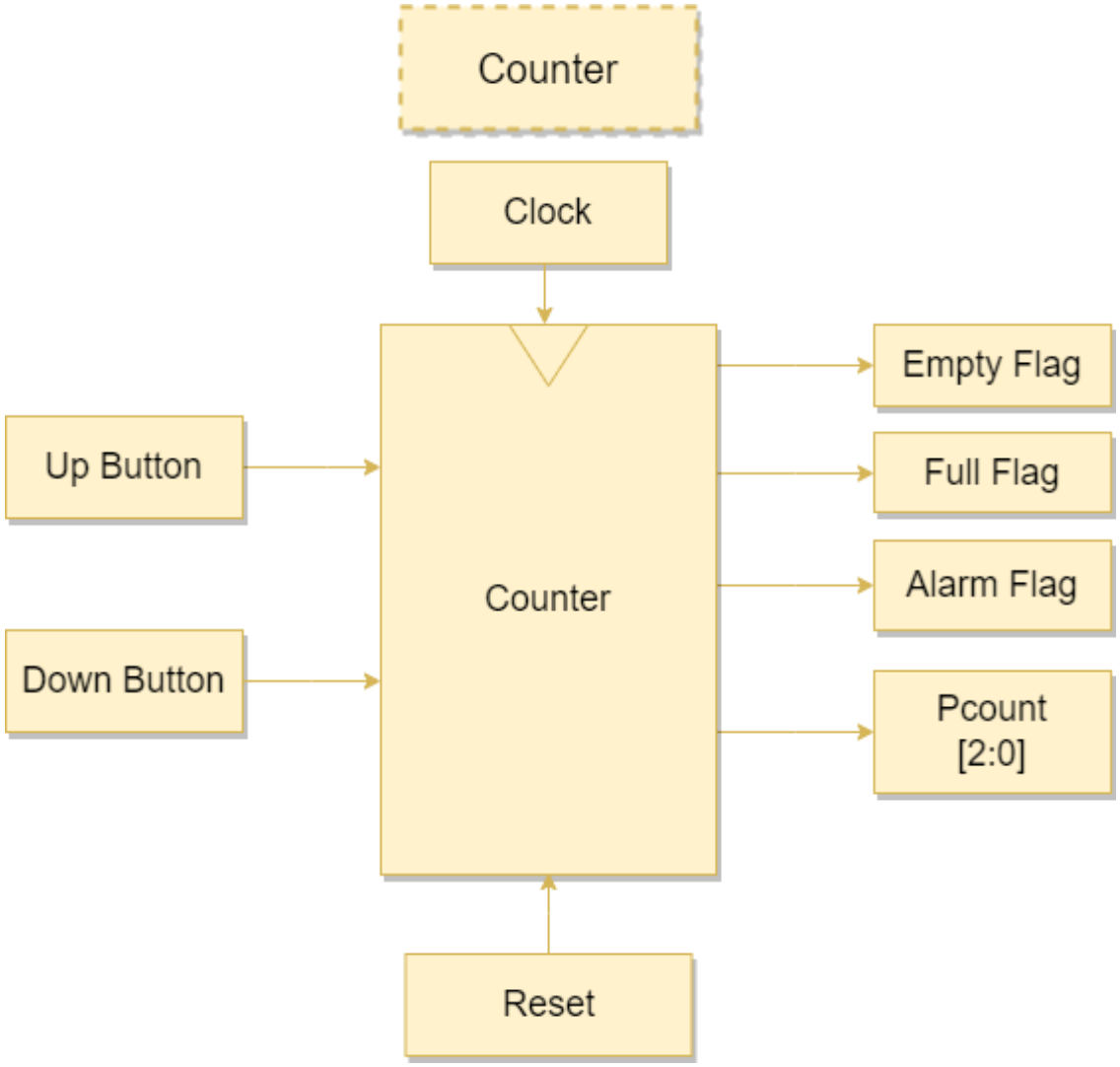

BBqM
Icon
Clock
Tcount
Switchs(0,1)
Up Button
Down Button
BBqM
Module
Reset
Empty Flag
Full Flag
Alarm Flag
Pcount 0
7 Segment
Leds
Pcount 1
7 Segment
Leds
Wtime 0
7 Segment
Leds
Wtime 1
7 Segment
Leds

Modules

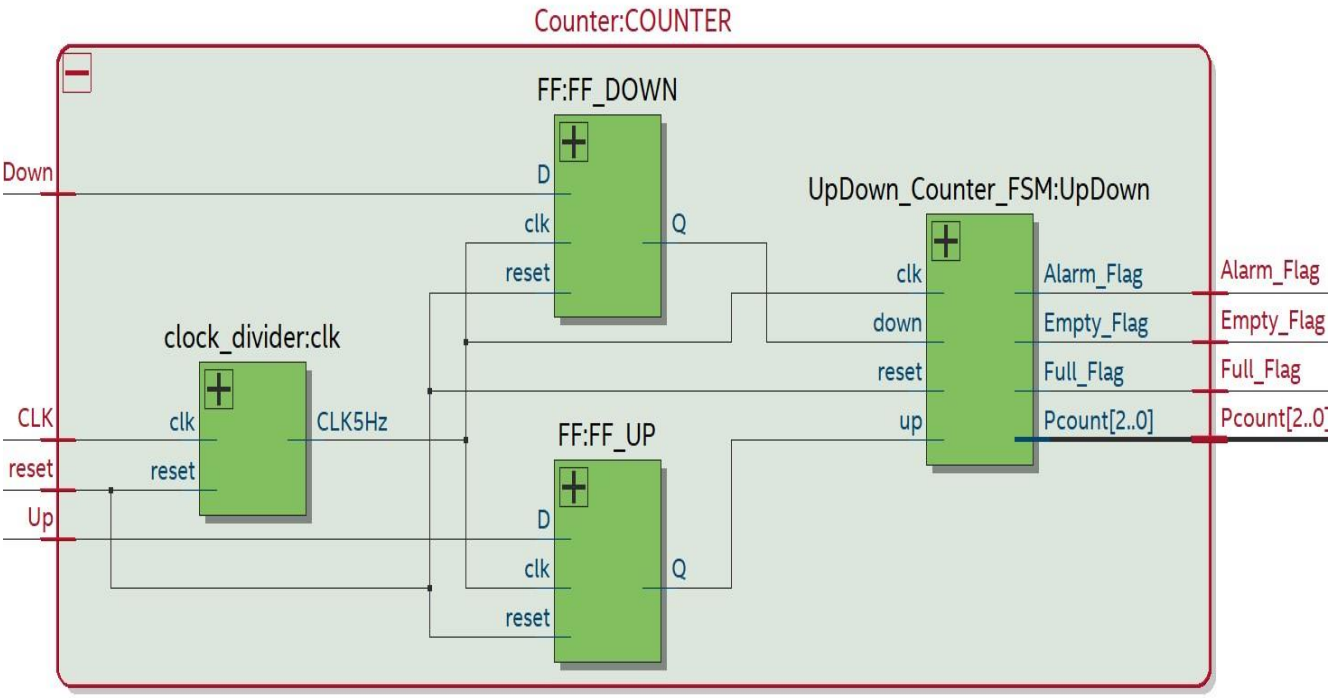
Counter

we have a (counter) which is increasing or decreasing using the push buttons with the maximum number of seven clients on the queue arranged by the (people counter), every one of clients has his own (wait time). The (wait time) differs from a client to another, affected by many factors like the number of clients being in the queue (people counter), the (wait time) of the other clients that standing in front of you and how many tellers is available to response the clients' needs. There are many parts (submodules) in the counter that every submodule of them has a unique job. (Clock divider, FF, UpDown Counter FSM)

- **Block Diagram of Counter**



- **Counter Structure**



- **Inputs**

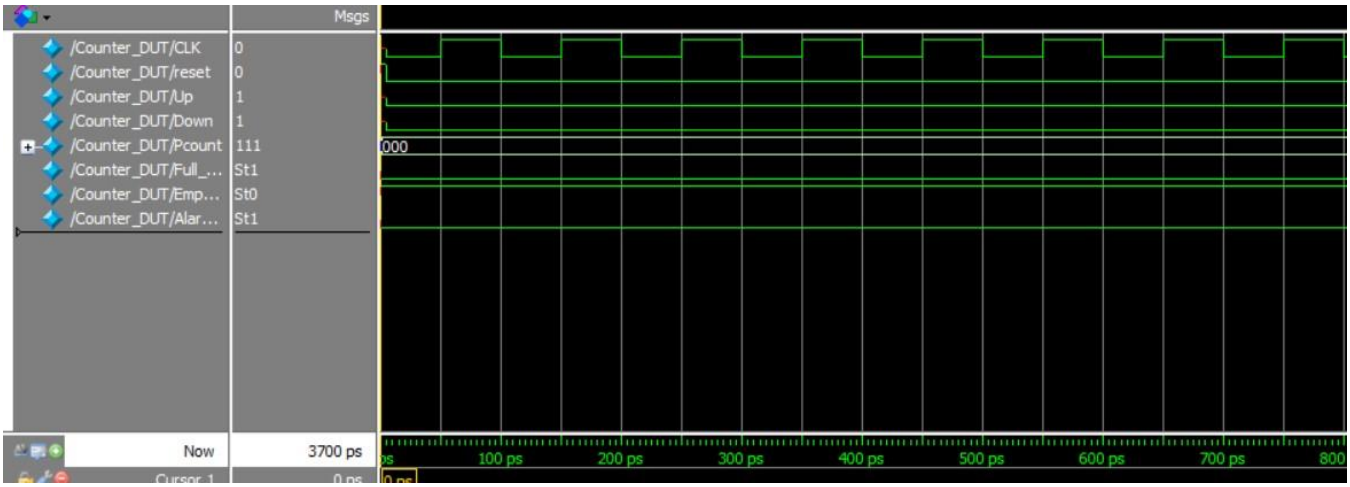
CLK	Clock which coming from FPGA (50 MHz)
reset	Reset switch which reset all the system to initial state
Up	Push button which increments Pcount (people in the Queue)
Down	Push button which decrements Pcount (people in the Queue)

- **Outputs**

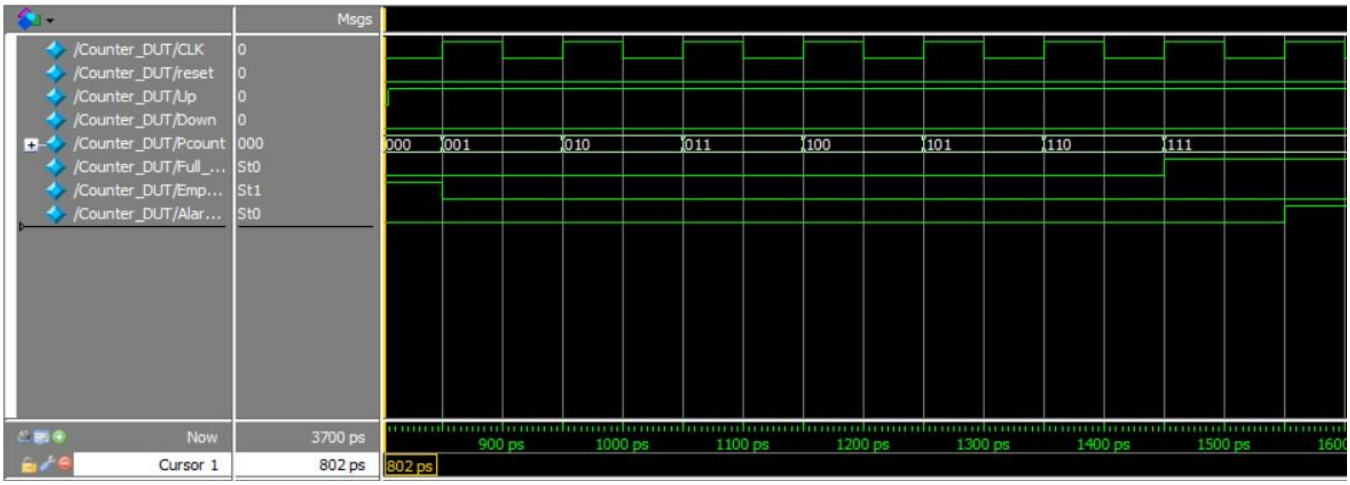
Pcount [2:0]	The counted people in the queue in 3 bit binary form
Empty_Flag	LED to show if the Queue is empty or not
Full_Flag	LED to show if the Queue is full or not
Alarm_Flag	LED to show the error states when the Queue is empty and Down Push button is pressed Or the Queue is full and Up Push button is pressed

- **Testbench**

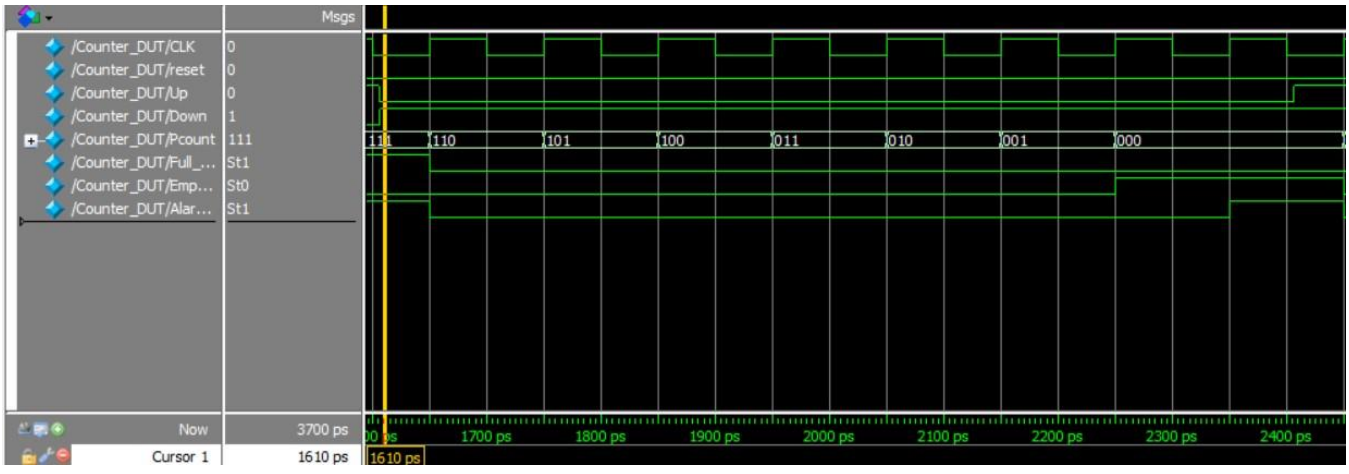
At Up = 0, Down = 0 for 800ps (No Pressing)



At Up = 1, Down = 0 for 800ps (Up pressed and Down not)



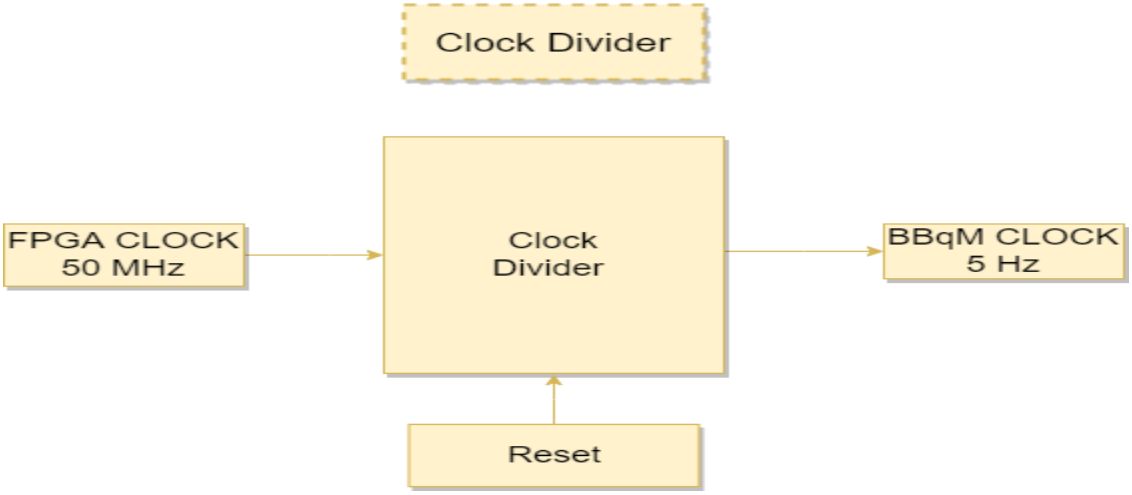
At Up = 0, Down = 1 for 800ps (Down pressed and Up not)



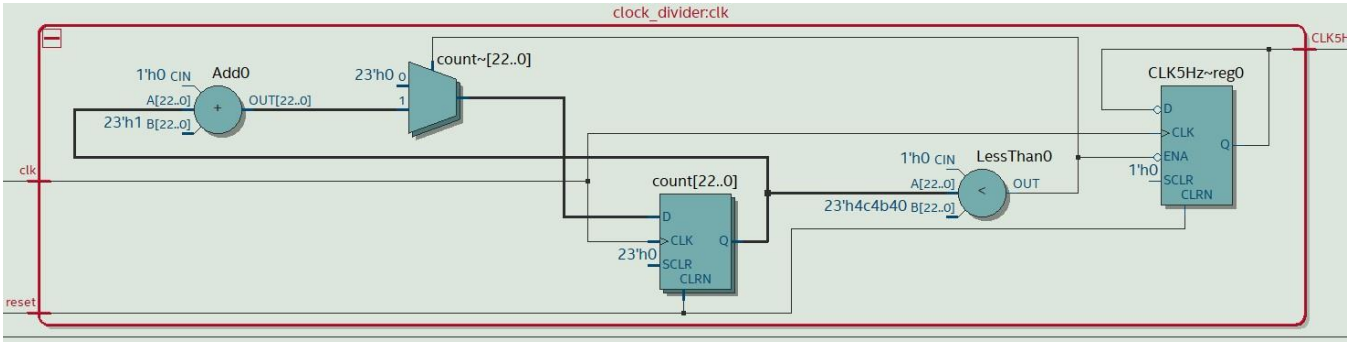
Sub Modules of Counters

➔ **First, the clock divider** that controls the time of increasing or the time of decreasing on the machine using the push buttons, we made it appropriate for the system of the (FPGA), it gives us the appropriate number of **Hs (5 Hz)** that we need to increase or decrease the counter which is 5 000 000

- **Block Diagram of Clock divider**



- **Clock divider Structure**



- **Inputs**

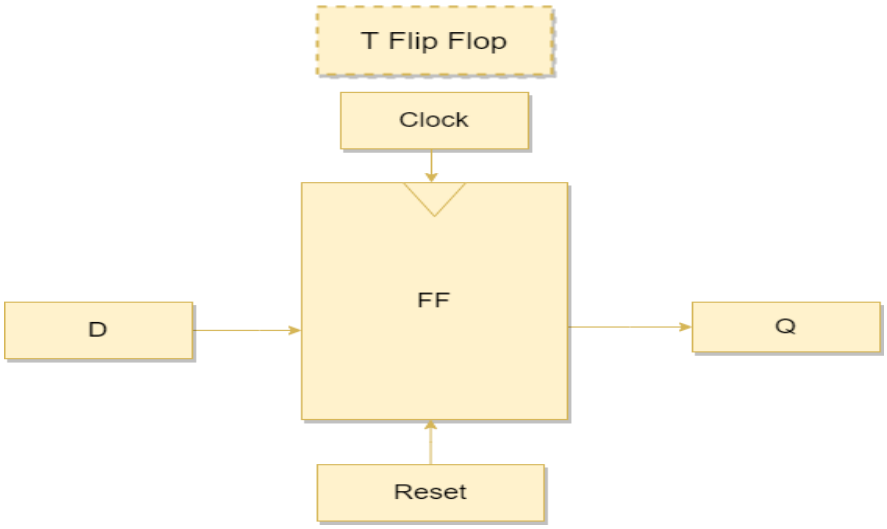
clk	Clock which coming from FPGA (50 MHz)
reset	Reset switch which reset all the system to initial state

- **Outputs**

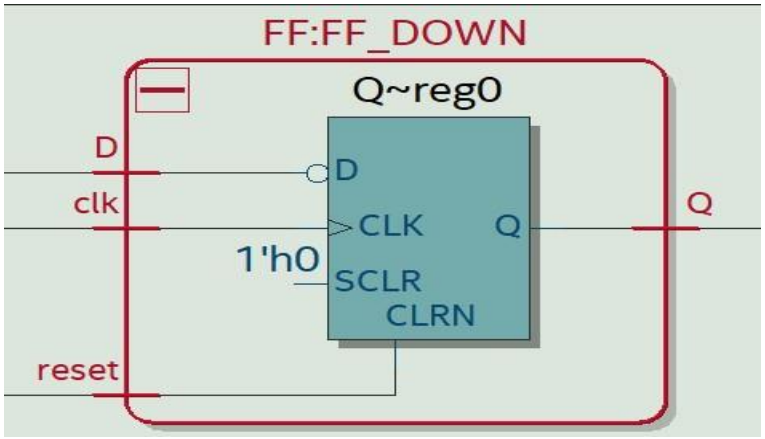
CLK5Hz	Clock which is suitable for BBqM system (5 Hz)

➔ **<u>Second, T flip flops</u>** we use it to solve the problem of denouncing (considering one click as a many of clicks) so that usage of counter will not have any problems.

- **Block Diagram of T Flip Flop**



- **T Flip Flop Structure**



- **Inputs**

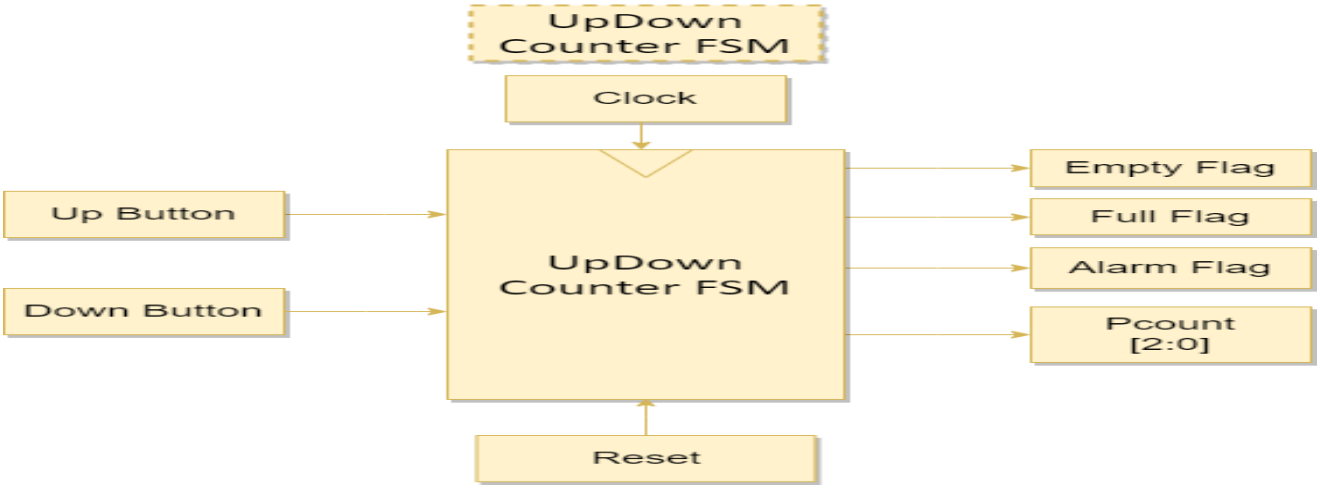
clk	5 Hz clock coming from Clock divider module
reset	Reset switch which reset all the system to initial state
D	Push button before debouncing problem (Up or Down)

- **Outputs**

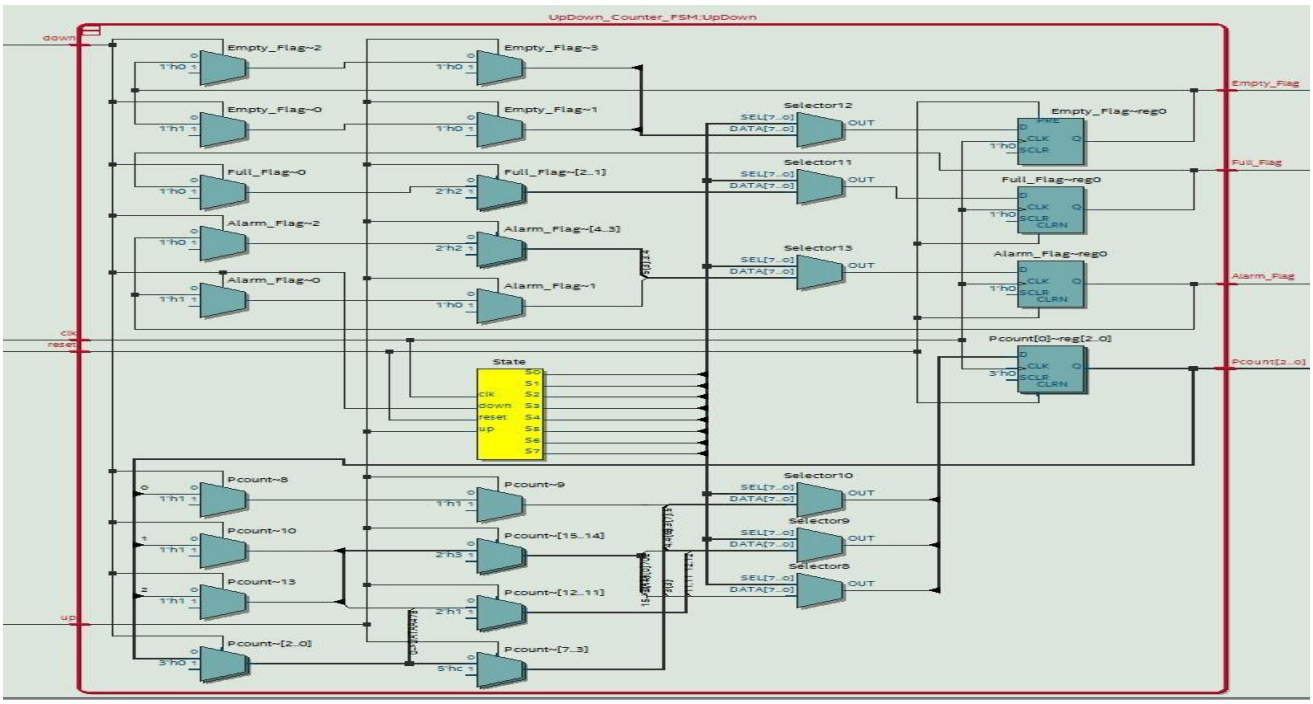
Q	Push button after debouncing problem (Up or Down)

➔ **Third, UpDown Counter FSM** itself that are responsible for increasing, decreasing to control the number of clients is in the queue based on the FSM that is built in the same module, increasing (using push button up) from zero to seven (the maximum value) then cannot add more, decreasing (using push button down) from seven to zero (the minimum value) then cannot less again. and show if the queue is empty or full to know if we enable to add or decrease. It used also in displaying function to be shown, also there's a flag called alarm which is on when there's an overflow (more than 7 clients on the row) or underflow (less than 0 clients on the row) on the system and it's also controlled by the FSM built in the up down counter

- **Block Diagram of UpDown Counter FSM**



- **UpDown Counter FSM Structure**



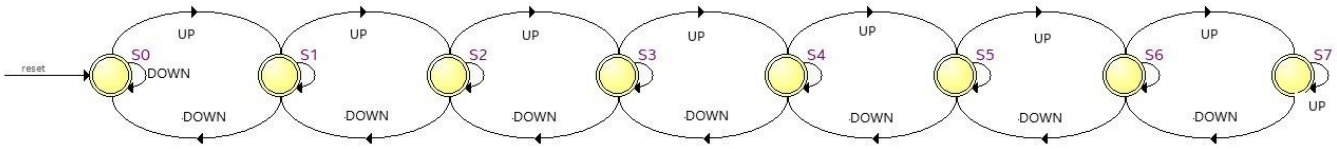
- **Inputs**

clk	5 Hz clock coming from Clock divider module
reset	Reset switch which reset all the system to initial state
up	Debounced push button coming from Up T FF which increments Pcount (people in the Queue)
down	Debounced push button coming from Down T FF which decrements Pcount (people in the Queue)

- **Outputs**

Pcount [2:0]	The counted people in the queue in 3 bit binary form
Empty_Flag	LED to show if the Queue is empty or not
Full_Flag	LED to show if the Queue is full or not
Alarm_Flag	LED to show the error states when the Queue is empty and Down Push button is pressed Or the Queue is full and Up Push button is pressed

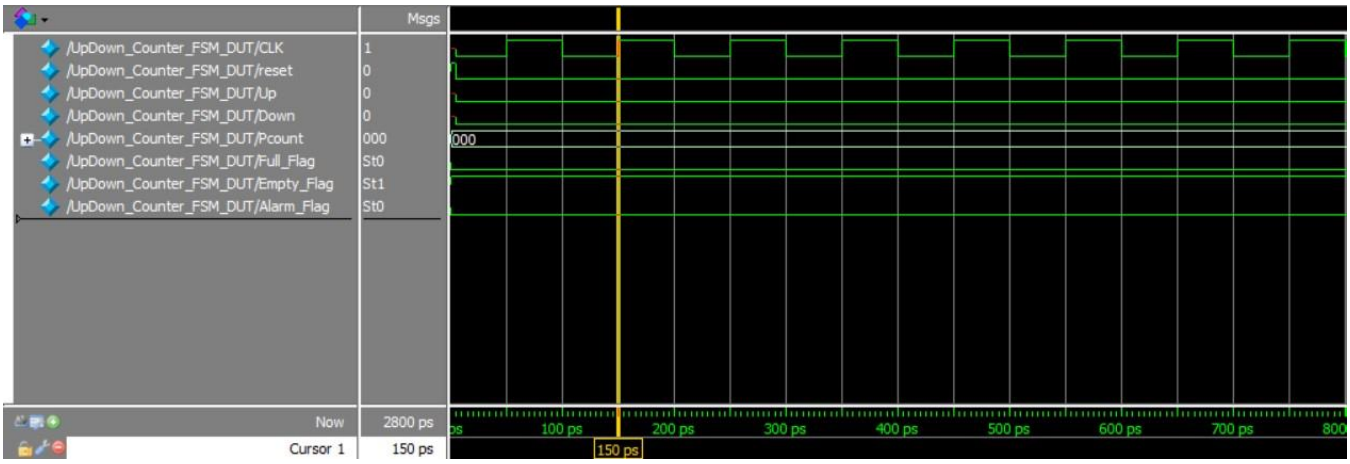
- **FSM**



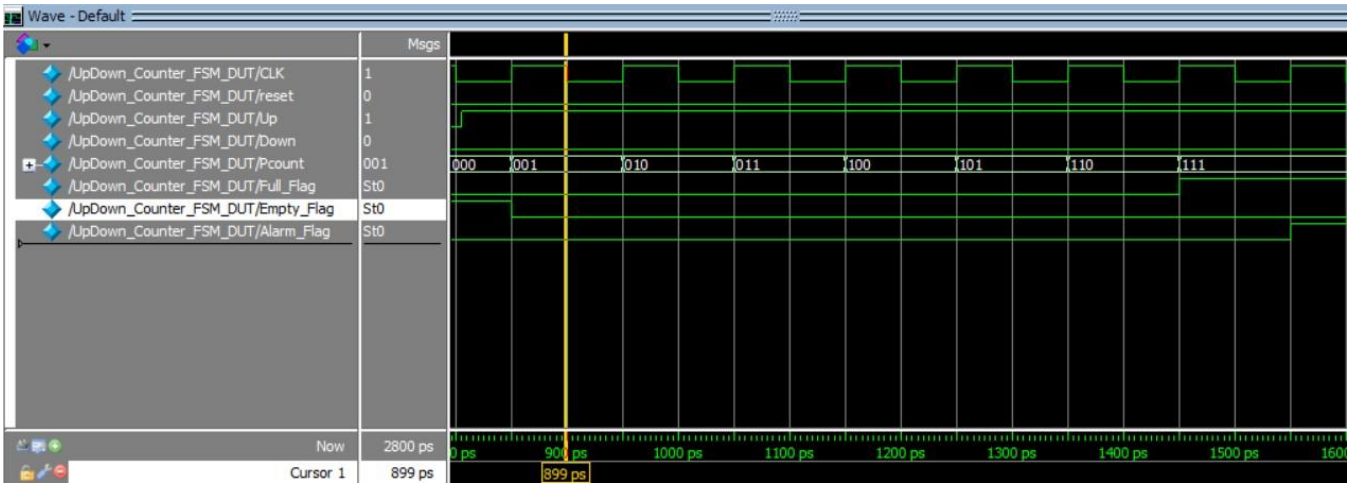
State	Description
S0 when pressing down	Pcount = 0, Empty_Flag = 1, Full_Flag = 0, Alarm_Flag = 1
S0	Pcount = 0, Empty_Flag = 1, Full_Flag = 0, Alarm_Flag = 0
S1	Pcount = 1, Empty_Flag = 0, Full_Flag = 0, Alarm_Flag = 0
S2	Pcount = 2, Empty_Flag = 0, Full_Flag = 0, Alarm_Flag = 0
S3	Pcount = 3, Empty_Flag = 0, Full_Flag = 0, Alarm_Flag = 0
S4	Pcount = 4, Empty_Flag = 0, Full_Flag = 0, Alarm_Flag = 0
S5	Pcount = 5, Empty_Flag = 0, Full_Flag = 0, Alarm_Flag = 0
S6	Pcount = 6, Empty_Flag = 0, Full_Flag = 0, Alarm_Flag = 0
S7	Pcount = 7, Empty_Flag = 0, Full_Flag = 1, Alarm_Flag = 0
S7 when pressing up	Pcount = 7, Empty_Flag = 0, Full_Flag = 1, Alarm_Flag = 1

- **Testbench**

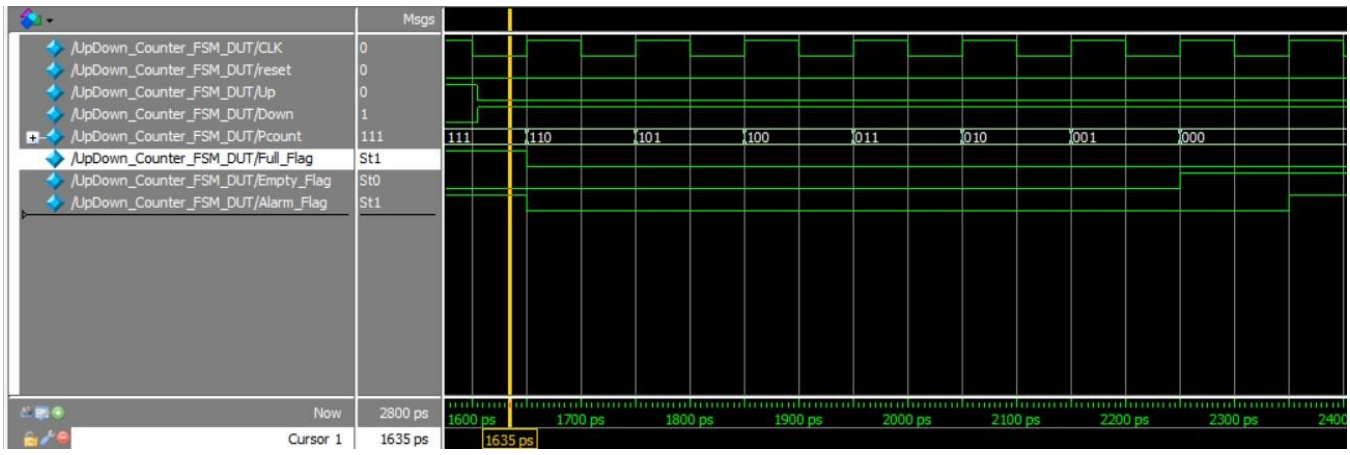
At Up = 0, Down = 0 for 800ps (No Pressing)



At Up = 1, Down = 0 for 800ps (Up pressed and Down not)



At Up = 0, Down = 1 for 800ps (Down pressed and Up not)



Read Only Memory (ROM)

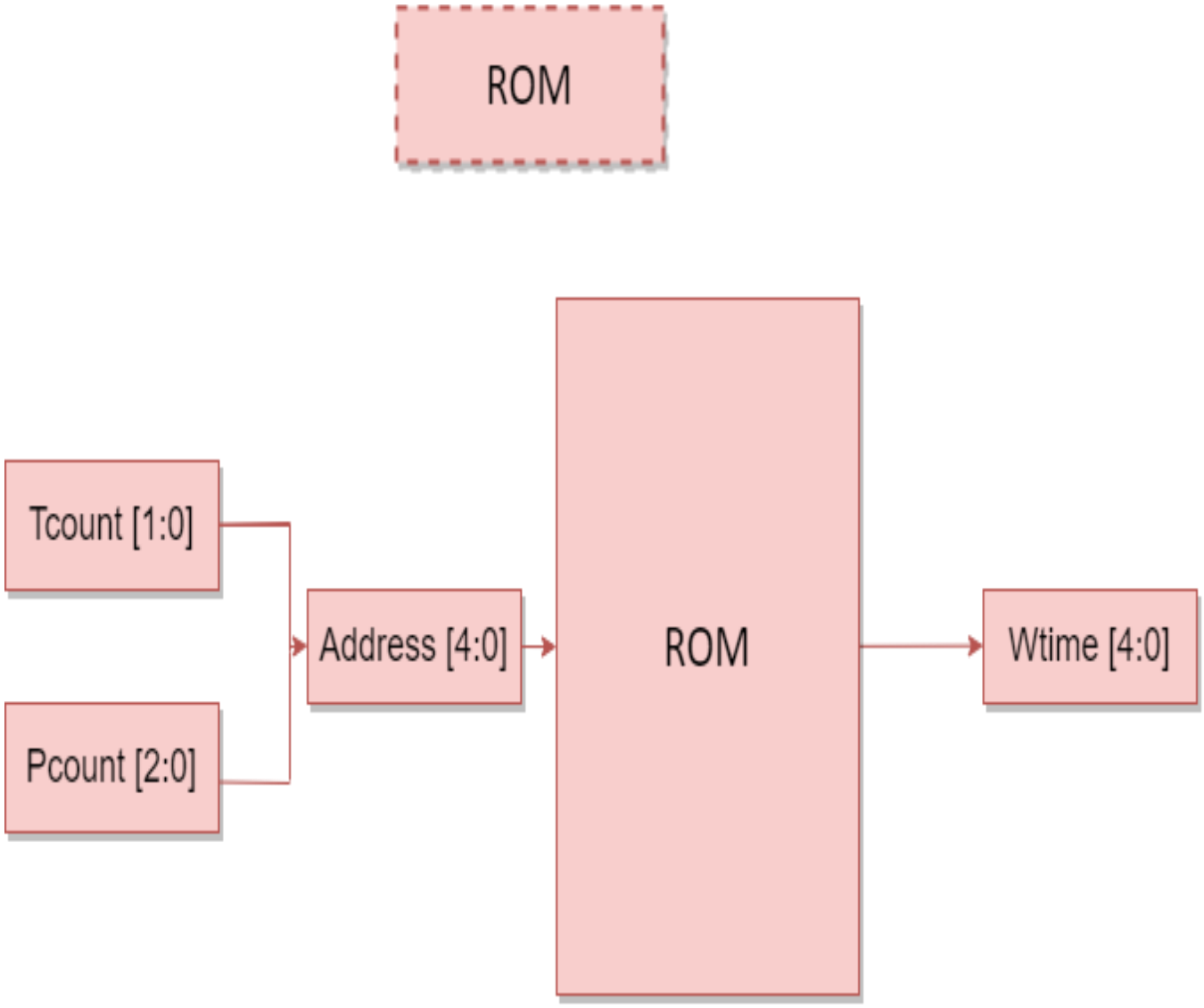
The ROM is the part that contain all the probabilities calculated of the waiting time that arranged by a specific sequence to make it easier and faster to find and select information that have been stored before on it, in order to achieve the reliability, quality and ideal performance of the system cause of all information are sequenced and exist in one place in addition to decreasing and saving time for tellers and clients. For results we used five bits for every state and assigned it's waiting time (that been calculated in the FSM before)

The (waiting time) is calculated by the following equation: -

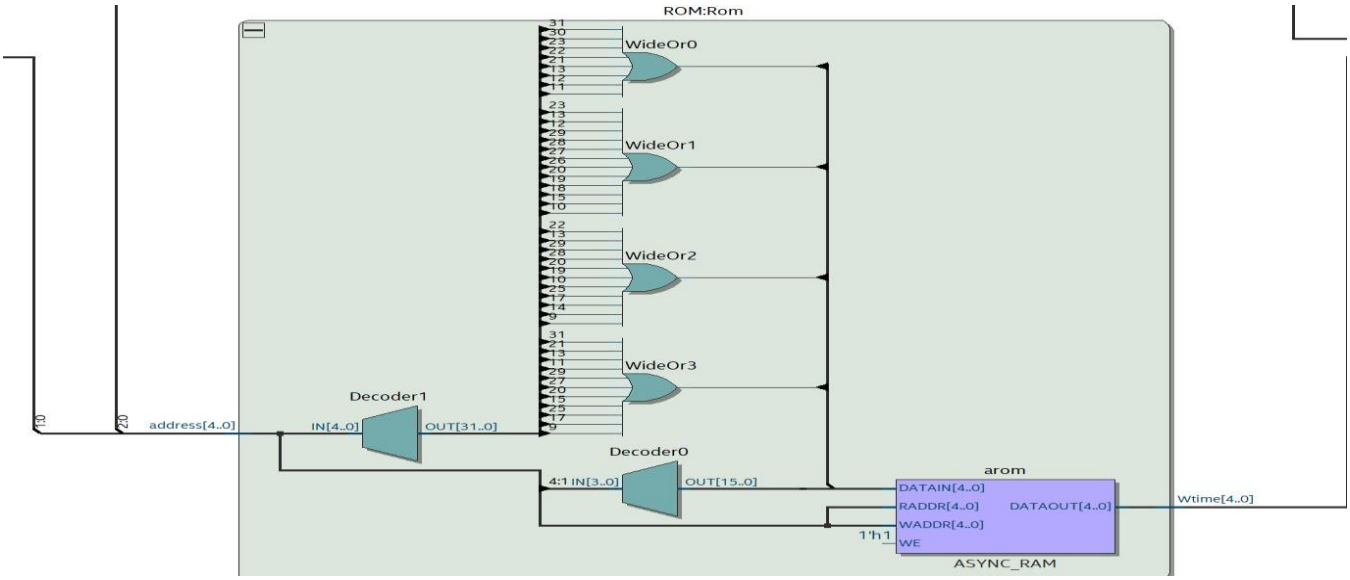
- Time (Pcount = 0) = 0,

- Wtime (Pcount != 0 , Tcount) = 3 * (Pcount + Tcount – 1) / Tcount .

- **Block Diagram of ROM**



- **ROM Structure**



- **Inputs**

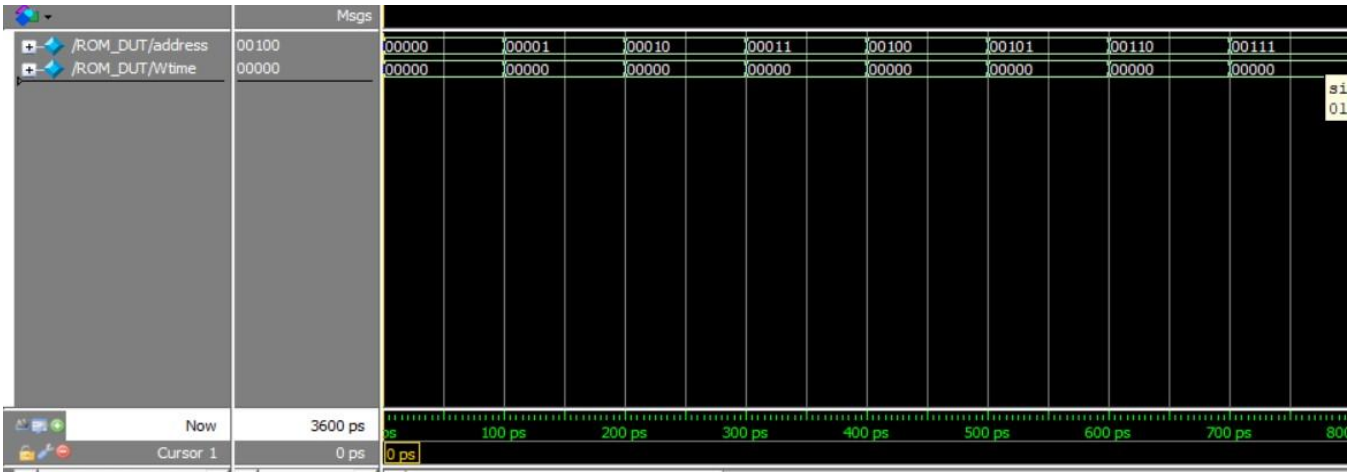
address [4:0]	Concatenation of Tcount (number of available tellers) and Pcount (people in queue) which is the index to get the value corresponding to the current state of Tcount and Pcount Address = { Tcount , Pcount}

- **Outputs**

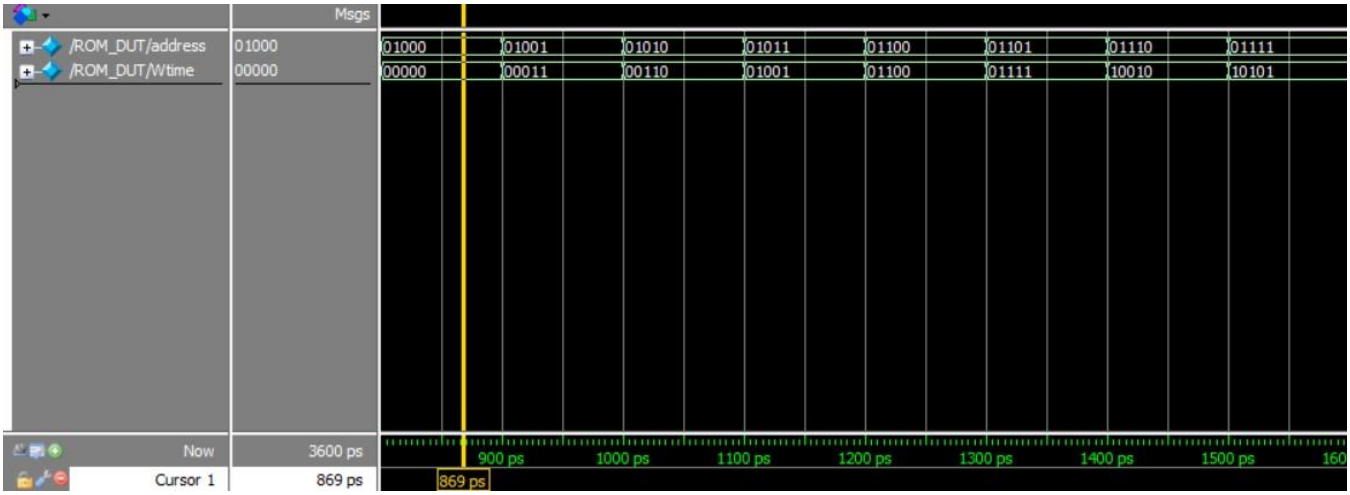
Wtime [4:0]	The waiting time in 5 bit binary form

- **Testbench**

At Tcount = 0

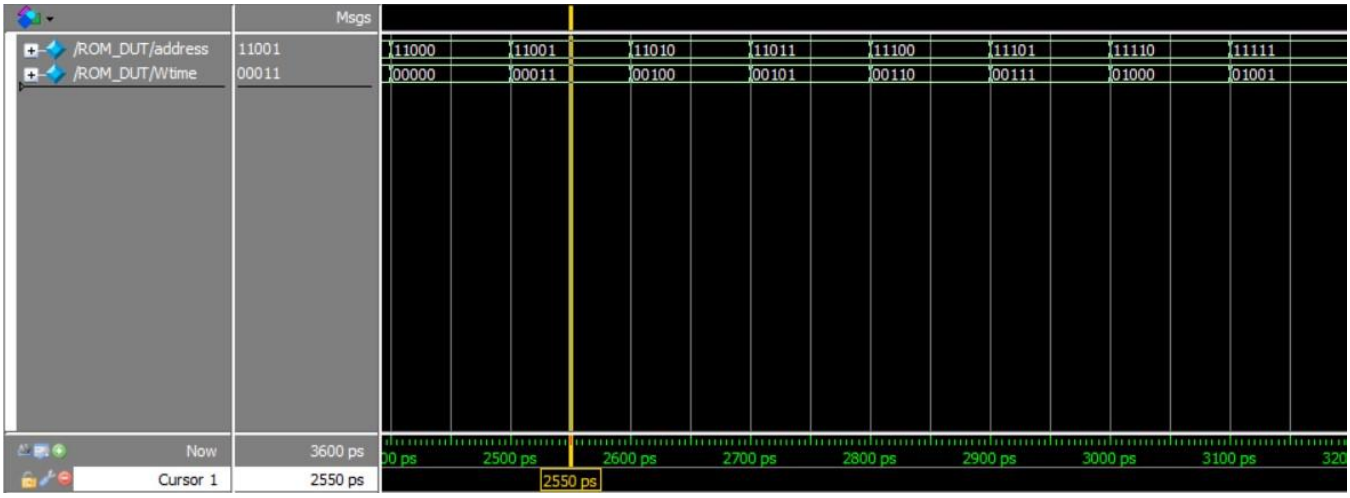


At Tcount = 1



At Tcount = 2

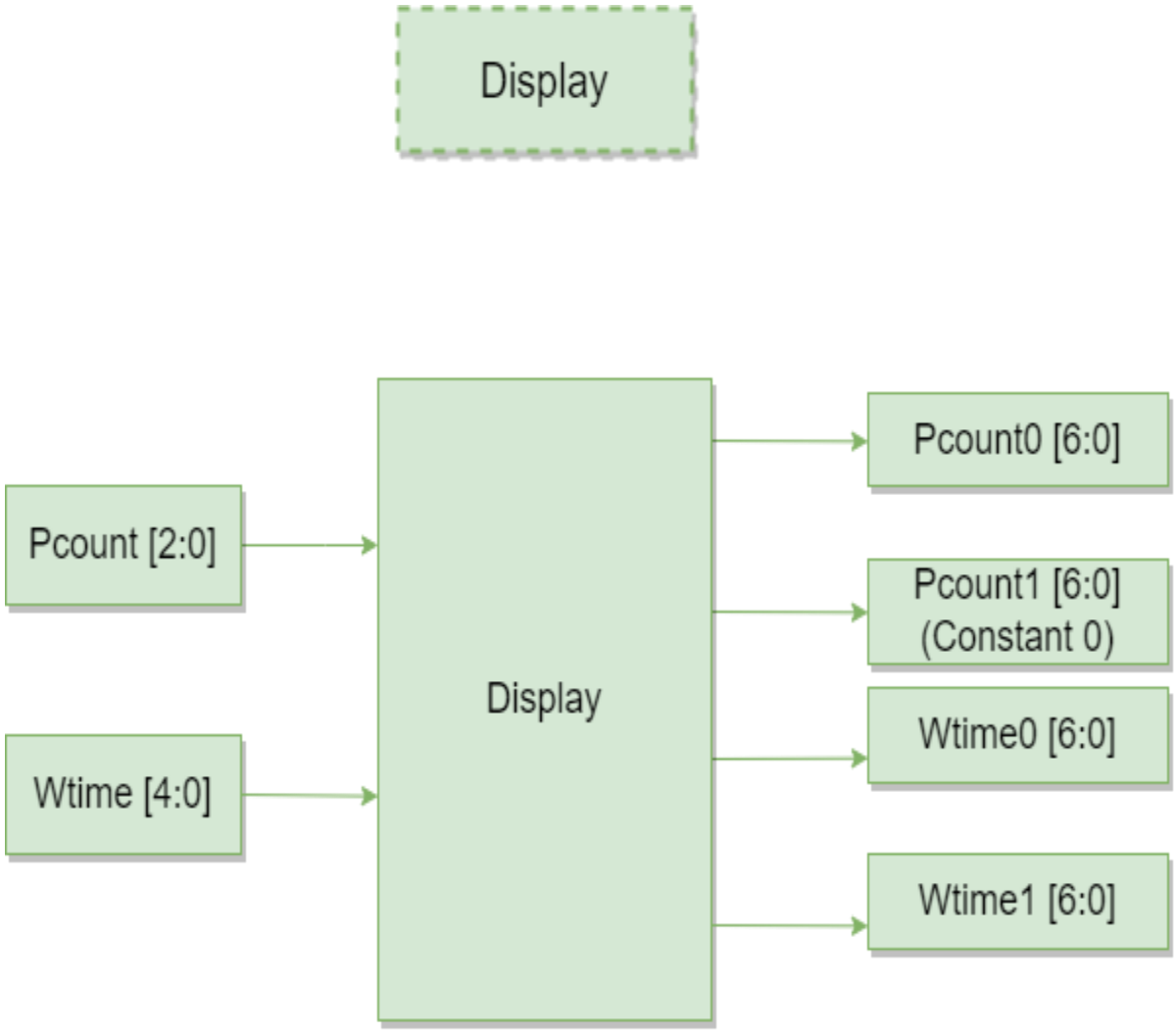
At Tcount = 3



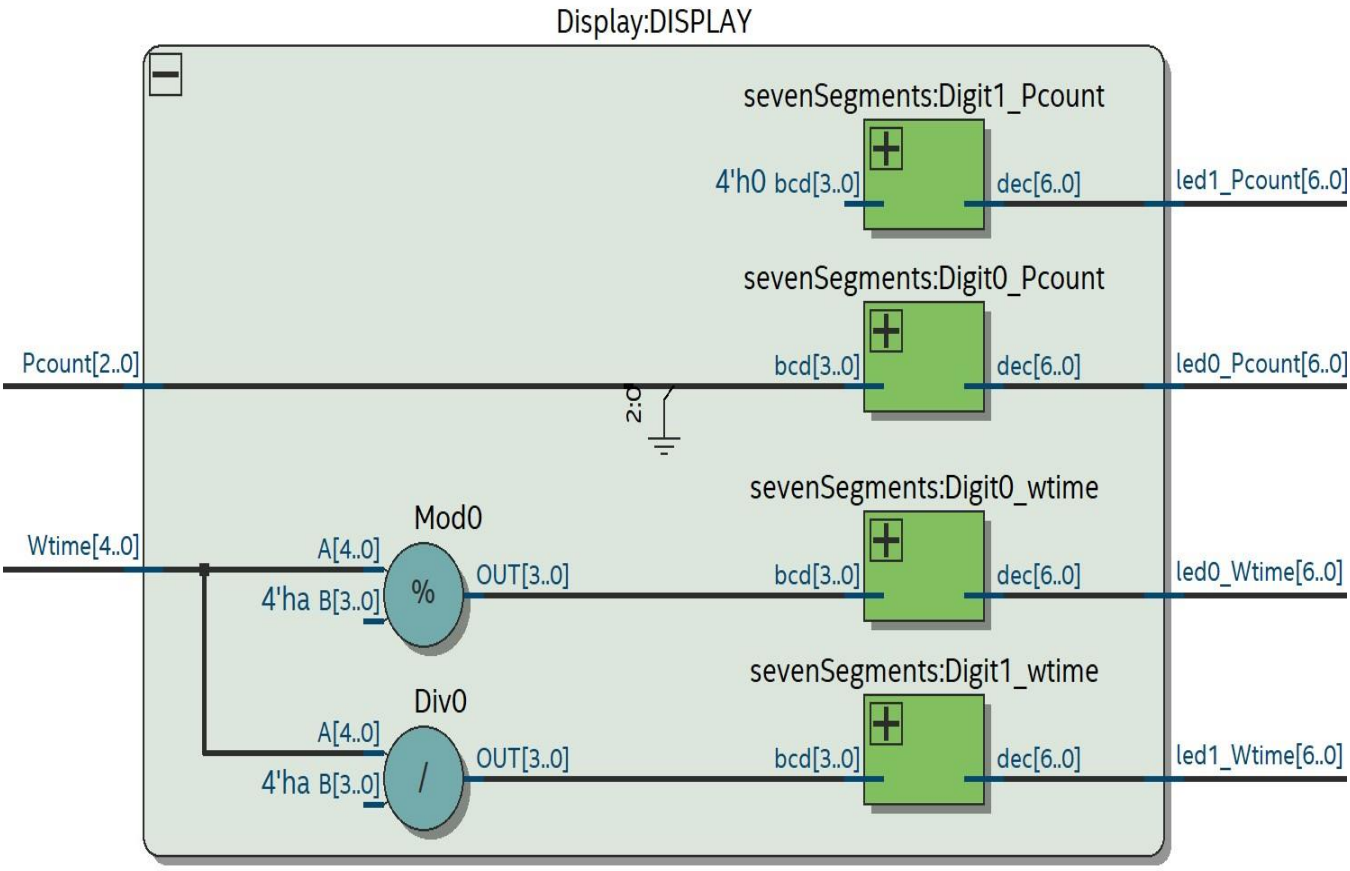
Display

The function of displaying , in this function we display all of our work in the machine , we control the number of tellers by two switches and another three switches to control the number of clients inter ,we increasing and decreasing using a two push buttons , there is also a switch to reset all of the system , we show the number of clients in a seventh segment for which we built a decoder and another two seventh segments for the waiting time with another decoder , using two lids for flags , the (full queue) case that work when there are seven clients in the queue and (empty queue) case that work, and another led for the alarm that work if any matter happened like (overflow) when trying to add more than seven clients, (underflow) when trying remove from no clients are in the queue, no tellers or no clients so on no any services available and otherwise.

- **Block Diagram of Display.**



- **Display Structure**



- **Inputs**

Pcount [2:0]	The counted people in the queue coming from Counter in 3 bit binary form
Wtime [4:0]	The waiting time coming from ROM in 5bit binary form

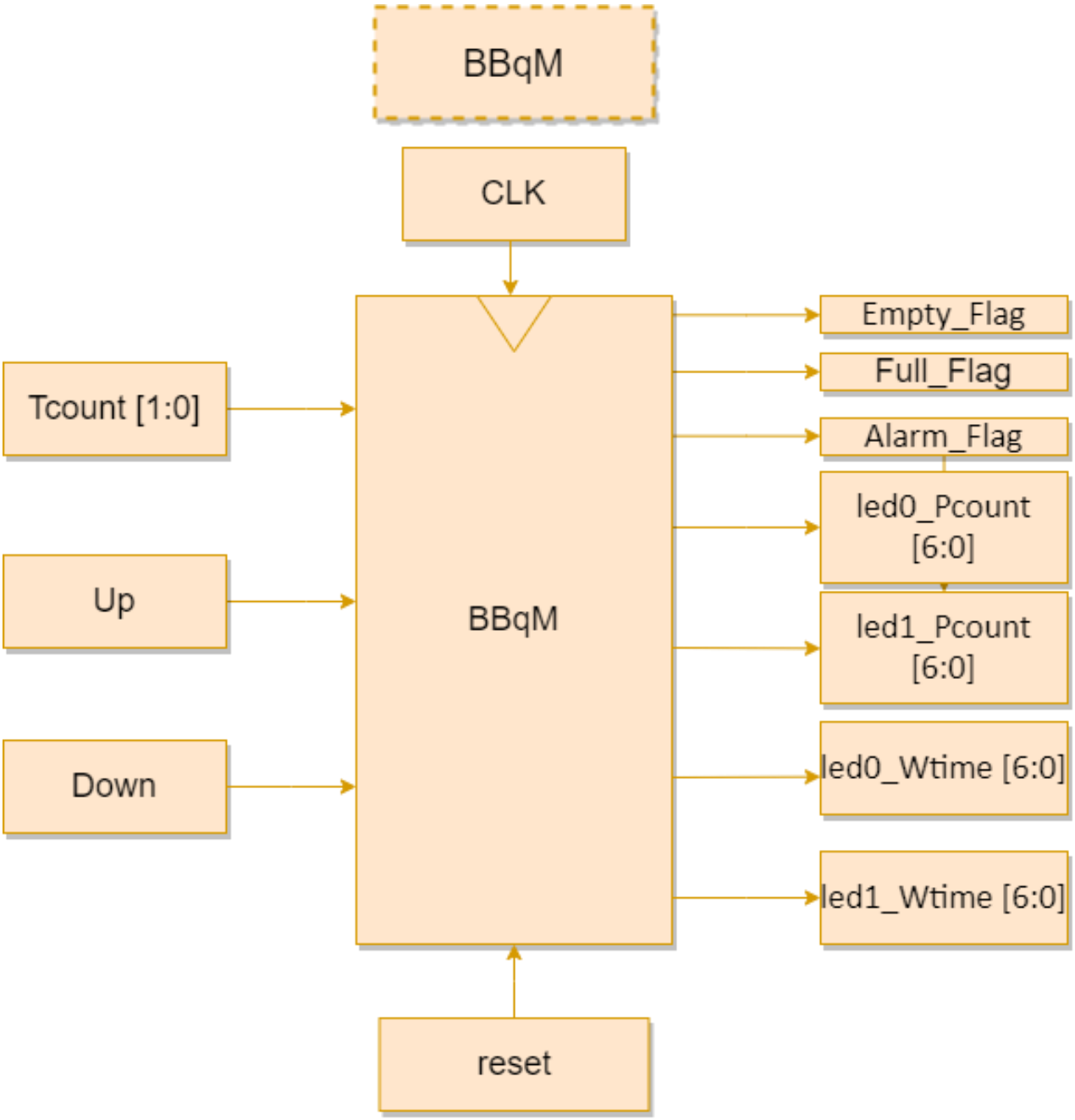
- **Outputs**

led0_Pcount [6:0]	The decoded Pcount to 7segment form
led1_Pcount [6:0]	The decoded 0 to 7segment form (give a good shape for Pcount)
led0_ Wtime [6:0]	The First Decoded digit of Wtime to 7segment form
led1_Wtime [6:0]	The Second Decoded digit of Wtime to 7segment form

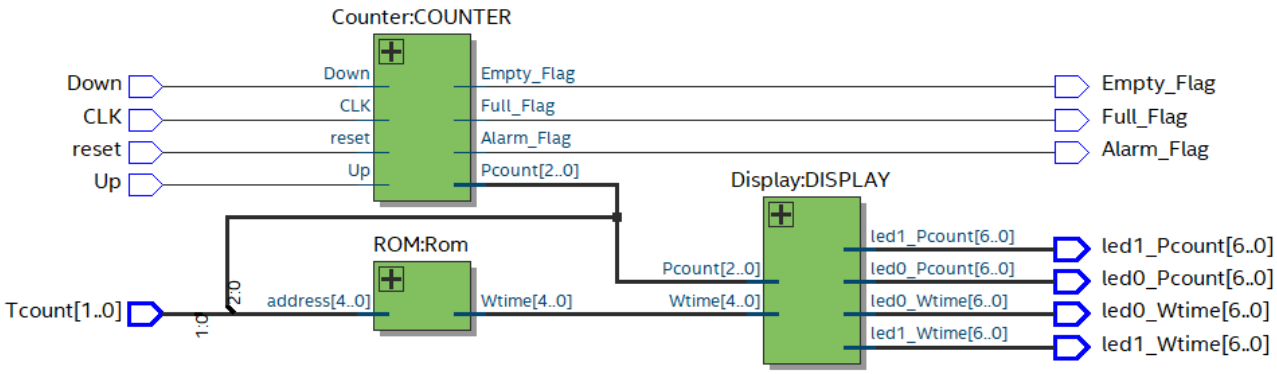
Top level Module (BBqM)

Finally, the top-level module that contain all the calls of functions needs and insertion of values, it arranges the work of every module and its functions and prevent any risks with the other modules, is the main module whose directly interact with the machine (FPGA), so it is consider the main chain between the other coded models and the machine (FPGA).

- **Block Diagram of BBqM.**



- **BBqM Structure**



- **Inputs**

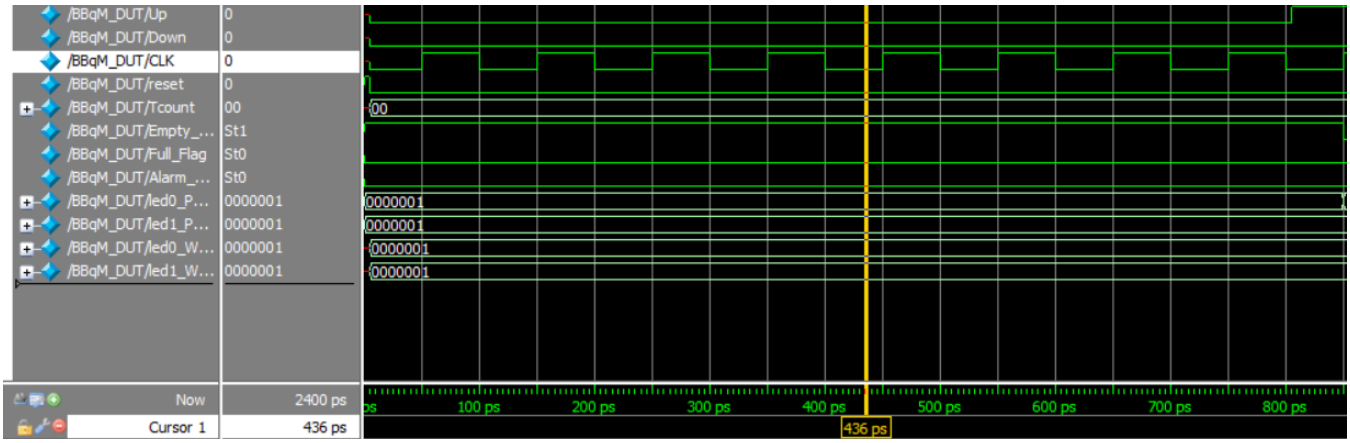
CLK	Clock which coming from FPGA (50 MHz)
reset	Reset switch which reset all the system to initial state
Up	Push button which increments Pcount (people in the Queue)
Down	Push button which decrements Pcount (people in the Queue)
Tcount [1:0]	Number of available tellers in 2 bit binary form

- **Outputs**

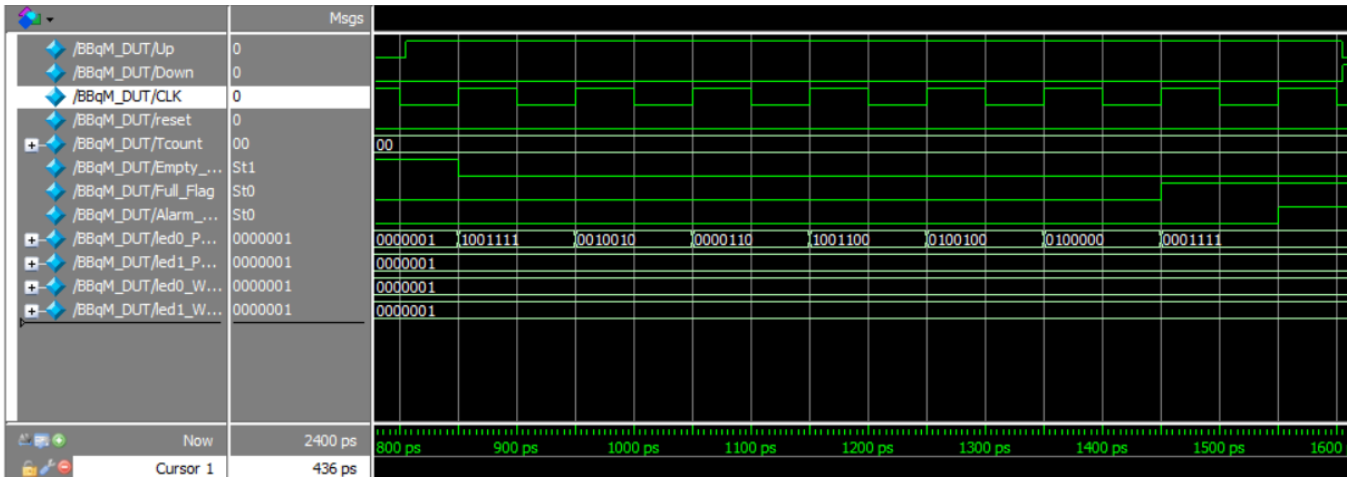
Empty_Flag	LED to show if the Queue is empty or not
Full_Flag	LED to show if the Queue is full or not
Alarm_Flag	LED to show the error states when the Queue is empty and Down Push button is pressed Or the Queue is full and Up Push button is pressed
led0_Pcount [6:0]	The decoded Pcount to 7segment form
led1_Pcount [6:0]	The decoded 0 to 7segment form (give a good shape for Pcount)
led0_ Wtime [6:0]	The First Decoded digit of Wtime to 7segment form
led1_Wtime [6:0]	The Second Decoded digit of Wtime to 7segment form

- **Testbench**

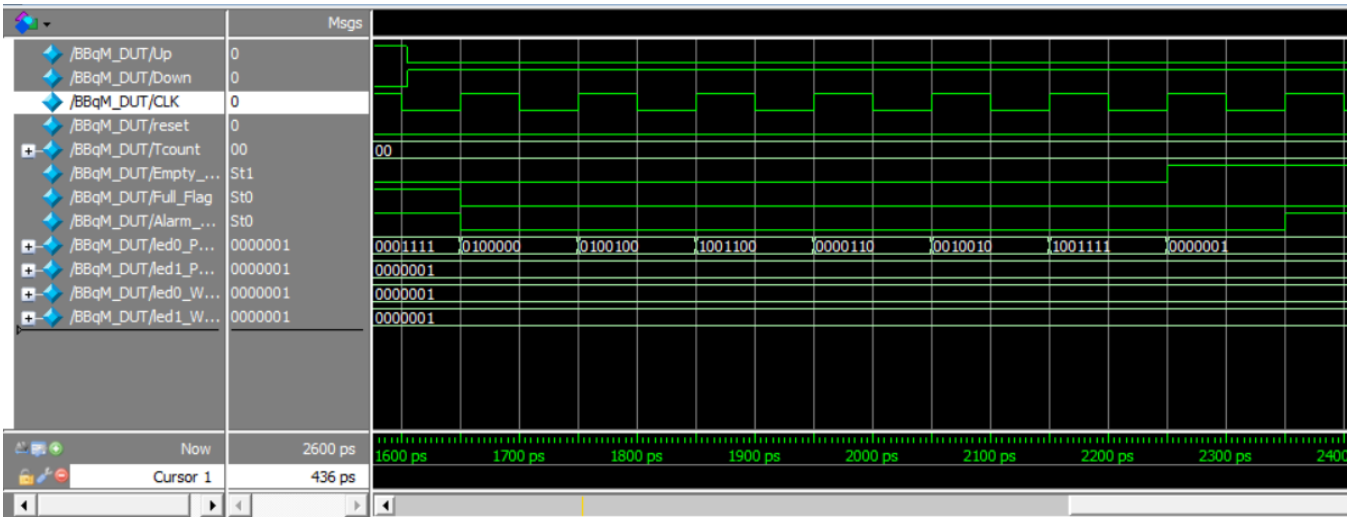
At Tcount = 0 , Up = 0 ,Down = 0 for 800ps (No tellers and No Pressing)



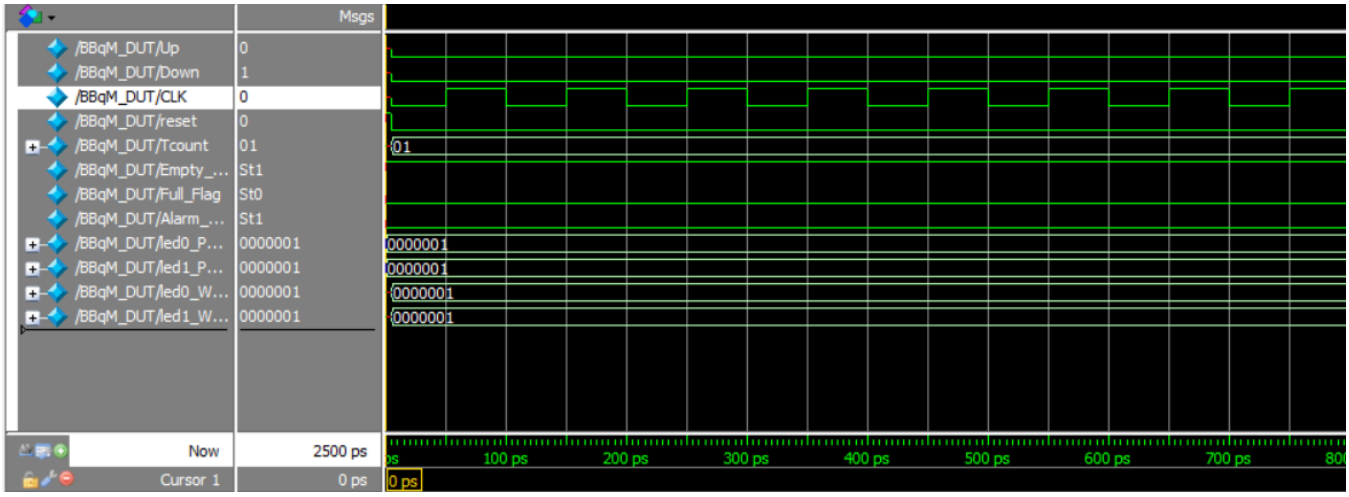
At Tcount = 0 , Up = 1 ,Down = 0 for 800ps (No tellers , Up pressed and Down not)



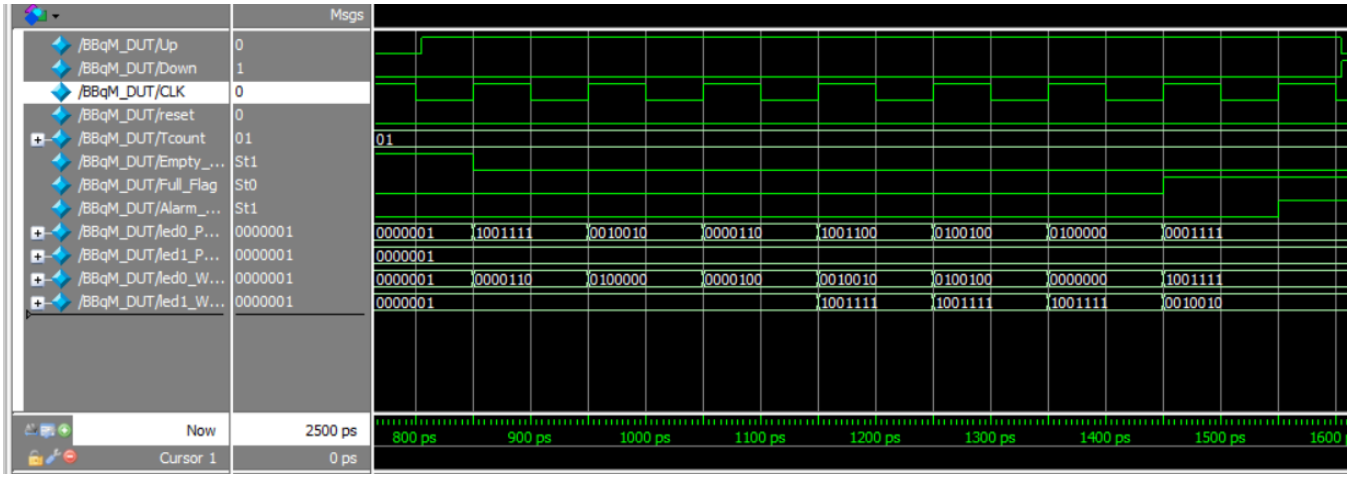
At Tcount = 0 , Up = 0, Down = 1 for 800ps (No tellers , Down pressed and Up not)



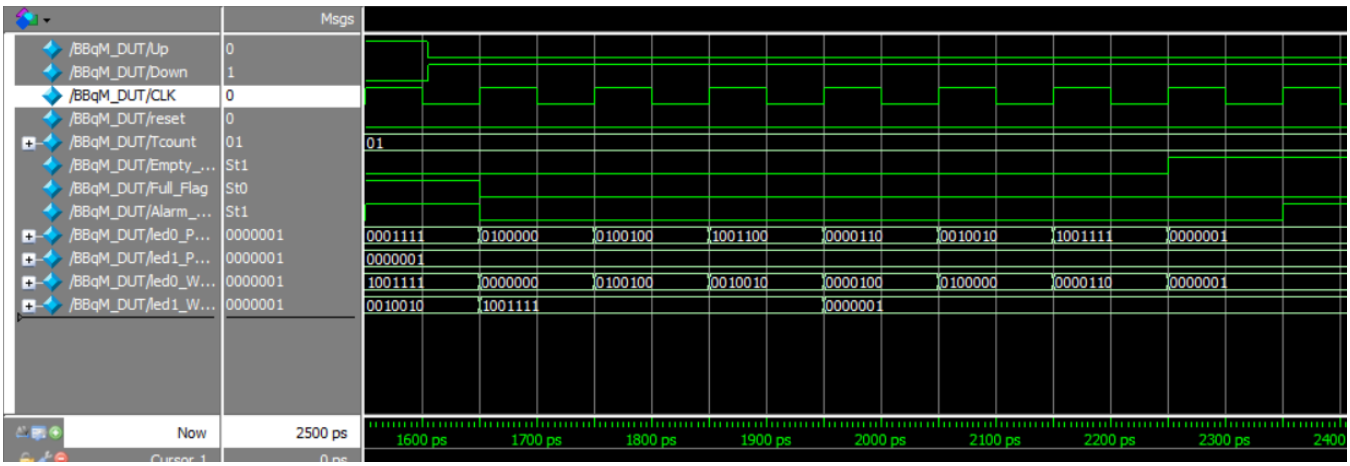
At Tcount = 1 , Up = 0 ,Down = 0 for 800ps (One teller and No Pressing)



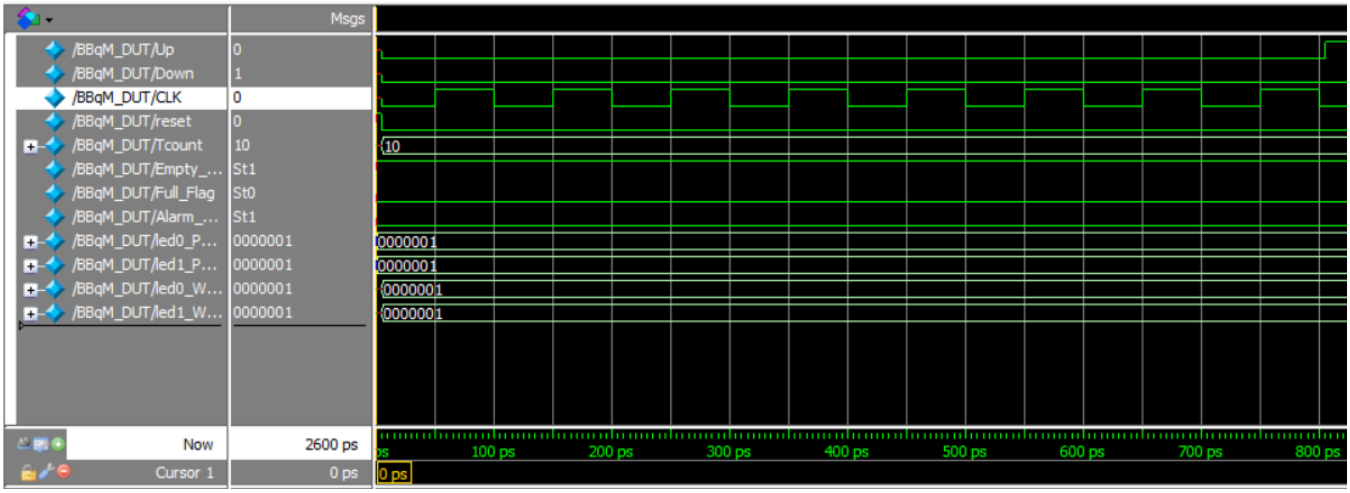
At Tcount = 1 , Up = 1 ,Down = 0 for 800ps (One teller , Up pressed and Down not)



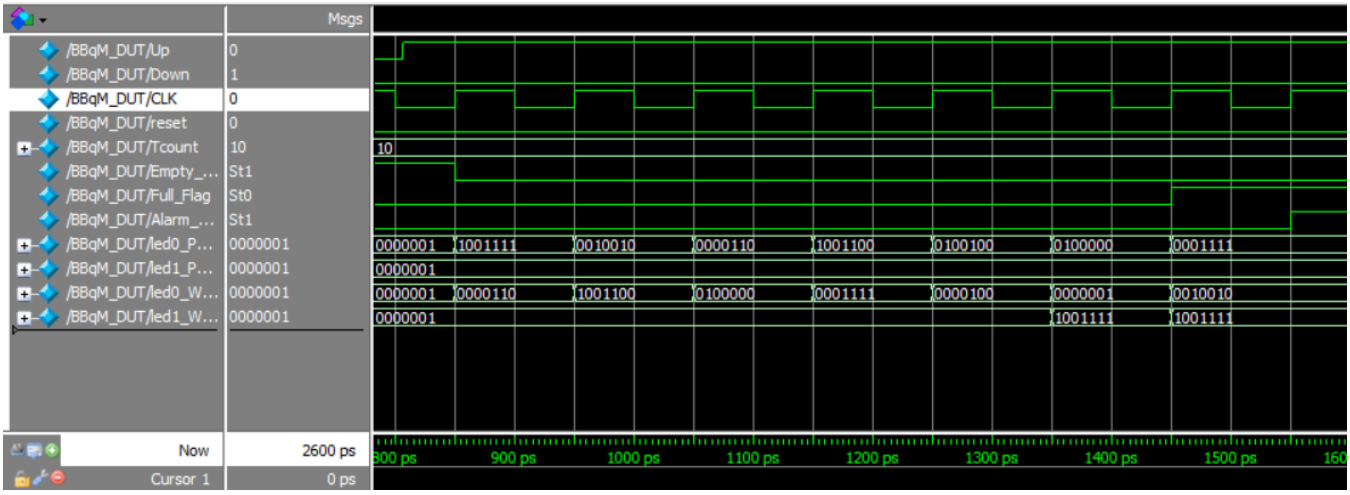
At Tcount = 1 , Up = 0, Down = 1 for 800ps (One teller ,Down pressed and Up not)



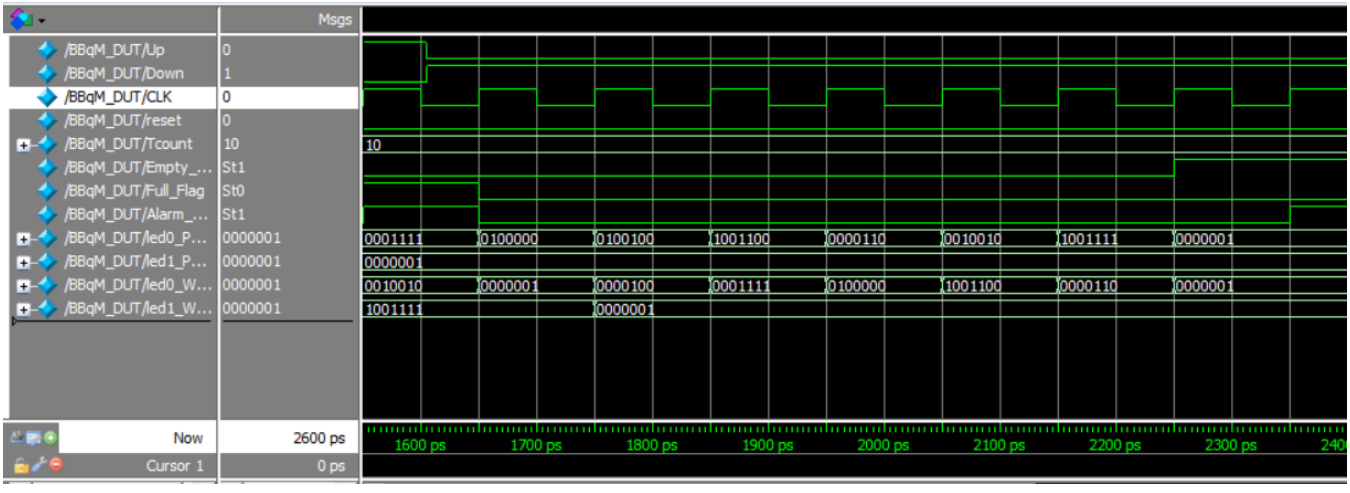
At Tcount = 2 , Up = 0 ,Down = 0 for 800ps (Two tellers and No Pressing)



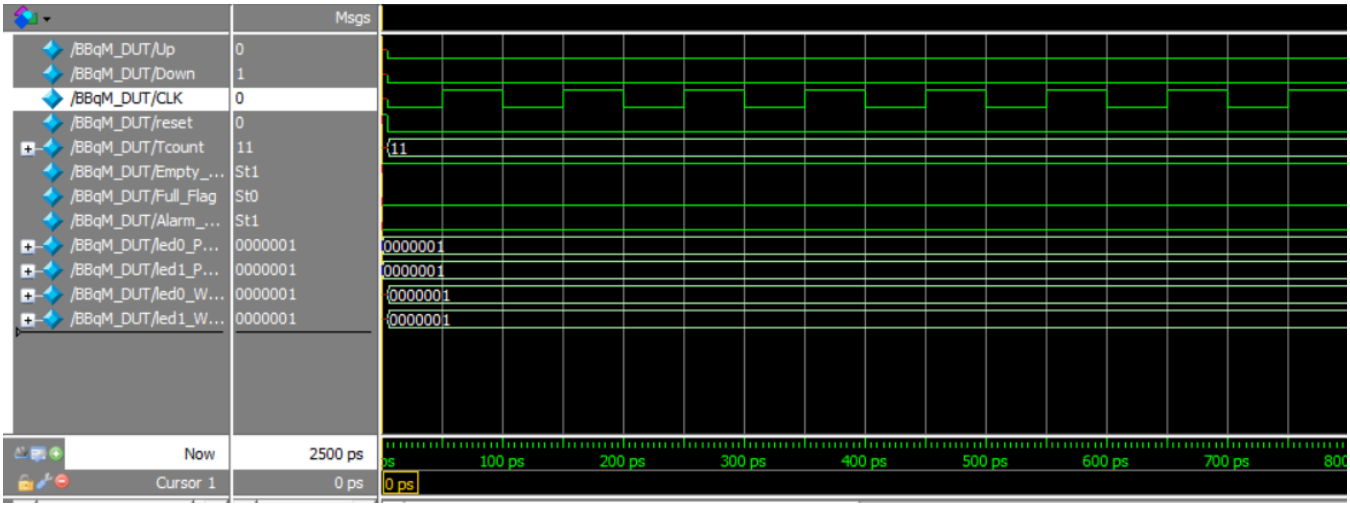
At Tcount = 2 , Up = 1 ,Down = 0 for 800ps (Two tellers , Up pressed and Down not)



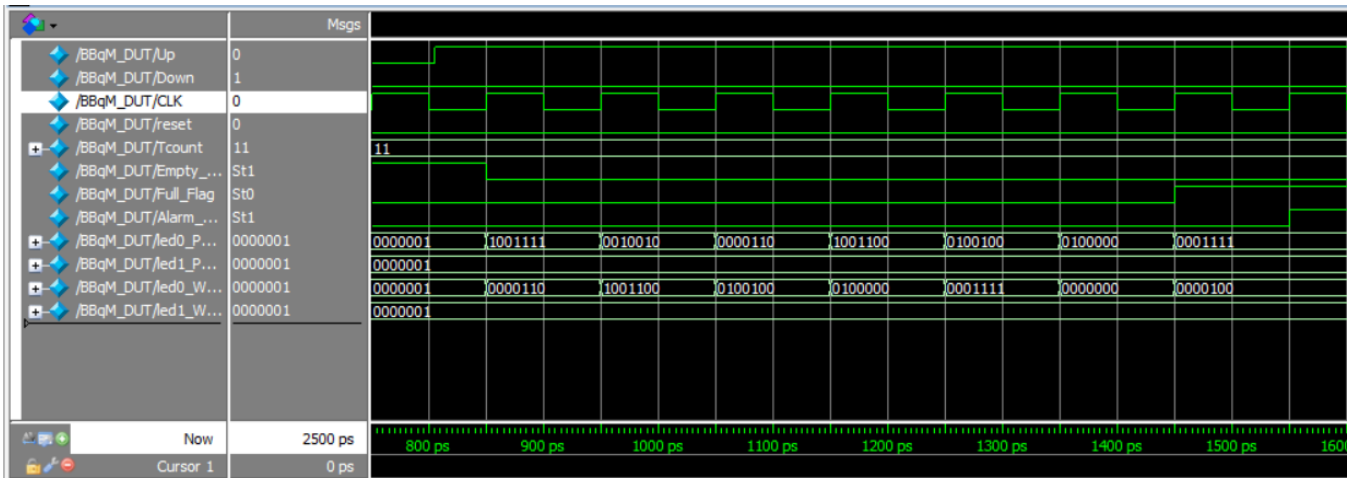
At Tcount = 2 , Up = 0, Down = 1 for 800ps (Two tellers ,Down pressed and Up not)



At Tcount = 3 , Up = 0 ,Down = 0 for 800ps (Three tellers and No Pressing)



At Tcount = 3 , Up = 1 ,Down = 0 for 800ps (Three tellers , Up pressed and Down not)



At Tcount = 3 , Up = 0, Down = 1 for 800ps (Three tellers ,Down pressed and Up not)