Mini-Project

The manager of the *Safebank* bank branch, located in a nearby mall, is proposing to install an embedded system to monitor the client queue in front of the tellers. The proposed system, called *SBqM™*, is to display various information about the status of the queue. Detailed specification of *SBqM™* is as follows:

- Both queue ends are equipped with a photocell. Each photocell generates a logic '1' signal if nobody interrupts a light beam generated at the corresponding queue end. When the light beam is interrupted, the photocell output changes to logic '0' and stays at that value until it is no longer interrupted.
- Clients are supposed to enter the queue only from the back end and leaves only from the front end.
- The number of people standing in the queue (waiting to be served by a teller), *Pcount*, and the expected waiting time in the queue before being served, *Wtime*, are to be displayed.
- *Pcount* is to be incremented by only one when a client enters the queue and is to be decremented by only one when a client leaves even if a client stands in front of the light beam for a long time period.
- *Wtime*, in seconds, could approximately be given by the formulas:

  $Wtime\ (Pcount = 0) = 0,$

  $Wtime\ (Pcount \neq 0, Tcount) = 3*(Pcount+Tcount-1)/Tcount$

  where *Tcount* is the number of tellers currently in service ($Tcount \in \{1,2,3\}$) and *Wtime* is rounded by ignoring the fraction part.

- *SBqM™* maintains binary *empty* and *full* flags that reflect the status of the queue.
- A responsible person should have the capability of resetting the system. Resetting the system clears the *full* flag and *Pcount*, and the sets *empty* flag.

The design team has met few times to decide on the following for the *SBqM*™ architecture:

- Although the system could be modeled as a one big FSM, it was decided to decompose the system into smaller FSMs. Design reuse is also recommended. That is, one small FSM model can be repeated multiple times in the overall system model.
- An n-bit up-down counter is to be used to generate *Pcount*. The maximum *Pcount* value will be ($\mathbf{2^n - 1}$), with a default value of 7, where *n* is a generic value, with a default value of 3.
- There will be an internal system clock.
- Although random logic can be used in calculating *Wtime*, it was decided to use a look-up table to achieve a better runtime performance. The lookup table is to be realized as a ROM.

In this project, you are going to model the operation of *SBqM™* and verify it via simulation. Then, you will synthesis the model. Here is the list of deliverables:

a) In a table, identify *SBqM™* inputs and outputs and briefly describe their meaning and possible values.
b) Draw an icon for the *SBqM™*, clearly showing its input and output signals.
c) Draw a block diagram showing the *SBqM™* structure.
d) Draw the necessary FSM diagram(s).
e) Model the up-down counter as a separate component using a Verilog behavioral description.
f) Model the required FSM(s) in Verilog.
g) Model *SBqM™* using a mixed architecture Verilog code. Use concurrent (dataflow) descriptions for the binary flags. The model should issue a warning message using (Assert/Report/Severity) if the queue is already full/empty and somebody tries to enter/leave.
h) In a table, propose a test strategy to verify the operation of the SBqM™ model. Carefully select an appropriate set of test cases that tests various aspects of the design. For example, you should test that the two flags reflect the correct status of the queue. Additionally, you should ensure that the counter does not wrap around. That is, it does not display '0' when it gets an increment signal while the queue is full or '7' when it gets a decrement signal while the queue is empty.
i) Design a testbench to verify the operation of the *SBqM™* model using the test strategy proposed in the previous point. Provide simulation result snapshots.
j) Synthesize the *SBqM™* architecture in part (g). Include a diagram of the sythesize output with your submision.

Here are some hints:

- Try to decompose the SBqM™ architecture into a number of small procedures. Each procedure should be responsible for updating some of the output signals. Avoid having multiple procedures updating one single signal.
- The ROM could be declared as a one-dimension array of integers holding the *Wtime* values. The index into this array could be constructed by concatenating *Tcount* and *Pcount*. For example, for *Pcount* = 3 and *Tcount* = 2, the array index in binary will be "10011". '&' is the concatenation operator in Verilog.
- Make sure that your Verilog code initializes the ROM only once.