專案(三)

專案名稱 - Approximate Average

簡介

Please design a computational system whose transfer function is defined as follow. A series of 8-bit positive integer is generated as the input of the computational system by the test bench. The output value Y is a 10-bit positive integer, which is calculated according to equations (1), (2), (3) and (4).

$$Xavg_j = \left\lfloor \frac{\sum_{i=j}^{j+n-1} X_j}{n} \right\rfloor \tag{1}$$

$$XS = \{X_j, X_{j+1}, X_{j+2}, \ldots, X_{j+n-1}\} \tag{2}$$

$$X_{appr_j} = \begin{cases} Xappr_j = Xavg_j, & if\ Xavg_j \in XS \\ X_i \mid (X_i \in XS) and \left(X_i < Xavg_j\right) and \left(Xavg_j - X_i\ is\ minimal\right), & if\ Xavg_j \notin XS \end{cases} \tag{3}$$

$$Y_j = \left\lfloor \frac{\sum_{i=j}^{j+n-1} X_i + Xappr_j}{n-1} \right\rfloor \tag{4}$$

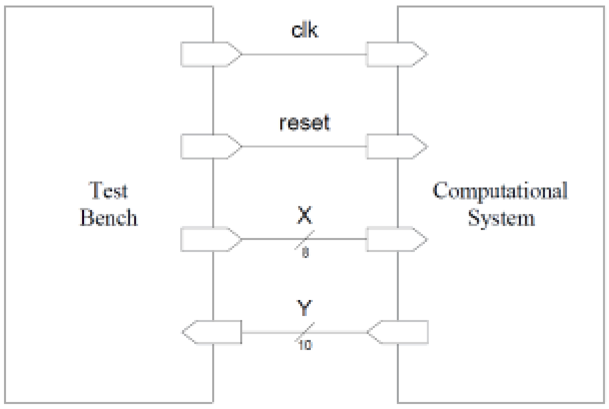
The computational system produces the output sequence according to the given input sequence. Each input and output data in the respective sequence is indexed. This index, in terms of hardware, is the relative time when the input data is given or the output data is ready. Thinking as a hardware designer, the approximate average is chosen from the last *n* input data which should be stored in the system. The system should be able to calculate the integral part of the real average of the last *n* input data first. The rules of calculation is detailed in the following. If integral part of the real average equals to any one of the last n input data, the approximate average is simply the integral part. Otherwise, the approximate average is the one which is one of the last n input data whose value is smaller than and closest to the integral part of the real average. The above descriptions stated the desired operations as those defined by equations (1), (2), and (3).

After the approximate average is obtained, the output value can be calculated according to equation (4). First, the last n input value is added by the corresponding approximate average. Then they are summed up and divided by n-1. The output value is the quotient after division.

For example, assume that n=4, X1=3, X2=24, X3=16, X4=8, and X5=3. After the first 5 input items are given, the system should store them and calculate the output value. The average of the first 4 input values is 12(only shows the quotient). Since it is not in the set of {X1, X2, X3, X4}, the system selects one from {X1, X2, X3, X4} as the approximate average whose value is smaller than 12 and close to 12. In this case, the approximate average is 8. So the first output value is calculated n as $\lfloor[(3 + 8) + (24 + 8) + (16 + 8) + (8 + 8)] /[(4 - 1)]\rfloor = 27$. Similar to those described above, when the 5th input value is given, the system should store X2, X3, X4 and X5 and calculate the corresponding output value. The 2nd output value should be the same as the first one because the values stored in the system is the same.

設計規格

Block overview

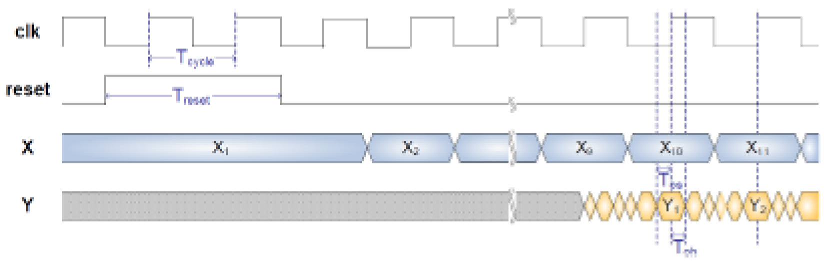


圖一、系統方塊圖

I/O Interface

Name	I/O	Width	Description
clk	I	1	clock for the computational system.
reset	I	1	reset the state of the computational system when it asserts.
X	I	8	input data of the computational system
Y	O	10	computed output

Function Description



圖一、時序圖

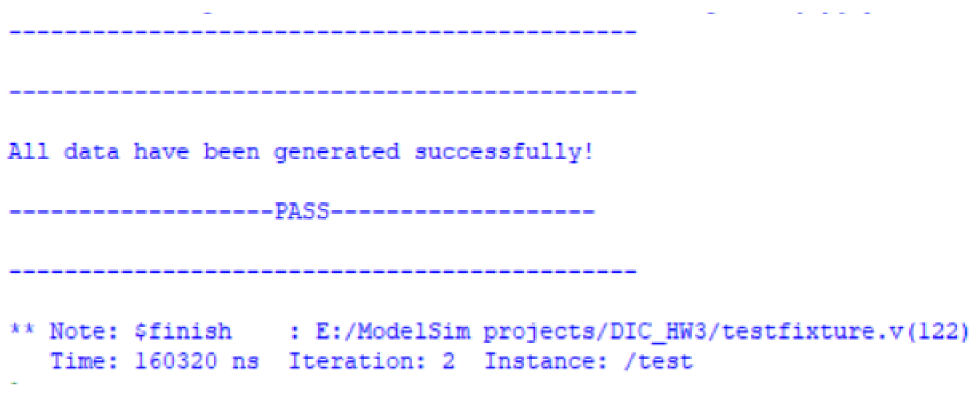
Symbol	Description	Value
T_{cycle}	clock period	user defined
T_{reset}	reset pulse width	2 T_{cycle}
T_{os}	setup time from valid output to positive edge of *clk*	0.5ns
T_{oh}	hold time from positive edge of *clk* to invalid output	0.5ns

圖二、Symbol

The I/O timing diagram is as shown above. For this homework, n in equations (1)-(4) are fixed to 9. The computational system is reset by asserting reset signal for 2 periods. The input X is changed to the next at the negative edges of the clock while output Y is checked by the test bench at positive edges of the clock. Note that the output should be stable around the positive edges of the clock. The setup and hold time requirements for the output are listed in Table. The first output data should be valid after the input data changes from 9th one to 10th and before the next positive clock edge. After that, the output should be changed to the next at the next positive clock edge and so on, that is to say, the test bench checks one output value per clock cycle.

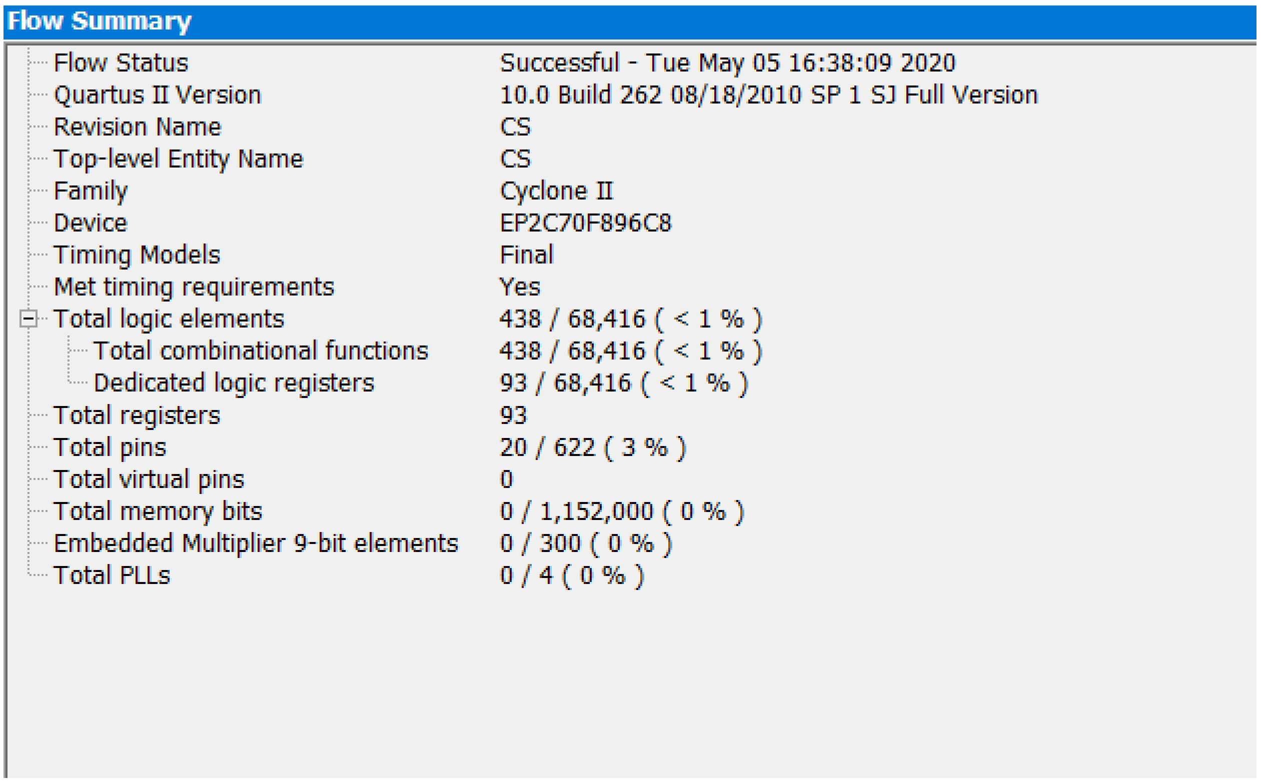
Result

Gate-Level simulation



```
---------------------------------------------

---------------------------------------------

All data have been generated successfully!

-------------------PASS-------------------

---------------------------------------------

** Note: $finish    : E:/ModelSim projects/DIC_HW3/testfixture.v(122)
   Time: 160320 ns  Iteration: 2  Instance: /test
```

Synthesis result



Flow Summary

Flow Status	Successful - Tue May 05 16:38:09 2020
Quartus II Version	10.0 Build 262 08/18/2010 SP 1 SJ Full Version
Revision Name	CS
Top-level Entity Name	CS
Family	Cyclone II
Device	EP2C70F896C8
Timing Models	Final
Met timing requirements	Yes
Total logic elements	438 / 68,416 (< 1 %)
Total combinational functions	438 / 68,416 (< 1 %)
Dedicated logic registers	93 / 68,416 (< 1 %)
Total registers	93
Total pins	20 / 622 (3 %)
Total virtual pins	0
Total memory bits	0 / 1,152,000 (0 %)
Embedded Multiplier 9-bit elements	0 / 300 (0 %)
Total PLLs	0 / 4 (0 %)