Worcester Polytechnic Institute

ECE 5723: Methodologies for System Level Design and Modeling

C++ RT-level Design and Modeling

by

Vladimir Vakhter

Laboratory Report

10/05/2020

Table of Contents

1. Introduction

In this work, a circuit that counts the number of transitions that occur on its serial input is designed (Fig. 1).

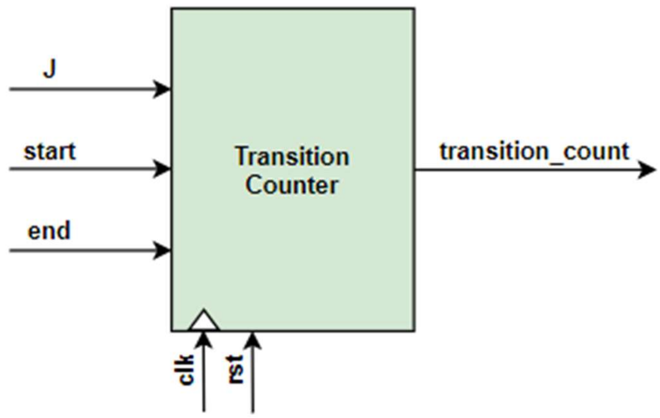

Fig. 1 – Transition Counter

The above circuit is a clocked circuit with a serial input, ***J***. There is a ***start*** pulse and an ***end*** pulse. After the ***start*** pulse (0-1-0, guaranteed one pulse), the counting begins and continues until a pulse is detected on ***end***. During this time, clocked-synchronous transitions are detected and counted on the ***J*** input. An 8-bit counter is used here and counting beyond 256 rolls over back to 0. Provide an asynchronous reset.

2. Analysis of the Task

Based on the information given in the introduction, the graph shown in **Fig. *2*** for the waveforms was derived:

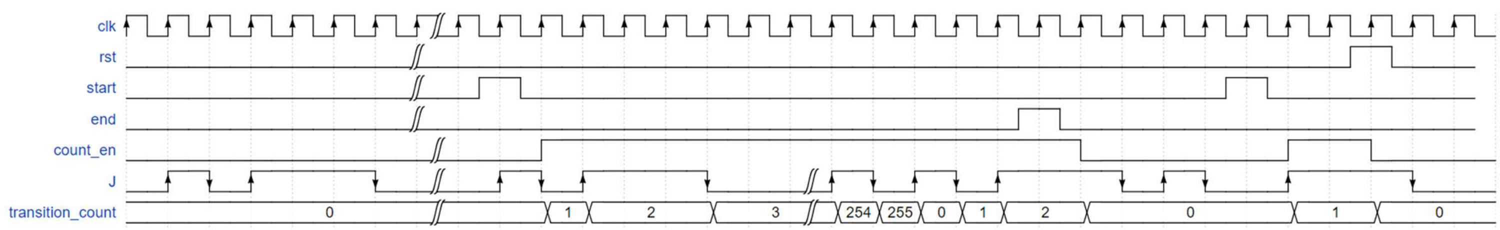

Fig. 2 – The waveforms demonstrate the operation of transition counter

3. Schematic Diagram of the Datapath

In this chapter, the schematic diagram of the data path of the designed circuit is presented in **Fig. *3***.

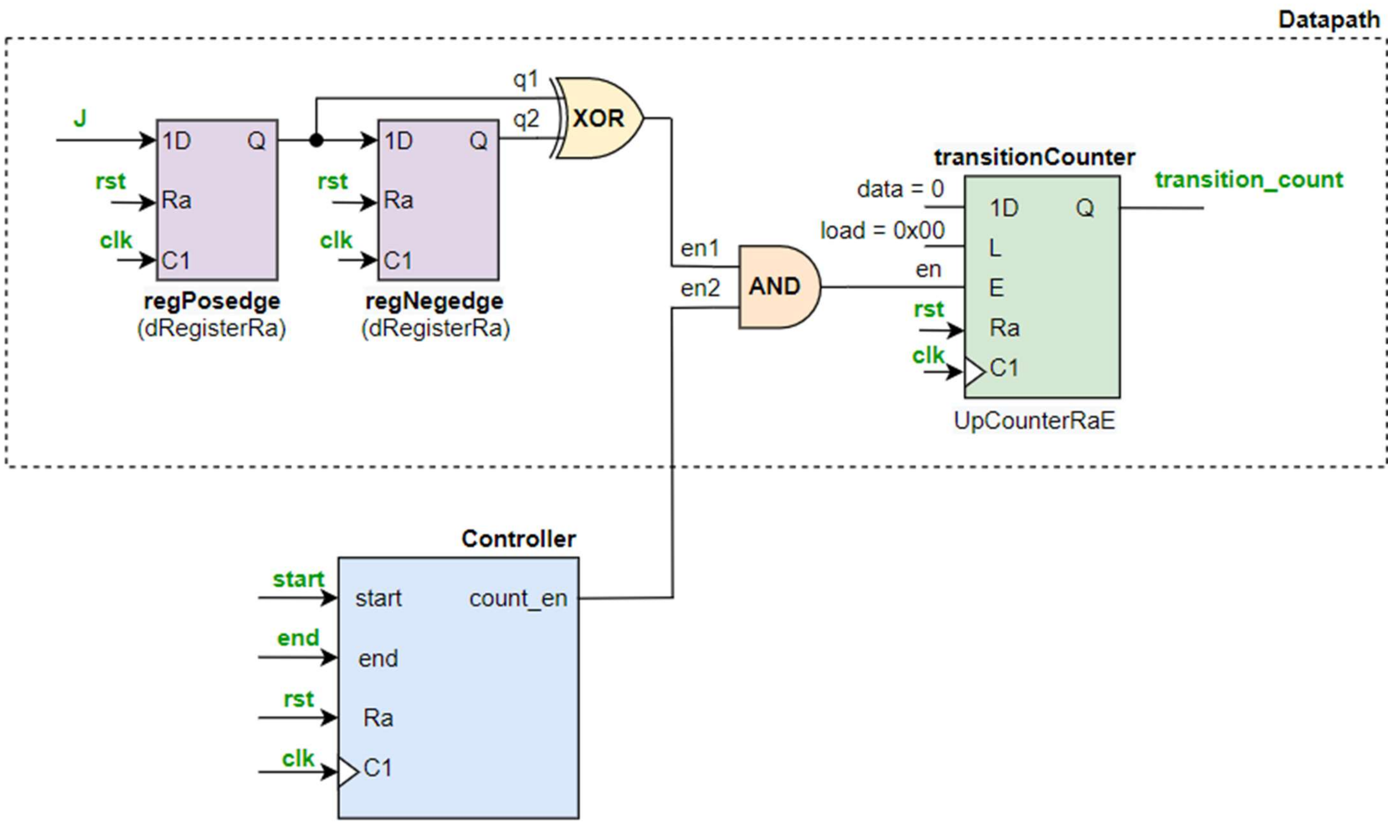

Fig. 3 – The schematic diagram of the datapath of transition counter

4. Controller State Diagram

In this chapter, the controller state diagram of the designed circuit is shown in **Fig. *4***.

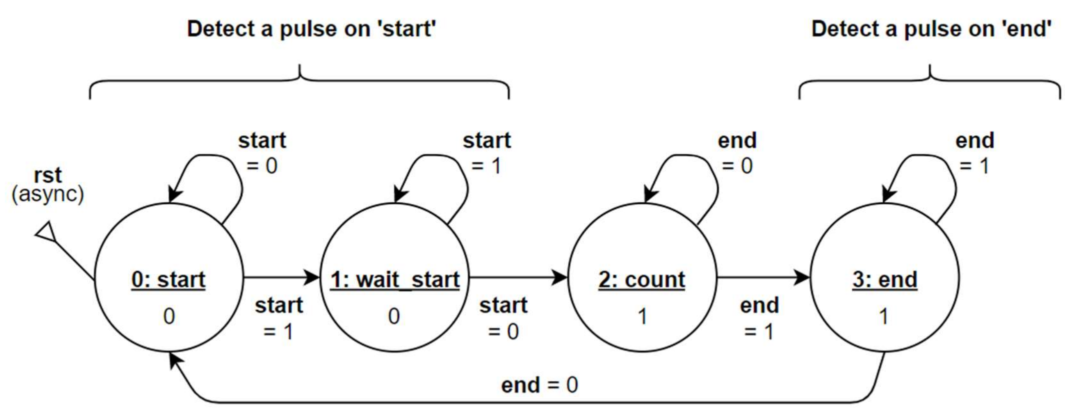

Fig. 4 – The state diagram of the controller of transition counter

5. Schematic Development in C++

The Huffman model of the design was implemented in C++ using the bus library discussed in class. The controller (CU) and the datapath (DP) of the circuit were built as separate modules. Then, these modules were wired in a top-level module class. The evl() function of this class ensures a proper order of the evl() functions of the contained components. The complete design was instantiated inside the main() function.

6. Testbench

In this chapter, the results of the verification of designed circuit are provided for four different scenarios.

6.1. Scenario 1

In this test scenario, there are no pulses neither on 'start', nor on 'end'. J changes from 0 to 1. Output: 0.

```
Test case 1 begins...
start = 0; end = 0; J changes 0->1. Expected output = 0x00.
Transition count: 00000000
```

6.2. Scenario 2

In this test scenario, a pulse on start is followed by two transitions on J. Output: 2.

```
Test case 2 begins...
pulse on start; end = 0; J changes 0->1, 1->0. Expected output = 0x02.
Transition count: 00000010
```

6.3. Scenario 3

In this test scenario, a pulse on start is followed by two transitions on J. Then, a pulse on 'end' occurs. Then J changes its value twice. Output: 2.

```
Test case 3 begins...
pulse on start; end = 0; J changes 0->1, 1->0; pulse on end; J changes 0->1, 1->0. Expected output = 0x02.
Transition count: 00000010
```

6.4. Scenario 4

In this test scenario, a pulse on start is followed by 256 transitions on J. Output: rolls back to 0 after 255 transitions.

```
Test case 4 begins...
pulse on start; end = 0; J changes 0->1, 1->0 - overall 256 times. Expected output = 0xFF on 255, 0x00 on 256.
Transition count, J changed 255 times: 11111111
Transition count, J changed 256 times: 00000000
```

7. Conclusions

In this work, a circuit that counts the number of transitions that occur on its serial input was designed and tested in C++.