嵌入式系统结构与设计课程实验

实验报告 6

Traffic Controller with Moore and Mealy FSMs

邱迪聪 / 11331262

2013/11/13

Content

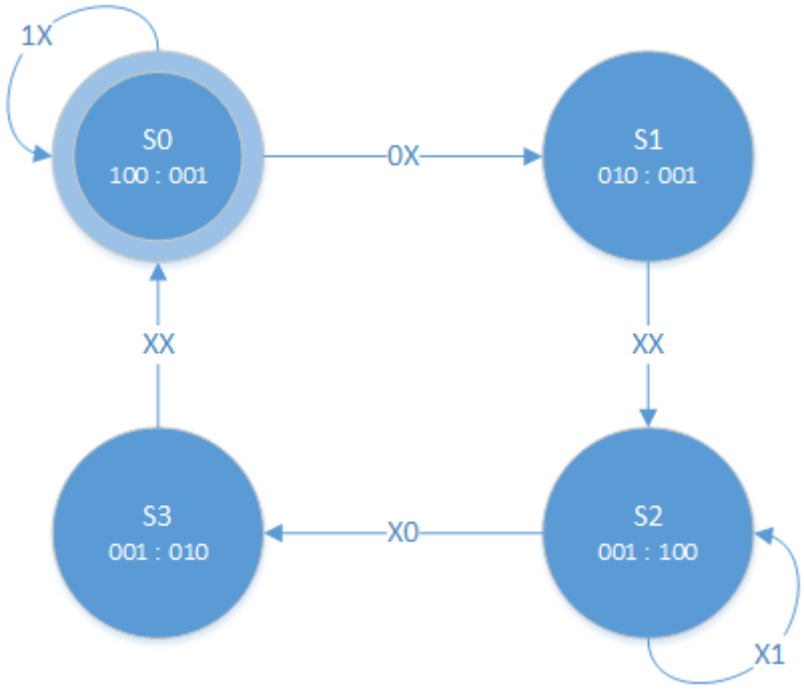
1. Target

Understand the High level design for FSM.

2. Traffic Controller with Moore FSM

2.1 Designing

To design the traffic controller with Moore finite state machine described as the textbook, the state transition diagram is necessary. And the diagram is as below.

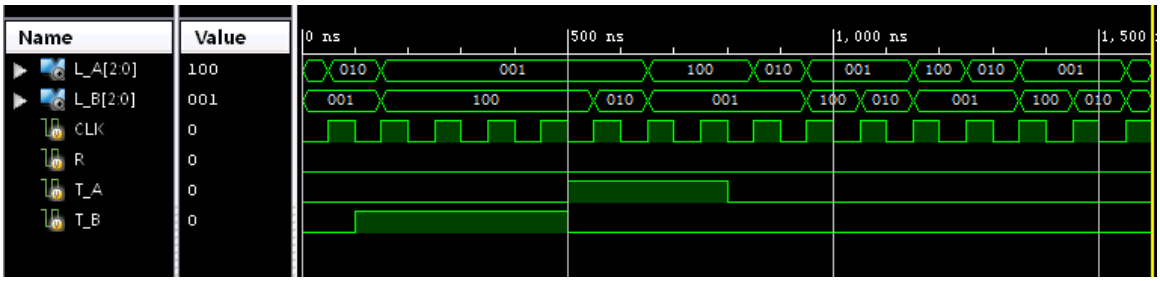


The state is represented by a size-of-two array; the traffic lights are represented by 3 bits for each one, representing Green, Yellow and Red, respectively.

For a Moore finite state machine, the output is only determined by the current state of the FSM. Hence all the inputs and the outputs are designed as a `wire` type. The output L_A and L_B is actually a combinational logic with the input as the state, which is an inner register array.

2.2 Simulation

Several different cases are picked up to be the simulation cases, and the simulation result is as follow.



2.3 Synthesization, Implementation and Program Generation

The implementation constraints file is as followed:

```
// Input: T_A, T_B
NET "T_A" LOC = "T5"; // left most button
NET "T_B" LOC = "V8";

// Input: R
NET "R" LOC = "B8"; // central button

// Clock: CLK_100M
NET "CLK_100M" LOC = "V10"; // 100MHz clock

// Output: L_A
NET "L_A[2]" LOC = "T11"; // left most LED
NET "L_A[1]" LOC = "R11";
NET "L_A[0]" LOC = "N11";

// Output: L_B
NET "L_B[2]" LOC = "M11";
NET "L_B[1]" LOC = "V15";
NET "L_B[0]" LOC = "U15";
```

The `V10` pin is a 100MHz user clock of the Nexys 3 board. The clock pulse will be counted in the `run` module, and only after 5 seconds will the FSM be refreshed.

The RTL diagram of the `run` module generated from Synthesization is as below:

(See Appendix 2: RTL Diagrams)

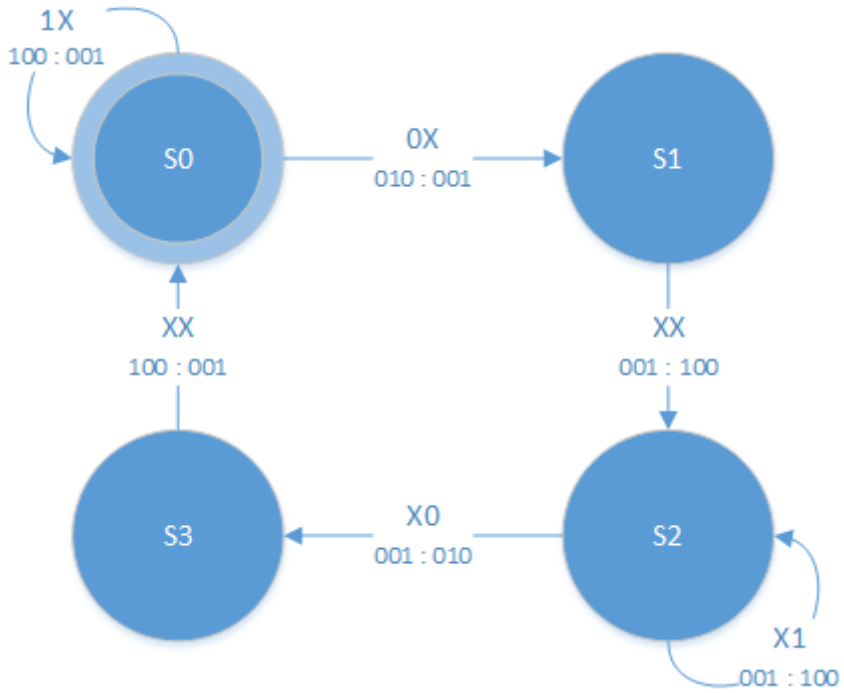
Below is the RTL diagram of the core control module, the `TC_Moore` module.

(See Appendix 2: RTL Diagrams)

3. Traffic Controller with Mealy FSM

3.1 Designing

To design a traffic controller with Mealy finite state machine, the state transition diagram is necessary. And the diagram is as below.

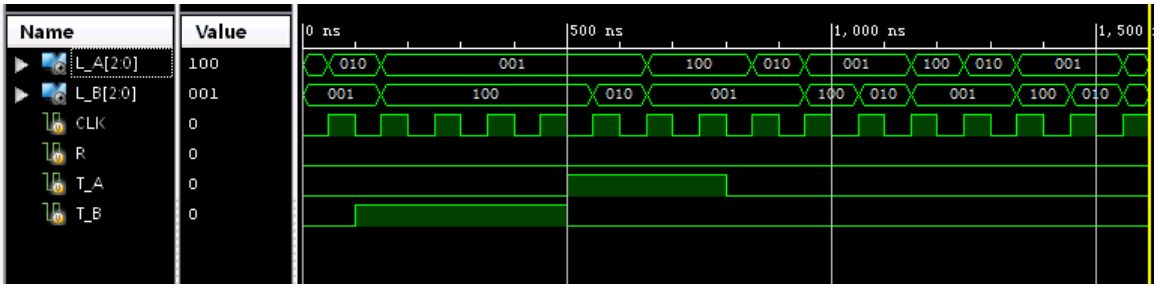


The state is represented by a size-of-two array; the traffic lights are represented by 3 bits for each one, representing Green, Yellow and Red, respectively.

For a Mealy finite state machine, the output is determined by the current state of the FSM and the input. Hence all the outputs are designed as a `reg` type in convenience of display. The output L_A and L_B are two register outputs, and their values will be changed only if a reset signal is triggered or a new clock pulse comes.

3.2 Simulation

Several different cases are picked up to be the simulation cases, and the simulation result is as follow.



3.3 Synthesization, Implementation and Program Generation

The implementation constraints file is as followed:

```
// Input: T_A, T_B
NET "T_A" LOC = "T5"; // left most button
NET "T_B" LOC = "V8";

// Input: R
NET "R" LOC = "B8"; // central button

// Clock: CLK_100M
NET "CLK_100M" LOC = "V10"; // 100MHz clock

// Output: L_A
NET "L_A[2]" LOC = "T11"; // left most LED
NET "L_A[1]" LOC = "R11";
NET "L_A[0]" LOC = "N11";

// Output: L_B
NET "L_B[2]" LOC = "M11";
NET "L_B[1]" LOC = "V15";
NET "L_B[0]" LOC = "U15";
```

The `V10` pin is a 100MHz user clock of the Nexys 3 board. The clock pulse will be counted in the `run` module, and only after 5 seconds will the FSM be refreshed.

The RTL diagram of the `run` module generated from Synthesization is as below:

(See Appendix 2: RTL Diagrams)

Below is the RTL diagram of the core control module, the `TC_Mealy` module.

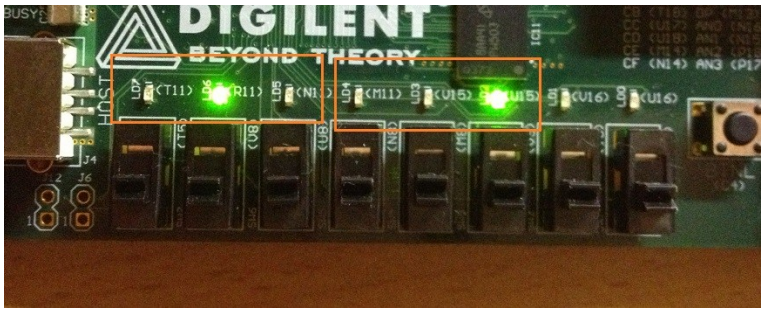
(See Appendix 2: RTL Diagrams)

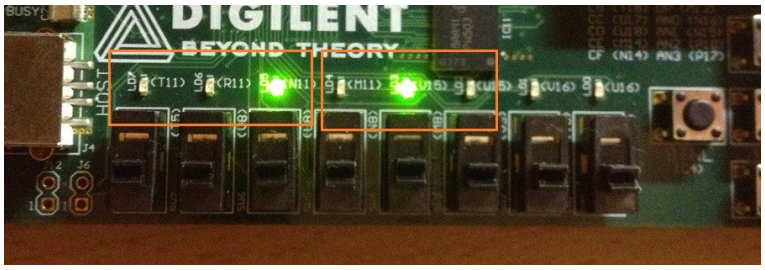
The following steps were successfully gone through and a .bit file was generated successfully.

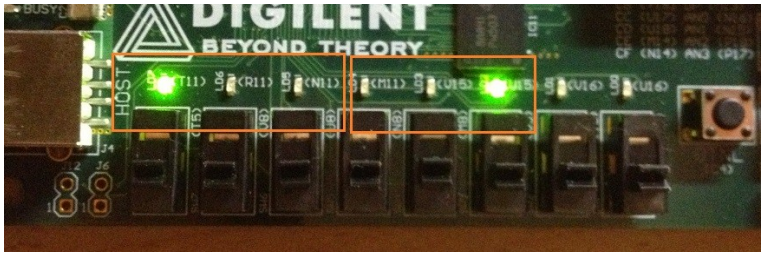
4. Online Testing

The results of online testing are exactly the same from Moore FSM design and Mealy FSM design, thus only one complete round and two holding states of the Moore FSM design will be shown on below. And the test on the other FSM design is not to be shown below.

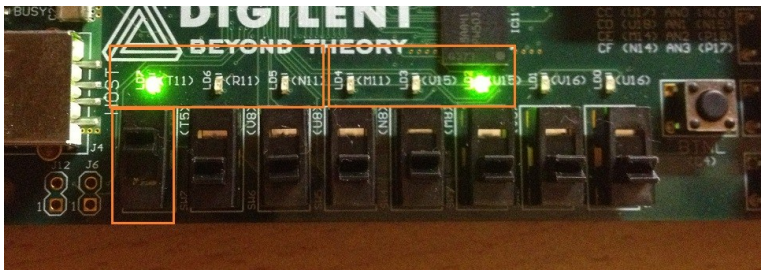
No Traffic Jam:



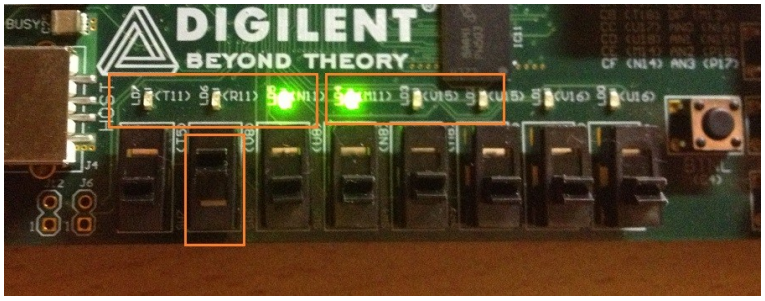




Traffic Jam on path A:



Traffic Jam on path B:



The test result matched expectation.

Afterthought

Finite State Machine (FSM) is a great talk in computer science. I've had this topic in three courses of mine, the Digital Circuit Designing Course, the Compiling Principles Course and this course, the Embedded System and Architecture Course.

As far as I am concerned, the FSM I designed above is between the other two courses, which means it is more like what I did on the Digital Circuit Designing Course but much easier to design one, and meanwhile, it is not so complex and automatic as what I've learnt from the Compiling Principles Course.

FSM is a logic that holds states, and can represent a fixed sequence. The traffic light is a very good example of the application of the finite state machine. And if we want to make advance use of it, compiler will be a great useful application, but it is quite complex as implementation. On an embedded system, I suppose that stream data processing, such as encryption, photo or video transformation, and so on, will see giant demand of such technique.

For such a reason, I am glad to have learnt this idea, and I am sure there must be still a lot of application for finite state machine to be discovered.

David Qiu (邱迪聪)

2013.11.13

Appendix 1: Attachment List

1. TC_Moore

    \ run.v

    \ TC_Moore.v

    \ TC_Moore _ctr.ucf

    \ TC_Moore _test.v

2. TC_Mealy

    \ run.v

    \ TC_Mealy.v

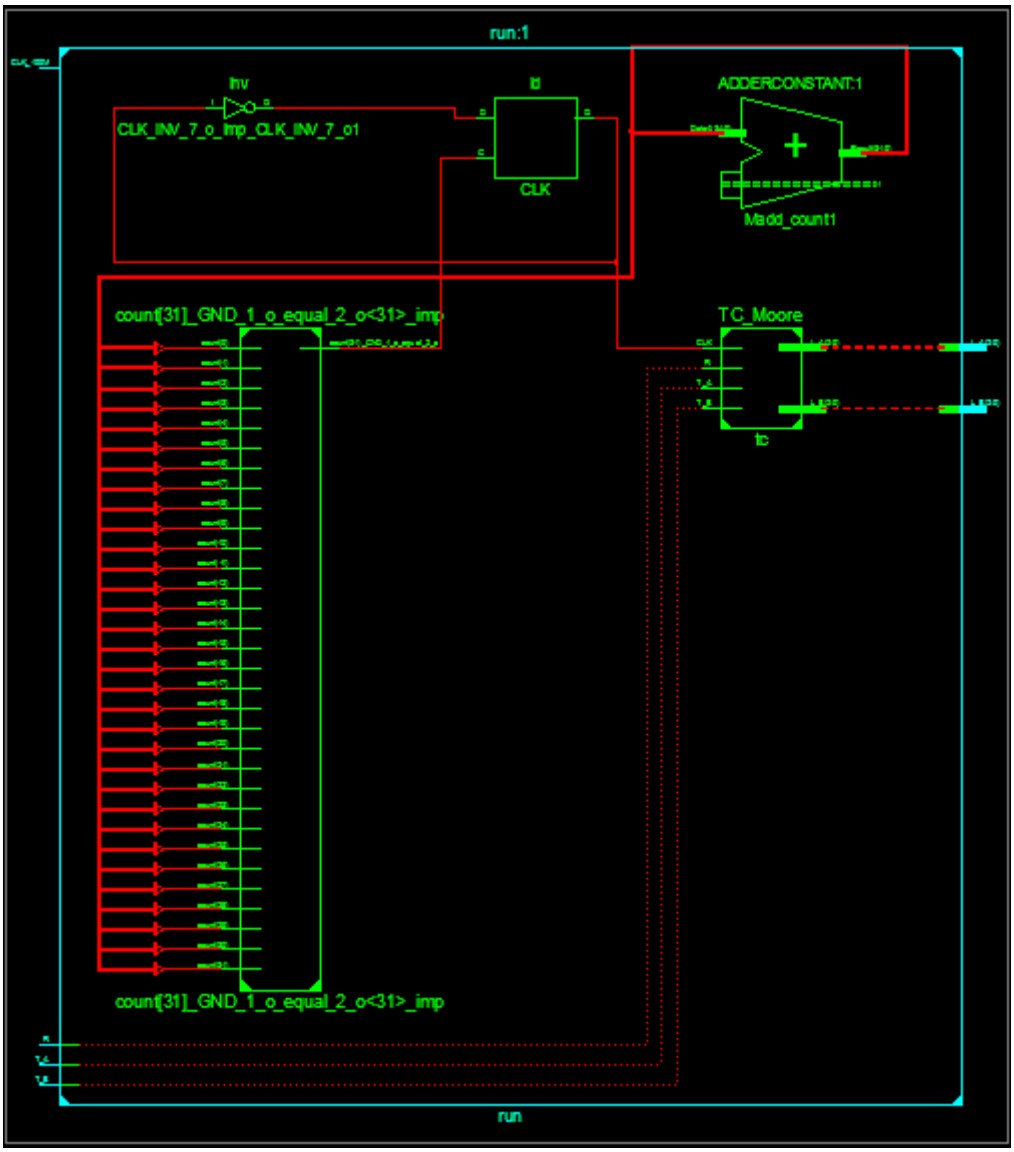
    \ TC_Mealy _ctr.ucf

    \ TC_Mealy _test.v

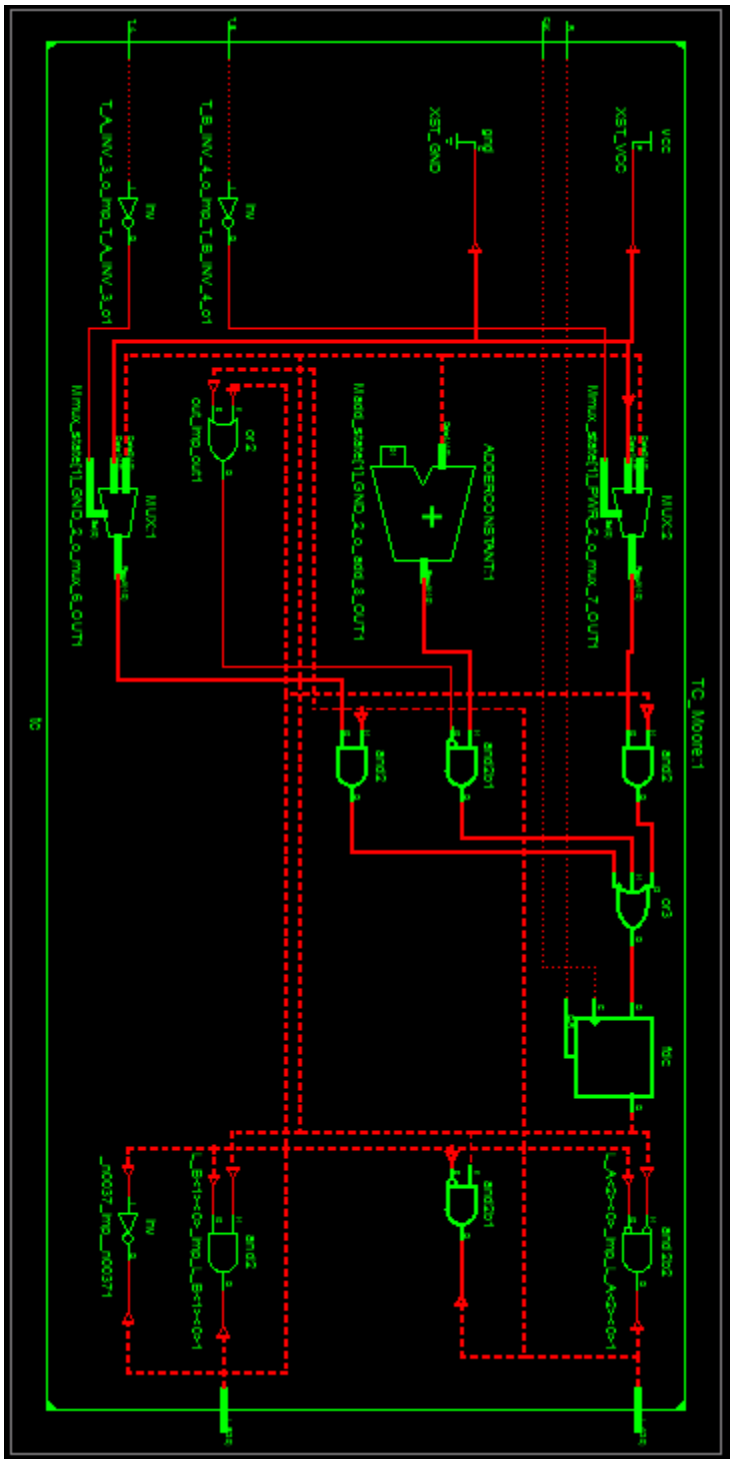
X.v files are primary code files; X_ctr.ucf files are implementation constraints files; X_test.v files are test files.

Appendix 2: RTL Diagrams

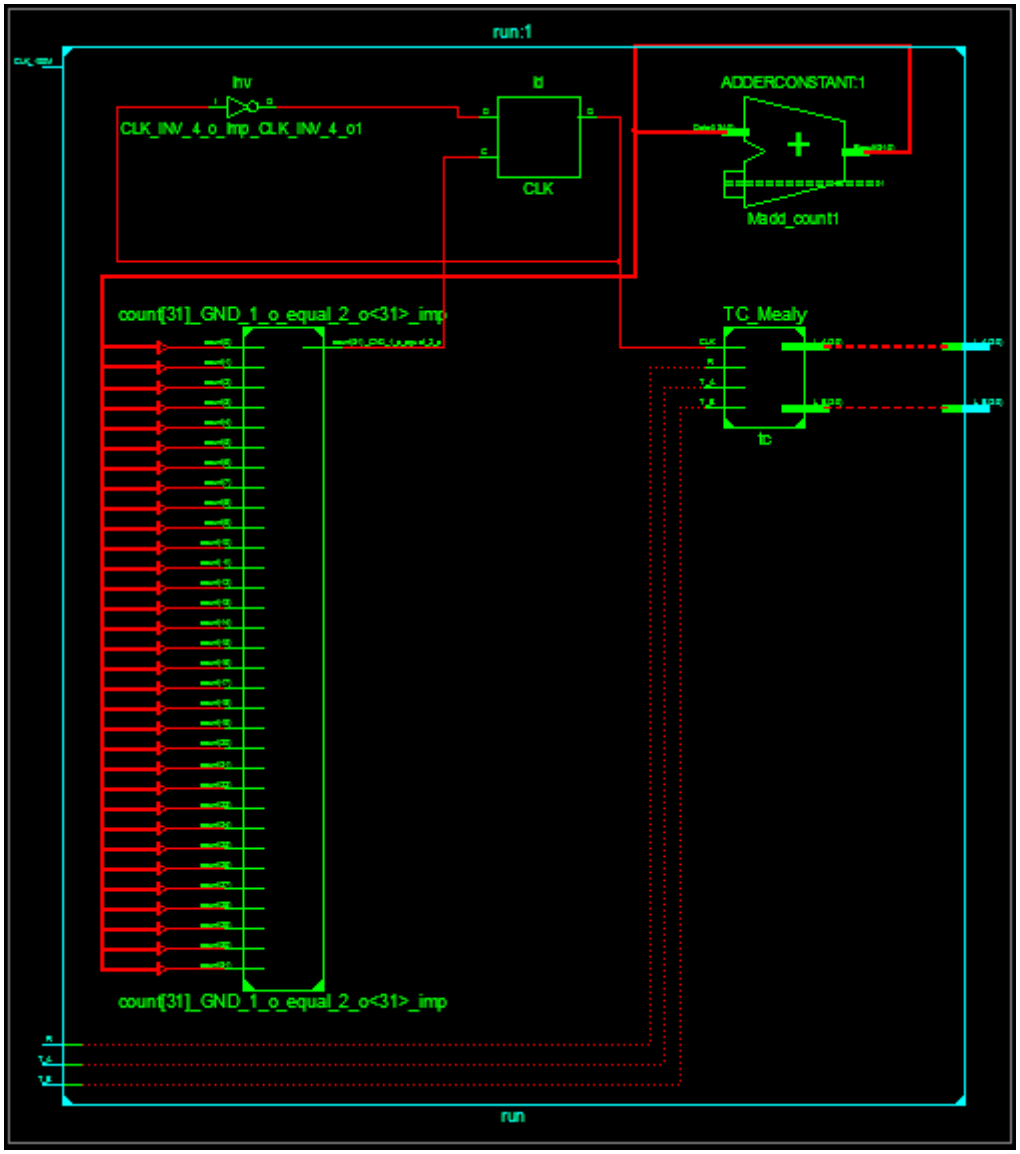
1. Traffic Controller (Moore), `run` module:



2. Traffic Controller (Moore), core controller module:



3. Traffic Controller (Mealy), `run` module:



4. Traffic Controller (Mealy), core controller module: