Computation II: embedded system design (5EIB0)

Part 2f: 2022-04-21 Finite State Machine (Design Washing Machine Control Unit - 7pt)

Due date: Thursday, 21 April 2022, 5:45 PM
Requested files: wcu.v, wcu_top.v (Download)
Type of work: Individual work

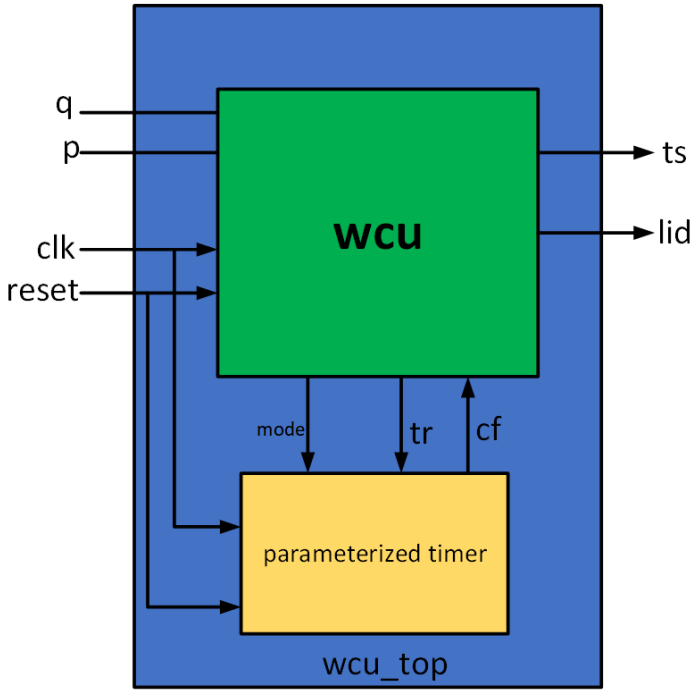
Introduction

This assignment is about the Verilog implementation of a washing machine control unit (**wcu**)
The behavior of the **wcu** repeats as follows:

First, the **wcu** is idle until a query for washing is issued. Then it does an evaluation to evaluate the weight of clothes entered into the washing machine.

If the validating is finished and the query has not been canceled, **wcu** starts the light washing program. It continues in this state until the specified time for the light washing program is completed. After that, the washing machine continues with a spin state if the light program is selected. If the heavy program is selected, it goes for another round of washing and when the second round is done, it continues with spin. The spin state continues until its specified time is over, and then **wcu** goes back into the idle state.

The top block diagram of the **wcu** unit is shown below:



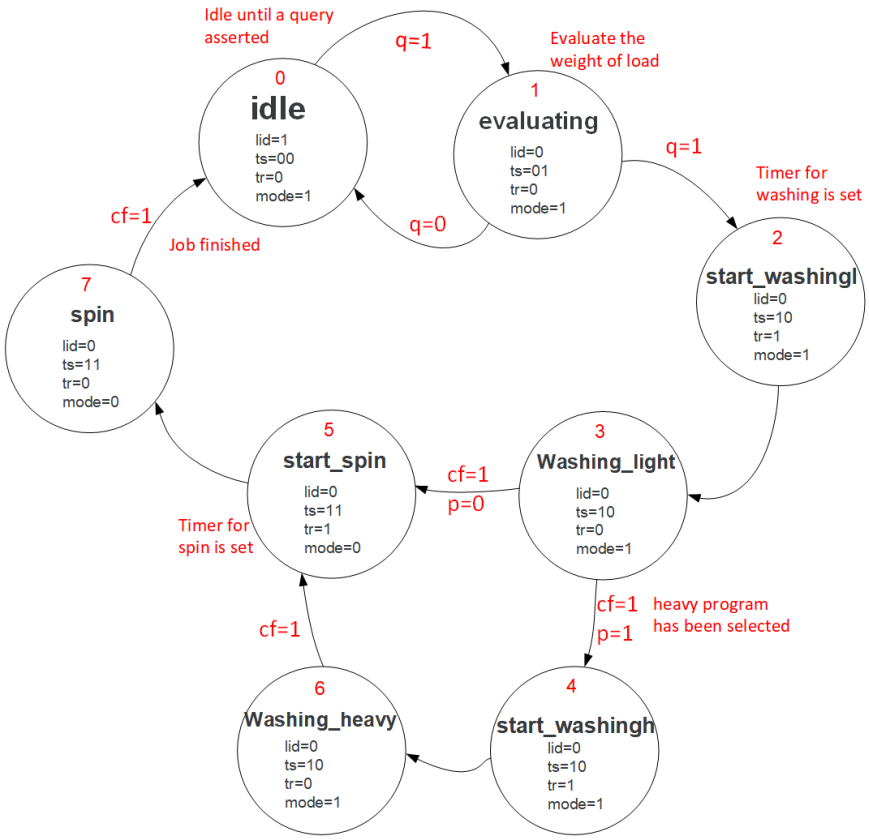
The block diagram consists of two modules: a finite state machine (FSM) module named **wcu**, and the parametrized counter (**parametrized_counter**) from the previous assignment. A module named **wcu_top** wraps these two modules.

The details of the signals are given below:

- **clk (1 bit):** Clock (positive edge)
- **reset (1 bit):** Reset (active high, synchronous)
- **q (1 bit):** Query input. **q = 1:** query is inserted or active. **q = 0:** query is completed or canceled.
- **p (1 bit):** Input signal to **wcu** indicating the light (=0) or heavy(=1) program

- **lid (1 bit):** Output signal for washing machine's lid (=1 ready to open)
- **ts (2 bits):** Status output of the current washing program (2 bits)(="00" idle mode, in which lid is open, ="01" just received the load and **wcu** is evaluating,="10" washing mode,="11" spin mode).
- **tr (1 bit):** Signal to *parametrized counter* to start counting for *specified* cycles
- **cf (1 bit):** Signal from *parametrized counter* that indicates *the specified* cycles have passed
- **mode (1 bit):** Signal indicating the mode of the parametrized counter.
  - 0: Run for ***tvalue*** – *2* cycles for spin
  - 1: Run for *3* × ***tvalue*** - *6* cycles for washing

The Moore state transition diagram for the **wcu** FSM module is given below:



The state names are indicated as follows: (idle, evaluating, start_washingl, washing_light,start_washingh,washing_heavy, start_spin, spin). Assertion of **reset** always causes a transition to the **IDLE** state.

Use the *parametrized counter* for counting the right amount of clock cycles so that the **wcu** state machine will be in the washing states (each of heavy and light) for **6 clock cycles** and spin state for **2 clock cycles**.

Assignment

In this assignment, you have to:

1. Set the correct value for the **tvalue** parameter in the **wcu_top** module.
2. Instantiate the **wcu** and **parametrized_counter** modules in the **wcu_top** module.
   (Follow the instructions regarding naming in the provided template!)
3. Implement the FSM module **wcu** (see the state transition diagram above).

Note: The **parametrized_counter** and **wcu_top** modules have already been provided. The **parametrized_counter** module synchronizes with the **wcu** module using the **tr** and **cf** signals.

Debug

Click on the symbol marked below to see the waveforms produced by your design. Please note that if your code has an error that prevents it being simulated it will not produce any waveforms.

***Note**: in the waveforms you might see that some of the inputs are undefined (x) at some moments in time. This means that the behaviour of your FSM should **not** depend on those values at that moment in time. (If your FSM incorrectly does use them anyway, it might cause undefined values (x) to appear at its outputs, and it will fail the test.)*

Requested files

wcu.v

```
`timescale 1ns / 1ps

    module wcu (
        input  wire [0:0] clk,
        input  wire [0:0] reset,
        input  wire [0:0] q,
        input  wire [0:0] p,
        input  wire [0:0] cf,
        output reg  [1:0] ts,
        output reg  [0:0] lid,
        output reg  [0:0] tr,
        output reg  [0:0] mode
    );

    // Add your FSM implementation here...

endmodule
```

wcu_top.v

```
`timescale 1ns / 1ps

/* Top-level module for crosswalk control unit */
module wcu_top (
         input wire clk,
         input wire reset,
         input wire p,
         input wire q,
         output wire [1:0] ts,
         output wire lid
     );

     // Number of cycles for each signal
     parameter tvalue = ...; // TODO: replace this with the correct value

     /* Use these wires for the signals between ccu and parametrized counter.
     Do not change the names of these signals, as the testbench will
     inspect them. */
     wire tr;
     wire cf;
     wire mode;



     /* TODO: Instantiate parametrized_counter module (provided for you).
     Make sure the instance is called parametrized_counter_inst. */

     /* TODO: Instantiate FSM module (which you have to implement in wcu.v).
     Make sure the instance is called wcu_inst. */

endmodule
```

VPL

◄ Part 2e: 2022-04-21 Design a parametrized counter - 6pt

Jump to...

Part 2a: 2022-04-21 Adding Custom Instructions (Selective Biasing Calculation - 1pt) (EDITABLE) ►