ESE 382 Digital System Design Using VHDL and PLDs

Spring 17, Ken Short
April 23, 2017 12:05 pm

Laboratory 11: FSM Based Temperature Measurement System

This laboratory is to be performed the week starting April 23rd.

Prerequisite Reading

1. Chapter 10 of the text.
2. LM34 Precision Fahrenheit Temperature Sensors Data Sheet (on Blackboard)

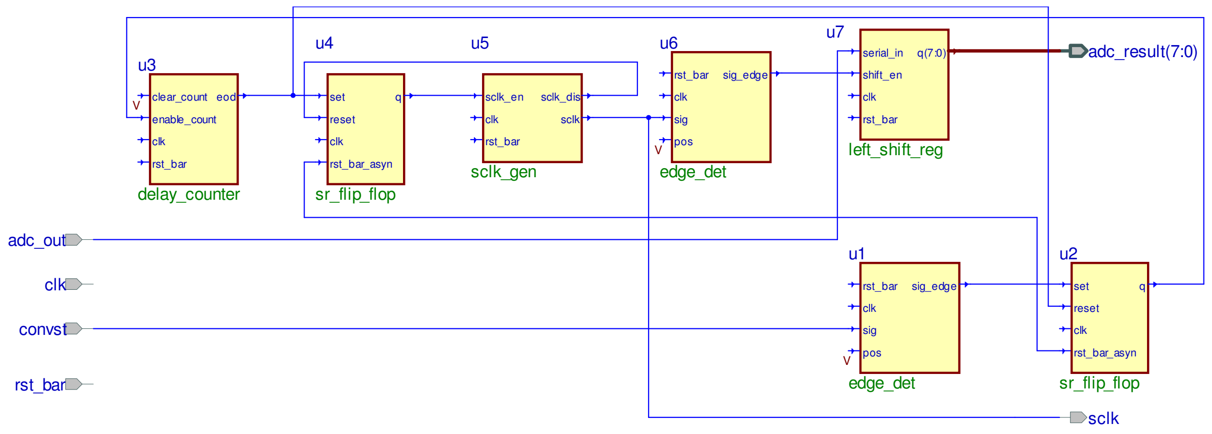
Purpose

The purpose of this laboratory is to provide you with experience in designing a sequential controller using a three-process finite state machine (FSM). The controller starts a conversion and then reads back and displays the serial results from a MAX1106 8-bit analog-to-digital converter (ADC). The analog input to the ADC is from a LM34 Precision Fahrenheit Temperature Sensor. The output voltage from the LM34 varies linearly with temperature and is 0 for a temperature of 0 degrees Fahrenheit. The slope of the LM34's transfer characteristic is 10mV/degree Fahrenheit. Therefore, the binary value read from the MAX1106 is the temperature in degrees Fahrenheit with a resolution of 1 degree.

Design Tasks

Design Task 1: FSM ADC Controller

In Laboratory 10 you created an ADC controller with a structural architecture consisting of seven entities, not counting the top-level entity itself.



In that heuristically created design, flip-flops were used to enable other entities to perform their functions. When flip-flop `u2` is set, it enables `delay_counter` to create the 36us delay. When flip-flop `u4` is set, it enables `sclk_gen` to generate the `sclk` pulses. The need for the flip-flops occurs somewhat naturally as one steps through heuristically creating a structure to implement the system.

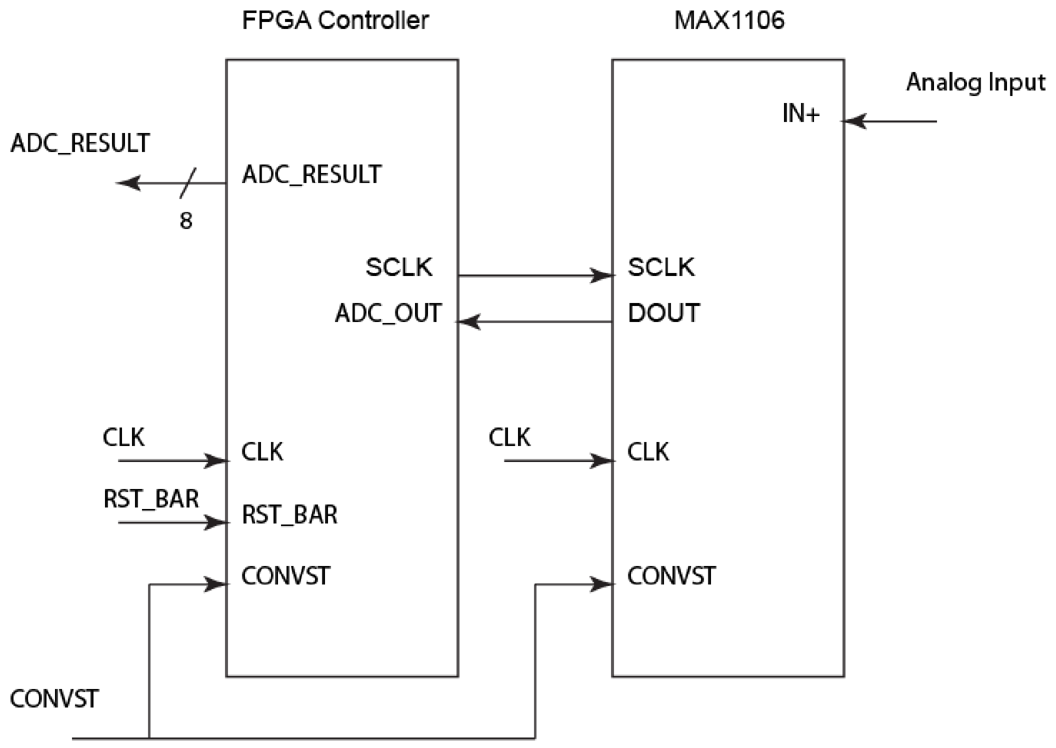
There is another way to look at the purpose of the flip-flops, they represent the state of the system (what the system is doing at a particular point in time). For example, one possible interpretation using this viewpoint is expressed in the following table.

Table 1: States of ADC Controller

u4	u2	State Name	Explanation
0	0	Idle	Waiting for convs pulse to start conversion
0	1	Delay	Waiting for the conversion to complete
1	0	Read	Reading ADC result

A more formal approach to designing the system would use an FSM to read pulses from the edge detectors, `delay_counter`, and `sclk_gen` as inputs and generate signals to `delay_counter` and `sclk_gen` to enable them to perform their functions. The flip-flops would then become part of the state register of the FSM and not be separate entities in the top-level structure.

Design a Moore type FSM to be used with the edge detectors and `delay_counter`, `sclk_gen`, and `left_shift_reg` entities to provide the same functionality as provided by the Simple Heuristic ADC Controller of Laboratory 10. Draw a state diagram for this FSM. Use enumerated state names corresponding to the state names from the previous table. If you need to add additional states, then do so. However, the goal is to use a few states as possible. Code the FSM using the three-process template from the textbook.



You must create your own entity declaration for your FSM. Name the entity `adc_fsm` and name its architecture `three_process`.

The top-level entity declaration and its functionality is the same as in Laboratory 10, except the entity name has been changed to `fsm_adc_cntrl`.

```
entity fsm_adc_cntrl is
   port(
      convst : in std_logic;      -- pulse to start a MAX 1106 conversion
      adc_out : in std_logic;     -- serial result from MAX 1106
      clk : in std_logic;         -- system clock
      rst_bar : in std_logic;     -- system reset, active low
      sclk : out std_logic;       -- serial clock
      adc_result : out std_logic_vector(7 downto 0)    -- parallel result
   );
end fsm_adc_cntrl;
```

The component entities stay the same, except the S R flip-flops are replaced by your FSM.

Draw the FSM state diagram. Write and simulate your FSM design description. Use Aldec stimulators for simulation. The clock frequency must be 1 MHz. Write and simulate the top-level entity. Include the signal `present_state` in your waveforms so that you can see your FSM change states.

Submit your state diagram, FSM design description, and simulation output waveforms as part of your prelab.

Design Task 2: Temperature Measurement and Display System
When a conversion is complete, output `adc_result` from the system of Task 1 gives the measured temperature in binary. This value must be displayed on two seven-segment LED displays to visually display the measured temperature. The `binary_7segment` entity from Laboratory 7 can be added to the design from Task 1 to accomplish this purpose.

The entity declaration for this top level entity is:

```
entity temp_meas_sys is
   port(
      convst : in std_logic;      -- pulse to start a MAX 1106 conversion
      adc_out : in std_logic;     -- serial result from MAX 1106
      clk : in std_logic;         -- system clock
      rst_bar : in std_logic;     -- system reset, active low
      sclk : out std_logic;       -- serial clock
      seven_seg0, seven_seg1 : out std_logic_vector(6 downto 0) -- 7seg outputs
   );
end temp_meas_sys;
```

An exhaustive self-checking sequential testbench is provided on blackboard. Create a second workspace for Task 2. Copy the code from Task 1 and modify it to produce the entity `temp_meas_sys`. Simulate your system with the testbench provided.

Submit your design descriptions and simulation output waveforms as part of your prelab.

Design Task 3: Removing Glitches from the Display System
In design Task 2, the input to `binary_7segment` entity was `adc_result` taken directly from the output of the `left_shift_reg`. When the ADC result is being shifted into this register, the register's contents are changing and will show up as glitches on the seven-segment displays. This might not be visible, since it only happens for only about 10us. However, in some applications these intermediate erroneous values can be a problem. Modify your design so that the inputs to `binary_7segment` do not have erroneous intermediate values. The top-level entity declaration does not change from what it was in Task 2. However, give the architecture name `improved`.

Write and simulate your design description using the testbench provided.

Submit your design descriptions and simulation output waveforms as part of your prelab.

Laboratory Tasks
Using Aldec Active-HDL create a new workspace named `lab11`. In this workspace create a separate design for each Task.

Laboratory Task 1: FSM ADC Controller

Import and compile your VHDL source files for Task 1. Functionally simulate your `adc_fsm` design entity using stimulators.

Demonstrate to a TA that your `adc_fsm` entity implements your state diagram. Obtain the TA's signature.

Simulate the top-level entity `fsm_adc_cntrl` using stimulators.

Have a TA verify that your top-level design simulation of `fsm_adc_cntrl` generates the appropriate output waveforms. Obtain the TA's signature.

Laboratory Task 2: Temperature Measurement and Display System
Import and compile your VHDL source files for Task 2. Functionally simulate your `temp_meas_sys` top-level design entity using the testbench provided.

Compile your design. Place and route your design to a Lattice LCMXO3L-6900C FPGA. Pin assignments will be provided in the laboratory. Place your programmed FPGA into the test fixture provided.

Have a TA verify that your programmed FPGA properly measures and displays the temperature when placed in the test fixture. Obtain the TA's signature.

Laboratory Task 3: Removing Glitches from the Display System
Import and compile your VHDL source files for Task 3. Functionally simulate your `temp_meas_sys` top-level design entity with architecture `improved` using stimulators.

Have a TA verify that your top-level design simulation shows that your `improved` architecture prevents the displays from changing while the shift register is being shifted. Obtain the TA's signature.