Thunderbird Turn Signal

Signature and Grading Sheet

Group #:_____ Name(s):_____ .

Signature

Section 4.3(f): _____.

Grading

- Section 4.1(a): VHDL code (10 points):_____.
  Attach code printout
- Section 4.2(a): FSM state diagram (15 points):_____.
  Attach 1-page state diagram
- Section 4.2(b): ASM chart (15 points):_____.
  Attach 1-page ASM chart
- Section 4.2(c): VHDL code (15 points):_____.
  Attach code printout
- Section 4.2(d): Simulation (15 points):_____.
  Attach simulation timing diagram printout.
- Section 4.3(a): VHDL code (10 points):_____.
  Attach code printout
- Section 4.3 (e) (5 points):_____.
  Attach one-page report printout with # of LEs circled .
- Section 4.3 (f) : demo signature (15 points):_____.

Total points: _____.

Experiment
Thunderbird Turn Signal

1 Purpose

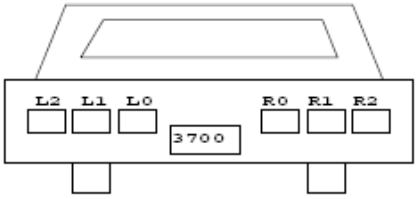
To design an FSM

2 Reading

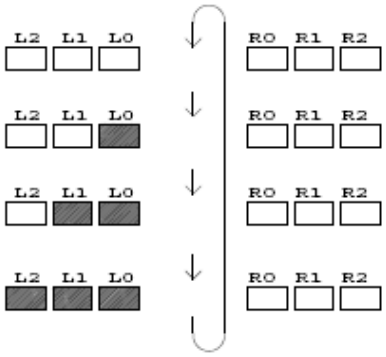
- Chapter 6 of *Embedded SoPC Design with Nios II Processor and VHDL Examples*

3 Project specification

The tail lights of a 1965 Ford Thunderbird is shown below:



There are three lights on each side that operate in sequence to indicate the direction of a turn. There are three flashing sequence: left turn, right turn, and hazard. The left-turn sequence is:



The right-turn sequence is similar and represents a "mirror" sequence of the left-turn pattern. In the hazard sequence, the six lights flash on and off alternatively. In all sequences, we assume that each pattern stays for 300 ms.

A simple FSM can be constructed to control the tail light operation. The input and output are

- input:
  - `clk`: 50 MHz clock signal from the DE1 board.
  - `reset`
  - `tick`: 1-bit 300ms "tick" signal. It is asserted for one clock cycle every 300 ms.
  - `left`: 1-bit left-turn signal.
  - `right`: 1-bit right-turn signal.
  - `haz`: 1-bit hazard signal.
- output
  - light: 6-bit light signal.

The system operates as follows:

- Anytime `haz` is asserted, the FSM enters the hazard sequence immediately. If the FSM currently in the middle of a left- or right-turn, sequence, the sequence will be aborted.
- When `haz` is not asserted and `left` is asserted and, the FSM goes through the complete left-turn sequence. This means that the lights should go through a complete left-turn sequence even if `left` is de-asserted sometime in the middle of the sequence or if `right` is asserted in the middle of a sequence. However, the FSM enters the hazard sequence if `haz` is asserted.
- When `haz` is not asserted and `right` is asserted and, the FSM goes through the complete right-turn sequence. This means that the lights should go through a complete right-turn sequence even if `right` is de-asserted sometime in the middle of the sequence or if `left` is asserted in the middle of a sequence. However, the FSM enters the hazard sequence if `haz` is asserted.
- We assume that `left` and `right` will never be asserted simultaneously.

The design must be synchronous or 50% will be deducted.

4 Design Procedures

We can divide this circuit into two segments:

- Counter: generate a one-clock pulse (`tick`) every 300 ms.
- FSM.

4.1 Counter

(a) Design the counter as an independent VHDL module and derive VHDL code.

4.2 FSM

(a) Derive the FSM and draw the complete state diagram.

(b) Convert the state diagram to an ASM chart following the notations used in the text.

(c) Design the FSM as an independent VHDL module. The entity declaration of this design is

```
entity tbird_fsm is
   port(
         clk, reset: std_logic;
         tick, left, right, haz: std_logic;
         light: out std_logic_vector(5 downto 0);
    );
END tbird_fsm;
```

Derive the architecture body.

(d) Compile the design and perform simulation to verify its operation. In the timing diagram, the internal symbolic state values of state register should be included and the different input combinations should be properly tested.

4.3 Implementation and testing

(a) Derive the VHDL code for the combined circuit. You can either cut-and-paste the previous codes or create a top-level VHDL file with component instantiations.

(b) Use the 50MHz oscillator for clock.

(c) Use 4 switches for the reset signal and three control signals.

(d) Perform pin assignment and synthesize the code.

(e) Look at the compiling report and determine the number of LEs used in circuit.

(f) Download the file to the FPGA device and verify the operation of physical circuit. Demonstrate the circuit to instructor and get signature.