EEL4740 –Embedded Computing Systems Final Exam Study Guide and Review Material

Chapter1 - Introduction

- What is an embedded system?
- Design challenge in terms of optimizing design metrics.
- Processor technologies
  - Single-purpose
  - Application-specific
  - General-purpose
- IC technologies
  - VLSI
  - ASIC
  - FPGA
- Steps in design technology
  - Come up with system specifications
  - Behavior synthesis by coming up with FSMD
  - RT synthesis by writing VHDL code and using HDL simulators
  - Logic synthesis by using gate simulators

Chapter 10 – IC Technology

- Full-Custom (VLSI)
  - The designer must design the transistor-level circuit for every processor and memory.
- Semi-Custom (ASIC)
  - Application Specific Integrated Circuit (ASIC) is designed for a specific application. Lower layers of the chip, this is all of the logic gates of the IC have already been laid out, leaving the designer with the task of connecting the gates.
- Programmable Logic Device (PLD)
  - A premanufactured IC that can be purchased and configured to implement the desired circuit. Field-Programmable Gate Arrays (FPGAs) are the most common PLD used for embedded system design. An FPGA consists of arrays of programmable logic blocks connected by programmable interconnect blocks.
  - Programming is done by setting bits within the logic or interconnect blocks. Those bits are stored using nonvolatile (EEPROM) or volatile (SRAM) memory. In case of SRAM based FPGA, a nonvolatile configuration memory and additional steps are needed to store the program in the FPGA board.

- FPGA Applications

Week 2 - Hardware Description Language: VHDL

- VHDL code has three parts:
  - **Library** *use clause*
  - **Entity** *declaration*
  - **Architecture** *body*
- VHDL simulation
- VHDL Synthesis
- VHDL Syntax
- Combinational Logic in VHDL
- Conditional Assignment in VHDL
- Internal Variables in VHDL
- Structural Modeling in VHDL
- FSM in VHDL
- VHDL testbench

Chapter 2 – Controller Design

A standard controller architecture consists of a state register and combinational logic. It implements an FSM.

- Controller Design Process
  Step1: Capture behavior
    - <u>Capture</u> the Finite State Machine (FSM), *FSM describes the desired behavior of the controller. A state diagram is a graphical drawing of FSM.*

  Step 2: Convert to circuit

- Modeling Finite-State Machine in VHDL

Datapath Components

Datapath stores and manipulates a system's data. Datapath can be configured to read data from particular registers, feed that data through functional units configured to carry out particular operations like add or shift, then store the operation results back into particular registers.

Controller carries out such configuration of datapath. It sets the datapath control inputs and monitors external control inputs.

Typical datapath components include:

- Registers
- Adders
- Multiplexers
- Comparators
- Multipliers
- Subtractors

- ALUs
- Shifters
- Counters and Timers

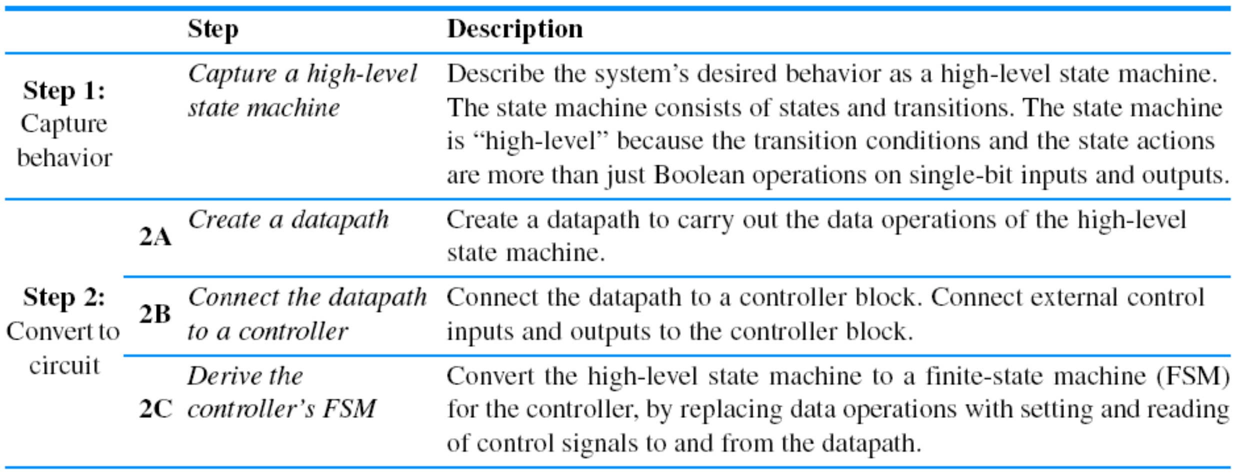
Custom Single-Purpose Processor Design

A custom single-purpose purpose design is specialized to implement one specific computation. It consists of a controller connected to a datapath. Single-purpose processor has no program memory, the program is implemented by the controller.

A complex state diagram called *finite state machine with data,* FSMD is used for the design of custom single-purpose processors.

In FSMD, states and arcs may include arithmetic expressions, and those expressions may use external inputs and outputs as well as variables. Our earlier state diagrams included only Boolean expressions, and those expressions could use only external inputs and outputs but not variables.

- Single-Purpose Processor Design Method



		Step	Description
Step 1: Capture behavior		*Capture a high-level state machine*	Describe the system's desired behavior as a high-level state machine. The state machine consists of states and transitions. The state machine is "high-level" because the transition conditions and the state actions are more than just Boolean operations on single-bit inputs and outputs.
Step 2: Convert to circuit	**2A**	*Create a datapath*	Create a datapath to carry out the data operations of the high-level state machine.
	2B	*Connect the datapath to a controller*	Connect the datapath to a controller block. Connect external control inputs and outputs to the controller block.
	2C	*Derive the controller's FSM*	Convert the high-level state machine to a finite-state machine (FSM) for the controller, by replacing data operations with setting and reading of control signals to and from the datapath.

- Two approaches for Custom single-purpose processor design in VHDL:
  - High-level state machine FSMD
    - Model FSMD in VHDL (*only this approach was covered in this course*)
  - Controller and datapath
    - Write the VHDL code for controller and datapath separately, use structural modeling and include datapath and controller components in the structure.

Chapter 3 - General-Purpose Processors

General-Purpose Processors are designed for a variety of computation tasks, the unit and NRE cost is low, time-to-market is short and they are flexible since they can be used for a variety of applications. The embedded system engineer has to only program and interface them with the physical structure.

- Embedded system design using general purpose processors requires:
  - A development board
  - IDE software loaded on a computer

    - ➢ Debugger

- Selecting a Microprocessor
  Selecting a microprocessor for a given embedded system involves technical issues like: speed, power, size, cost; and other issues like: development environment, prior expertise, licensing, etc.

Chapter 4 – Peripherals

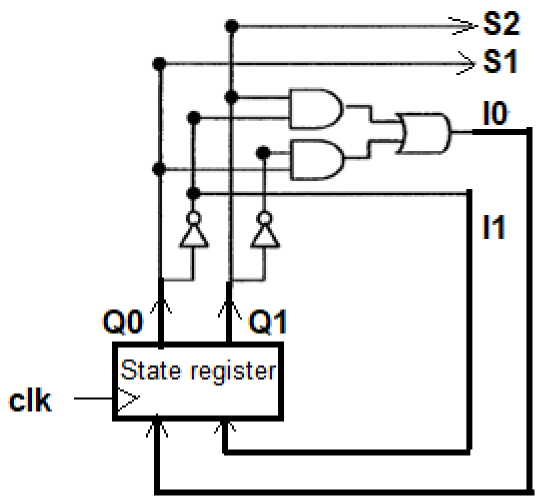
Common peripherals for embedded processors include: Timer/Counters, UART for serial transmission, LCD controller, Keypad controller, Stepper motor controller…. These peripherals (also called Standard single-purpose processors) pre-designed for a common task. They are commonly part of SoC(system on the chip) for embedded systems. For FPGA development the IP (Intellectual Property) of these peripheral is available in the design suite.

- Timers
  - ➢ Resolution
  - ➢ Range
  - ➢ Prescaler
- Counter: like a timer, but counts pulses on a general input signal rather than clock
- UART: Universal Asynchronous Receiver Transmitter
  - ➢ Takes parallel data and transmits serially
  - ➢ Receives serial data and converts to parallel
  - ➢ UART standard and protocol
- Pulse width modulator
- Serial Interface standards
  - ➢ I^2C Bus

Review Problems:

1. Design an 8-bit register using D flip-flops. Write the VHDL code for it.

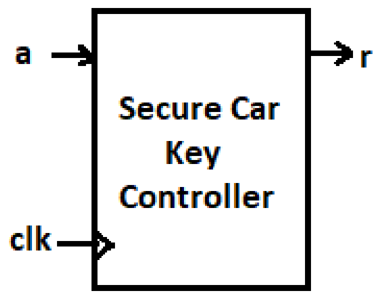
2. Convert the following sequential circuit to FSM, draw the state diagram.



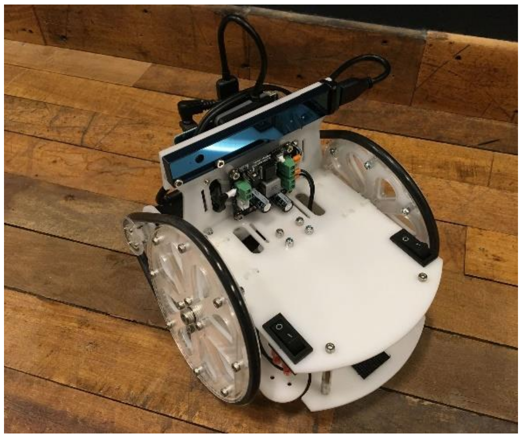
3. In case of a secure car key, when the driver turns the key in the ignition, the car's computer sends out a radio signal asking the car key's chip to respond by sending an identifier via a radio signal. A secure car key then responds by sending the identifier (ID). Design a controller for such a car key having an ID of 1011. Write the VHDL code for this controller.

input: a ;signal from car' computer

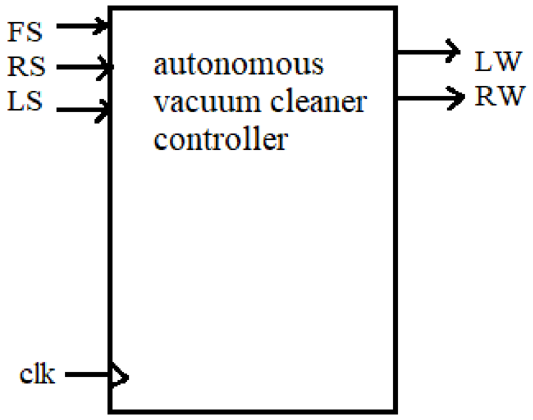
output: r ;ID bits sent in serial



4. Design a controller to control an autonomous vacuum cleaner. It runs on two wheels. If both wheels run, cleaner moves forward. If the right wheel is off, cleaner makes a right turn; if the left wheel is off the cleaner makes a left turn. There are three sensors mounted on the cleaner that can detect an obstacle, one in the front, one on the right, and one on the left.
   Once the cleaner is turned on, it moves forward; if an obstacle is ahead, it makes a right turn. Unless there is an obstacle on the right side too, then it makes a left turn. Cleaner will stop, if there are obstacles in front, on the right, and on the left.
   Come up with state diagram for this design, then use it to write the VHDL code.



inputs:	FS	front sensor
	RS	right sensor
	LS	left sensor
outputs:	LW	left wheel
	RW	right wheel


FS
RS
LS
autonomous vacuum cleaner controller
LW
RW
clk