CSEE W4823x **Homework 4** Handout 21
Prof. Steven Nowick October 18, 2016

This homework is due **at the beginning of class** on Tuesday, October 25. *No group work is allowed on this assignment.*

*Note:* A correct answer without adequate explanation will have points deducted. To get full credit, (a) write legibly/type, and (b) show all work (label relevant items, show deriviations, include explanations).

1. (35 points) **Low Power Design: Precomputation-Based Sequential Logic Optimization.** In this problem, you are to create two different low-power versions of a design of an *ASCII alphabetic character detector* stage in a pipelined sequential system. Refer to Handout #19 and the corresponding class lectures.

   *Setup.* The ASCII character code is described in Brown/Vranesic, ch. 5.8 (3rd edition). This popular 7-bit code is used to encode letters, digits, and some control characters. The code is given in Table 5.3.

   In this problem, you are to focus on a combinational logic block which detects if a given ASCII code is an alphabet symbol. This alphabet detector block has 7 inputs ($x_6$, $x_5$, $x_4$, $x_3$, $x_2$, $x_1$, $x_0$) and has 1 output ($f$). The block's output is 1 if the input is an ASCII alphabet letter (either lower-case [a-z] or upper-case [A-Z]). In all other cases, the block's output is 0.

   This block used in a pipelined system architecture as shown in Figure 1 of Handout #19, and will serve as *"Block A"*.

   For simplicity, *you do not need to design this function block!* Assume it is given as a black box. You are to use this block, while designing the rest of the logic needed for two precomputation architectures.

   (a) **First Precomputation Architecture.** Design a low-power sequential optimized version of Figure 1, using the alphabet detector circuit as "Block A", based on the first precomputation architecture shown in Figure 2 of Handout #19.
   *What to Do:*
   *(i) Show all work:* draw the complete final microarchitecture, clearly label your figure, include a *clear derivation and explanation of the logic equations* for $g_1$ and $g_2$, and *draw gate-level designs* of $g_1$ and $g_2$. (A minimally-labelled or minimally-explained correct design may have points deducted.)
   *(ii) Clearly derive the "hit rate" of your precomputation blocks:* assuming random input vectors, for what percentage of vectors will the register disabling be activated and the bypassed precomputed outputs be generated?
   ***Requirement:*** Your "predictor" block (containing $g_1$ and $g_2$) should examine exactly 4 different inputs: $x_6$, $x_3$, $x_2$ and $x_1$. It should be as simple as possible.

   (b) **Second Precomputation Architecture.** Repeat part (a) above, but now based on the second precomputation architecture shown in Figure 3.
   *What to Do:* follow same guidelines as for part(a), and complete the same steps and answer the same questions.

2. (35 points) **Low Power Design: Bus-Invert Coding.** In this problem, you will explore the bus-invert coding method of Stan/Burleson, as covered in Handout #22 and class lectures. Assume *10-bit data words,* which you will encode through bus-invert coding.

   We consider two alternative bus-invert codes for the 10-bit data:

   **Code #1: Monolithic Code.** One 10-bit data field followed by 1 invert bit.

   **Code #2: Partitioned Code.** 10-bit data divided into two 5-bit fields, where each of the two data fields is followed by its own invert bit (i.e. 2 invert bits total per codeword).

(a) **Simple Encoding Example.** You are given the following sequence of *7 data bytes,* transmitted in order:

00000 00000
01100 11011
11110 01011
00001 10110
11001 10110
11000 01001
00000 00000

Assume that the first data word (0000000000) is transmitted with invert bit(s) set all to 0. *Write out* the corresponding transmitted code sequence for this seven data byte transmission using **(i) using Code #1,** and **(ii) using Code #2**.

*Note:* Clearly indicate the fields of each codeword, highlighting data field(s) and I bit field(s).

(b) **Cost Evaluation: Original (unencoded) Sequence.** Using the methods presented in Handout #22 and in class lectures, derive the (i) *average power per data byte transmission,* and (ii) *coding efficiency*, for the given unencoded sequence, shown above.

The power metric should be **averaged over the actual sequence of 7 transmitted codewords above**.

This is a different average than was calculated in class: we covered an average over a random distribution, while for this problem the average is for a particular transmission sequence.

*Note:* By "average power", we mean "average number of bit transitions per transmission", assuming the above concrete sequence of 7 codewords. Since one of these (0000000000) is the starting codeword, you will compute average power *on only the 6 subsequent data transmissions.* By "coding efficiency", we mean "number of bits/number of wires".

*For full credit, show all work and clearly label and explain each step.*

(c) **Cost Evaluation: Code #1.** Repeat part (b), but now for Code #1.

(d) **Cost Evaluation: Code #2.** Repeat part (b), but now for Code #2.

(e) **Cost Evaluation and Tradeoffs: Summary.** To compare the results, (i) *create a table listing the 3 encodings (original, Code #1, Code #2), and their two cost functions*; and (ii) *provide a 3-4 sentence summary,* clearly identifying the tradeoffs between these two different bus-invert codes (Code #1, Code #2), for the above two cost functions (transition power, coding efficiency). Discuss concretely which one is better and which one worse, for each cost. Also, propose an explanation on why these trends occur.

3. (10 points) **Sequential VHDL (part 1): Modelling TFF's.** Do B/V problem 7.10 (3rd edition only).

4. (20 points) **Sequential VHDL (part 2): Modelling a Universal Shift Register.** You are to model a 16-bit universal shift register using behavioral VHDL (not structural!). This unit can shift in either direction, hold, or load. Assume it has a 16-bit input, $X$, and a 16-bit output, $Z$. It also takes as inputs the Clock (operating on the positive edge), and two select bits $L$ and $R$. It also takes in a left input bit, $Lin$, and a right input bit, $Rin$. Its operation is based on the select bits: $LR = 00$ means hold, $LR = 10$ means left shift, $LR = 01$ means right shift, and $LR = 11$ means load. Assume the unit also has an asynchronous reset. *Write clean behavioral VHDL for its entity and architecture.*