EE431 Second Lab Midterm for 2005

Date: Thursday, March 31, 2005
Time = 3 hours
Text Books, Notes and Computer Files Only

1. Start a new project called "MT2_Q1_2005". The main entity will be "MT2_Q1_2005.v" Down load the file MT2_Q1_2005.v from I drive (I:) → classes → EE431. Then open the file and modify it by instantiating your microprocessor in place of the instantiation of the instructor's microprocessor. Do not compile the project yet.

   The Verilog file that was downloaded is a prototype for a circuit that provides an input to "i_pins" and that scrambles the 4-bit output ("o_reg") to a 12-bit number and samples the scrambled output when a counter reaches a certain value and places the sample in a register called "answer". The value of "answer" is zero for all but one clock cycle, and that is the value to be reported.

   The 12-bit input called seed must be set to a constant for the duration of the simulation. The value of that constant is given on the colored sheets. The value of "answer" must also be reported on the colored sheet.

   To help with debugging, the value for "answer" is given when "seed = 12'HACE".

(10) (a) Copy the hex file Q1a.hex from folder I drive (I:) → classes → EE431 and make it the program file for your micro. Compile the micro.
   The listing file for Q1a.hex is also in the folder. This may be useful for debugging.
   Simulate the circuit with the following settings. Set the end time for the simulation to 1040 micro seconds. Display signals" clk", "reset", and "answer". You may want to display "o_reg" to help with debugging. Set the clock, "clk", to a frequency of 1 Mhz (1 μs period) and start the clock low so that the first positive going edge is at 0.5μs. Make "reset" high for 2.25$\pm$ 0.1 μs.
   Observer "answer". It will be nonzero for one clock period at or about 1025 μs. If your circuit is working properly the value will be 12'HA32. This value is also given on the colored sheet.
   Change input seed as per the colored sheet, re-simulate and report "answer" on the colored sheet.

(10) (b) Repeat the process using program file Q1b.hex. When seed is 12'HACE, "answer = 12'H5A7".

(15) 2. This question uses the same test circuit as question 1. The top entity does not change. However your microprocessor must be modified.

Save a copy of your program sequencer. Modify the program sequencer of your microprocessor to change the jump instructions to jump-to-subroutine instructions. Do this as follows.

Copy the program counter (PC) to a new register called "return_address" upon execution of every jump instruction (which includes jmp 8'HF0 even though this instruction will be converted to a return-from-subroutine instruction) and on conditional jump instructions that occur at a time when r $\neq$ 0 (i.e. conditional jump instructions that make jumps). Load the "return_address" register with the rising clock edge that executes the jump instruction. On all other instruction and on conditional jump instructions that occur at a time when r = 0, "return_address" is not changed.

Modify the next address logic so that instruction "jmp 8'HF0" becomes a return-from-subroutine instruction. This is done by making
"pm_address = return_address + 8'H1"
when the instruction is "jmp #8'HF0", i.e. when the instruction is jump and ir[3:0] = 4'HF.

Use the same top entity as question 1. Make the program file Q2.hex. Compile and simulate. The non-zero value of "answer", which occurs at about 1025 μs, will be 12'HF8C when input "seed" is the constant 12'HACE.

Make "seed" the value given on the colored sheet and re-simulate. Report the value of "answer".

(10) 3. Again use the top entity and instantiated microprocessor of question 1. Start with the original program sequencer. Modify the original program sequencer so that both the unconditional and conditional jump instructions are two byte instructions that permit jumps to any address. That is to say unconditional and conditional jump instructions take two bytes of program memory: The first byte is the normal instruction (but the 4 least significant bits will not be used).

The second byte of the instruction (stored in the next higher address of program memory) is the 8-bit address of 'where to jump'. For example "jump to 8'H1C" could be the two byte sequence '8'HE0 8'H1C'. 8'HE0 would be stored at some location in the ROM, say at address 8'H05, then 8'H1C would be stored at the next location, which is address 8'H06. The least significant four bits of the first byte are ignored so the two byte sequence 8'HEF, 8'H1C would execute the same jump.

This modification is made as follows.

(a) Revise the port list of the program sequencer to include"pm_data", which is to used as an 8-bit input. Further revise the list by removing the 4-bit input connected to ir[3:0], this would have been called jump_address or something like it.

(b) Revise the connection list in the instantiation made in the top entity so that 'pm_data' is connected to 'pm_data'.

(c) Build the circuit shown in Figure 1 inside the sequencer. Call the output of this circuit "clk_delayed".

(d) Build an 8-bit register called "J_address" and load this register with "pm_data" on every **negative edge** of "clk".

(e) Alter the circuit that produces "pm_address" to satisfy the following:
If the instruction in the "ir" register is **neither** a conditional jump **nor** an unconditional jump then "pm_~~data~~address" is as before.
If the instruction in the "ir" register is **either** a conditional jump **or** an unconditional jump then "pm_~~data~~address" must be such that:

i. While "clk_delayed" is high, $pm\_address = PC + 8'H01$.

ii. While "clk_delayed" is low *pm_address* depends on whether or not the jump is going to be executed:

A. If the jump is going to be executed (i.e. an unconditional jump or a conditional jump at a time when r $\neq$ 0) $pm\_address = J\_address$.

B. If the jump is not going to be executed, i.e. a conditional jump instruction at a time when r = 0, then $pm\_address = PC + 8'H02$

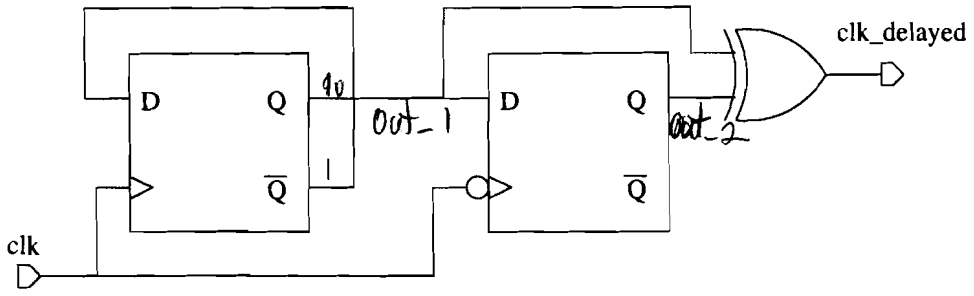

Figure 1: Circuit that delays clk by a little bit

Again use the same top entity and simulation waveforms as question 1. Make file Q3.hex your program file. The non-zero value of "answer", which occurs at about 1025 μs, when "seed = 12'HACE" will be 12'H070 if your modifications are correct.

Change seed as per the colored sheet, re-simulate and report the value of "answer".