UPC. EETAC. Bachelor degree. Second course 2A. Digital Circuits and Systems (CSD). Dr F. J. Robert. Grades will be available online on June 11th. Questions about the exam at office time.

Exam 2. **June 4th, 2021**

Problem 1. (2.5p)

Analyse the circuit represented in Fig. 1.

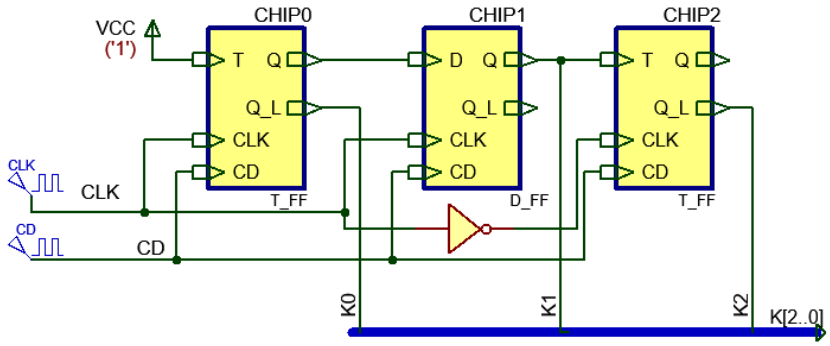
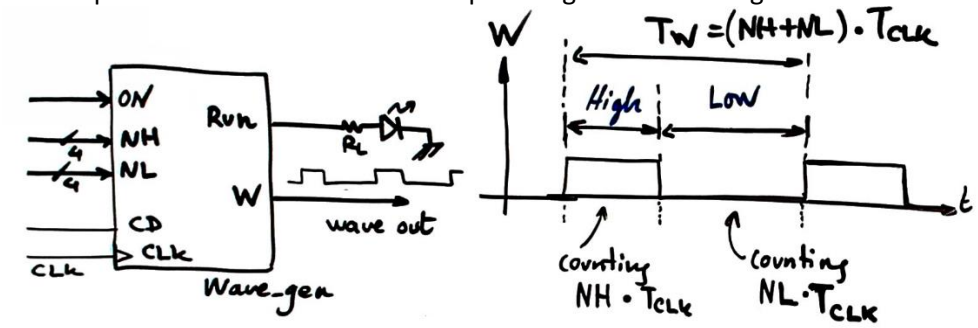


Fig. 1
Circuit based on 1-bit memory cells and logic gates.

1. Determine the output vector **K[2..0]** drawing a timing diagram considering enough CLK periods.
2. Write down the binary codes generated.
3. Explain how many VHDL files are necessary to develop and simulate the circuit using EDA tools.

Problem 2. (3.5p)

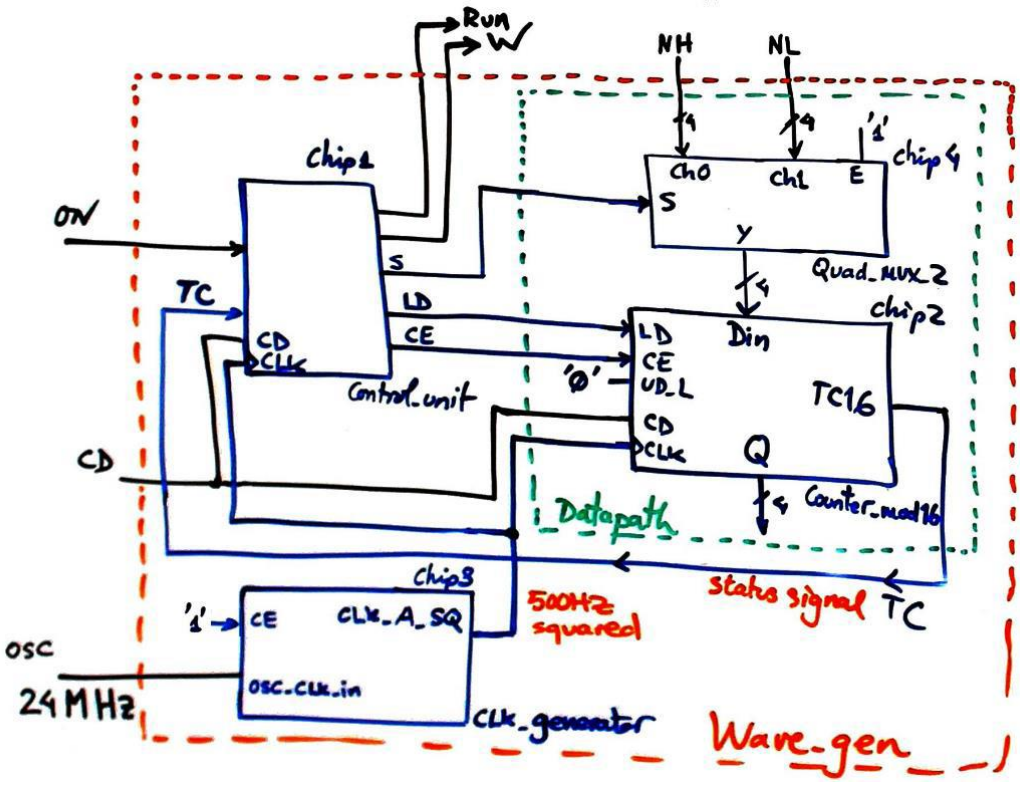
Design (specify and plan) the programmable rectangular wave generator represented in Fig. 2 using VHDL techniques and structural plan C2 for a target FPGA chip. The FSM is controlling a datapath based on a *Counter_mod16*. The 4-bit radix-2 number **NH** establishes the number of CLK pulses where wave output (**W**) is high, **NL** establishes the number of CLK pulses where W is low. **Run** output is high when running and low when idle.



a) b)

Fig. 2.
a) *Wave_gen* symbol.

b) Example W waveform when running.



c)

c) Proposed dedicated processor architecture for this project.

UD_L = '0' means that the *Counter_mod16* is configured as **down counter** and thus TC16 is a zero detector status signal to be used by the FSM.

The function table of *Counter_mod16* is in Fig. 4.

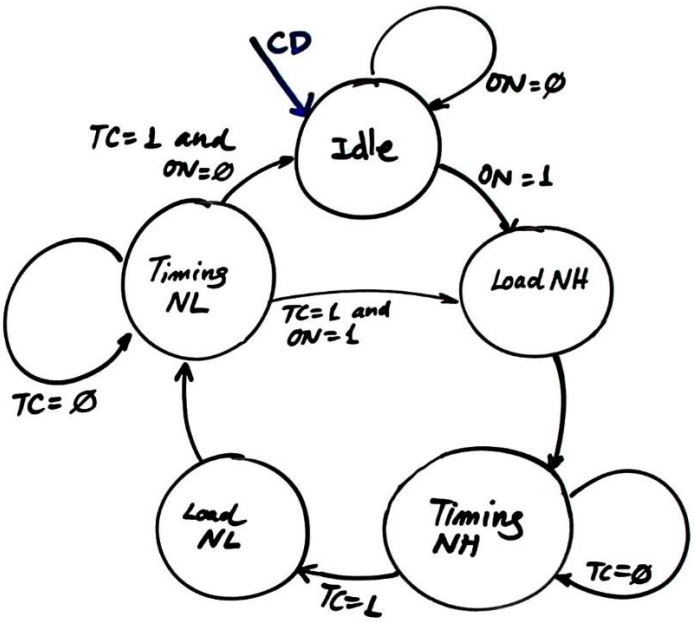
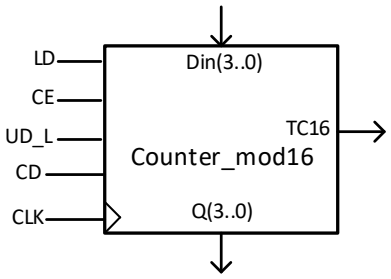


Fig. 3. State diagram proposed for Chip1 control unit (FSM) showing only states and transitions.

1. Invent the Chip3 *CLK_Generator* to obtain a 500 Hz square wave from the 24 MHz crystal oscillator.
2. Calculate the frequency of the output rectangular wave W when NH =10 and NL = 4.
3. Explain how many *D_FF* registers will contain this project *Wave_gen.*
4. Draw the Chip1 internal FSM architecture connecting all the control unit inputs and outputs.
5. Draw the Chip1 CC2 truth table to determine all the outputs represented usually in parenthesis in Fig. 3.
6. Invent an alternative architecture for the datapath if *Counter_mod16* is used as **up counter** with UD_L = '1'.

Problem 3. (4p)

Design the PIC18F4520 microcontroller version of the *Counter_mod16* using a plan Y adaptation (no state enumeration, *current_state* is a RAM variable to save the current binary number that is copied to Q outputs).



LD	CE	UD_L	Q^+	Synchronous operation after the CLK's rising edge
1	x	x	Din	Parallel load (register data)
0	0	x	Q	Do nothing (inhibit)
0	1	1	$(Q+1)_{mod16}$	Up counting in binary
0	1	0	$(Q-1)_{mod16}$	Down counting in binary

TC16 = '**1**' when CE = '1' *and* [(Q = 15 *and* UD_L = '1') *or* (Q = 0 *and* UD_L = '0')]; '0' otherwise

Fig. 4 *Counter_mod16* symbol and function table to be used in Problem 3 and Problem 2.

1. Draw the hardware: input switches, buttons, outputs, reset (MCLR_L) and 12 MHz quartz crystal oscillator.
2. Draw the hardware-software diagram. Why the CLK for counting has to be connected to RB0/INT0 pin? What the interrupt service routine *ISR()* is used for?
3. Organise and name RAM variables for the project. Explain how to configure port pins and interrupts in *init_system*().
4. Explain how to poll the input values using bitwise operations in *read_inputs()*.
5. Explain how to drive the five output pins using bitwise operations in *write_outputs().*
6. Draw the truth table and flowchart for the *output_logic()*.
7. What functions will be modified and how if we like to add an LCD to represent counter states?
8. How to configure and program the TMR0 to replace the external CLK if we require counting at 100 Hz?

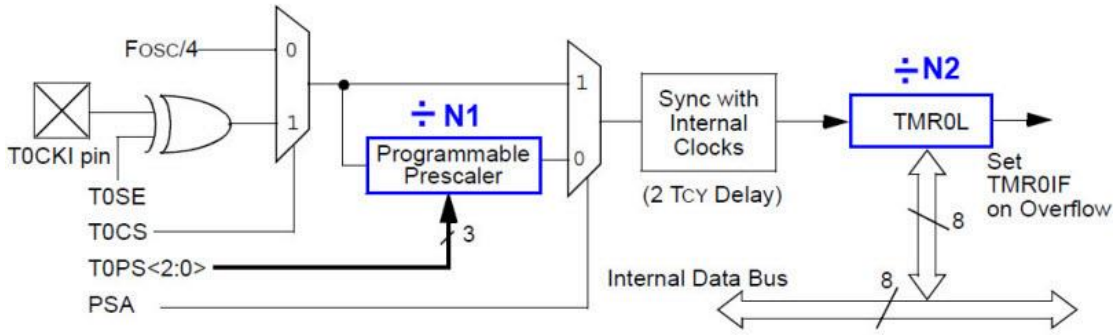


Fig. 5 TMR0 architecture for 8-bit mode.

(Complementary questions for developing and testing the final project using microcontroller EDA tools.)

9. Draw the truth table and the flowchart for the *state_logic()*.

10. Develop the project in Proteus and MPLABX, debug and verify. This is another P10 example.

- Phase #1: Only up counting. Adapt hardware and software from the tutorial plan Y *Counter_mod1572*.
- Phase #2: Up and down. This is basically modify *state_logic()*
- Phase #3: Parallel data inputs. This is again basically modifying *state_logic()*
- Phase #4: Add an LCD to represent binary and decimal unsigned numbers.