

LAB 2

Design and Simulation of Sequential Logic Circuits - Synchronous Counters -

1. Purpose:

This lab will enable students gain practice in the conversion of functional requirements into logic circuits and their implementation using EMP7128S complex programmable logic device (CPLD), on the Altera UP2 board.

The purpose of this lab is to introduce students to the design of sequential circuits based on Altera's Quartus development environment and their implementation and testing with CPLD.

- Enter the design of synchronous counters using Quartus II graphics editor
- Assign the input-output pins and prepare the design for downloading and testing on the Altera UP2 board
- Test the counter:
 - Display the counter outputs as binary values on LEDs
 - Using an oscilloscope, trace and record the waveforms at various flip-flops.

2. Requirements of the Lab:

The following results need to be submitted in your report.

- * The log of what you did
- * The screen shots of all schematics and all waveform diagrams
- * Compilation, simulation and downloading messages (if any)
- * Your test results

3. Equipment and Supplies:

- * Quartus II (student edition or web edition)
- * Altera UP2 board with
 - Byte blaster cable
 - EMP7128S CPLD
 - Power supply 7 VDC, 250 mA
- * Tools: anti-static wrist straps, 22 gauge wire, handtools

4. References:

- Chapter1 and 2 of the Text book: Computer Systems Architecture, Morris Mano, 3rd Ed.
- Quartus II Tutorial: http://www.altera.com/literature/manual/intro_to_quartus2.pdf
- UP2 Education Kit User Guide
http://www.altera.com/literature/univ/upds.pdf?GSA_pos=1&WT.oss_r=1&WT.oss=UP2%20manual

5. PreLab

1. For each of the following counters (**a.** and **b.**):
 - Draw the state diagram and derive the excitation table for all the flip-flops involved in the counter (the excitation table for counter **a.** is already given below),
 - Derive and simplify the Boolean expression of every flip-flop input using Karnaugh maps. Visit the Altera website to familiarize yourself with the Altera UP2 board (reference **iii.**)

- a. A **3 bit synchronous modulo 6 counter** - block diagram in Fig. 1.(a), has to observe the following counting sequence: 000 --> 010 --> 110 --> 011 --> 101 --> 100 --> 000. This counter is to be implemented with JK flip-flops which have active-low asynchronous *Reset* inputs ($\overline{\text{CLR}}\text{N}$); the flip-flops' clock inputs (CLK) are connected all together to CCLK (counter clock).

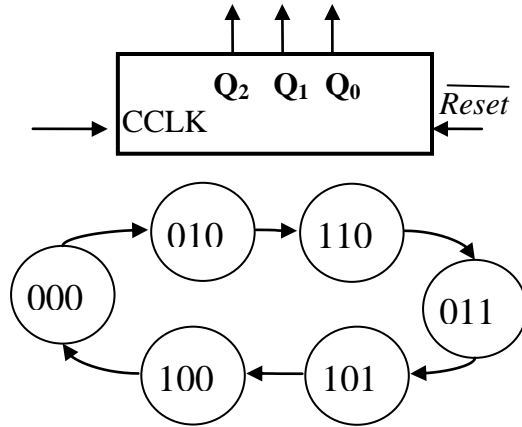


Fig. 1. (a) Block diagram and
(b) State Diagram of a Modulo 6 counter

Present State	Next State	Synchronous Inputs		
msb Q ₂ Q ₁ Q ₀ lsb	msb Q ₂ Q ₁ Q ₀ lsb	msb J ₂ K ₂	lsb J ₁ K ₁	lsb J ₀ K ₀
000	010	0 x	1 x	0 x
001	xxx	x x	x x	x x
010	110	1 x	x 0	0 x
011	101	1 x	x 1	x 0
100	000	x 1	0 x	0 x
101	100	x 0	0 x	x 1
110	011	x 1	x 0	1 x
111	xxx	x x	x x	x x

Table 1: The Excitation Table for the JK flip-flops Modulo 6 counter

- b. **4-bit synchronous BCD counter**


0000 --> 0001 --> 0010 --> 0011 --> 0100 --> 0101
--> 0110 --> 0111 --> 1000 --> 1001 --> 0000 --> ...


PART I (Design and simulation)

To capture your design in the Altera's development environment you can use Quartus' "New Project Wizard" or you can proceed manually with the following design flow.

For each circuit designed above (*3 bit modulo 6 and 4-bit BCD synchronous counters*):

1. Draw the circuit diagrams using the graphics editor of Quartus in a schematic file and save the corresponding .bdf file. Use the counter's signal names suggested in the block diagram (Fig. 1. a) when editing the names of the pins of your circuit. In the **Project Navigator** pane select the **Files** tab; right-click on your *schematic file* (.bdf) and select **Set as Top Level Entity**. Also save the schematics as a **jpeg** file or print it for inclusion in your report.
2. To assign **EMP7128SLC84-7** to your project go to **Assignments** in the main menu, select **Device** and in the window **Settings** chose **MAX7000S** for the **Device Family** and then from the list of **Available Devices** choose **EMP7128SLC84-7**.
In the main menu select **Processing** → **Start compilation** or click on the toolbar icon ►.

3. To visualize the input and output signals of your counter (clock, reset, flip-flops' outputs) you have to create a **vector waveform file** with the same name as the **.bdf** but with the extension **.vwf**, where you will catch the time diagram of these signals.
4. To define the set of pins of your **test circuit**, while in the **.vwf** tab, do select in the main menu **Edit >> Insert >> Insert Node or Bus...** and click on **Node Finder**.
5. In the option **Filter** of the popped-up **Node Finder** window choose '**Pins: All**', then click on the button **List** and move all **found nodes** (from the left list) to the right pane by clicking , then press OK to return to your **.vwf** file.
6. To set time characteristics of the simulation **clock** click on the clock signal (CCLK) to select it, then do **Edit >> Value>> Clock** and in the **Clock** window put a **Period** of 40 ns (a close approximation of the UP2 board clock) and click **OK**. Make sure you assign logic 1 to **Reset** by **Forcing High (1)**, to allow your counter operates under the CCLK control. At this point you can run a **functional** simulation. The binary representation of the counter's states can be displayed by **grouping Q₂ – Q₀** in a bus.
7. To choose a grid of 20 ns do: **Edit >> Grid Size**, then put 20 ns for **Period**.

Run your simulation (**Processing → Start simulation** or click on the toolbar icon ) and inspect the time diagram of your **Simulation Report – Simulation Waveforms** window and verify if your synchronous counter follows the given counting sequence; if it doesn't, verify your equations and/or debug your circuit. Show the simulation to your instructor and capture it in a graphic format for your lab report (copy to clipboard all the waveforms and paste them into a **.doc** file); to get a better visualization of your waveforms, you may want to change the time base in your **.vwf** file by choosing in **Edit/End Time** a **Time = 0.5 μs**).

6. In-Lab Procedure:

Show the design and demonstrate the simulations to your TA

PART II (Testing experiment)

Two (different speed) approaches can be considered to experimentally test the counters designed above:

- II-1. *Automatic, free running (highest speed)* - by connecting CCLK to the CPLD's general clock (GCLK1 - generated on board) and visualizing the flip-flops' outputs with an oscilloscope
- II-2. *Manual control* - by deriving CCLK from a push-button (MAX_PB1) and displaying the flip-flops' outputs on LEDs as shown below (block diagram of Fig. 2).

II-1. Automatic free running counter

In this mode you can visualize the signals of your real circuit by employing an oscilloscope. The real signals should be similar to the waveforms you obtained by simulating your circuit in PART I.

1. Return to your .bdf file and from **File >> Create/ Update >> Create Symbol Files For Current File** you can create a symbol (.bsf file) for your synchronous counter.
2. Now you can create a **test circuit** for the counter with the *Quartus graphic editor (test1.bdf)*. Save the file and set the project to the current file with **Set as Top Level Entity**. The *counter Symbol* created above can be inserted into your schematics by going **Edit >> Insert Symbol** and looking for it in the **Project** directory of the **Symbol Libraries**. The clock input of the counter (clk) has to be connected to the system clock (GCLK1 - from the on-board oscillator of 25.175 MHz) that is directly connected to the FPGA PIN 1 through the PCB (do not connect any other component to this pin!!!).
Remember to compile your file before assigning the pins!
3. Assign the EPM7128SLC84-7 device number to your design (Assign/Device) and then **assign pin numbers** as shown in Table 2a.

Function	CPLD Pin
GCLK1	83
Q ₃	80
Q ₂	81
Q ₁	4
Q ₀	5

Table 2a.

You can select other pins for your circuit but you have to avoid using the following dedicated pins, as shown in the next Table 2b.

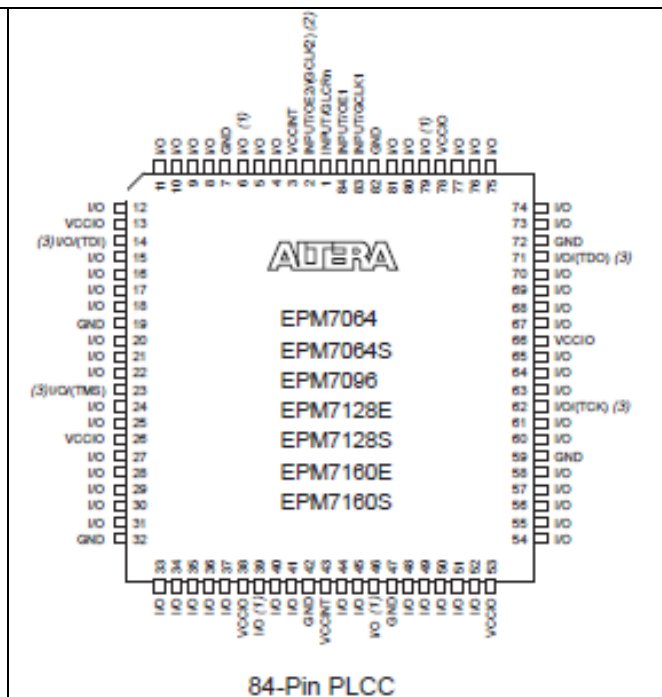
Dedicated Pin	84-Pin
INPUT/GCLK1	83
INPUT/GCLRn	1
INPUT/OE1	84
INPUT/OE2/GCLK2	2
TDI	14
TMS	23
TCK	62
TDO	71
GNDINT	42, 82
GNDIO	7, 19,
VCCINT (5.0 V)	3, 43
VCCIO (3.3 V or 5.0 V)	13, 26,

- a. To this extend, do **Assignments → Assignment Editor**; under **Category** select **Pin**. Double-click on the entry <<new>> which is highlighted in blue in the column labeled **To**.
- b. The drop-down list of the circuit's signals (**GCLK1, Q3, Q2, Q1, Q0**) will appear; click on **GCLK1** as the first pin to be assigned. Double-click on the box to the right of this **GCLK1** entry, in the column labeled **Location**, to drop-down the list of the device's pins. Scroll down and select PIN_83. Instead of scrolling down the menu to find the desired pin, you can just type the pin's number (83) in the **Location** box; redo the process for the rest of the pins.
- c. Compile your project with the assigned pins.

Alternatively, instead of the **Assignment Editor**, you can use the **Pin Planner**, from **Assignments** → **Pins**.

4. You can verify the operation of your circuit by defining another *vwf* file and running a simulation to catch the time diagrams in your associate *vwf* file. To assign the characteristics of the clock GCLK1 that you will use in the timing simulation, go to **Assignments** → **Timing Analysis Settings**. New **Settings** windows will pop up and there you have to select **Classic Timing Analyzer Settings** and click on **Individual Clocks**. In the **Individual Clocks** window click **New** and then give a name to **Clock Settings** and select GCLK1 from **Applies to node**. Re-compile your *.bdf* file; now you will be not given warnings referring to *undefined clock*.
5. Select the **Programmer** option in the main window of *Quartus II* and download the resulted configuration files to the Altera UP2 board through the **Byteblaster(MV)** cable (over LPT1) to the CPLD's JTAG interface. Don't forget to turn on the power to your board and check **Program/Configure** before you click **Start**.

6. Use an oscilloscope to visualize the clock and the flip-flops' outputs. Connect the probe of oscilloscope channel 1 to the m.s.b. of your counter and use it to trigger the oscilloscope. Use the probe of channel 2 to visualize the other signals and draw these time diagrams. Measure the rising and falling times of your signals



II-2. Manual control

- For this mode you have to derive the counter's clock CCLK from a push-button (MAX_PB1) and displaying the flip-flops' outputs on LEDs as shown below (block diagram of Fig. 2).

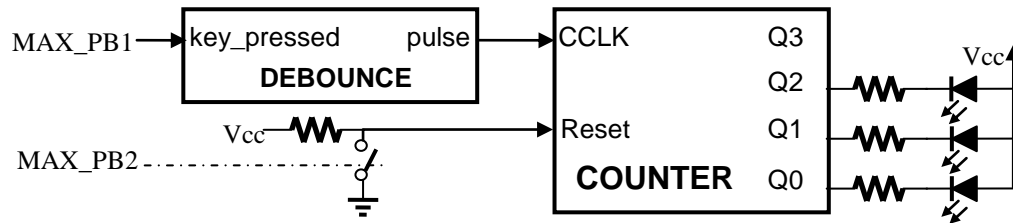


Fig. 2. COUNTER test circuit

1. MAX_PB1 and MAX_PB2 are two push-buttons on your UP2 board that provide active-low signals and are pulled-up through 10-KΩ resistors, as shown in Fig. 3. You will use MAX_PB1 to manually clock your counter.

When the contacts of any mechanical switch bang together, they rebound a bit before settling, causing bounce and generating an uncontrolled number of oscillations before coming to a still, as shown in the time diagram of Fig. 3.

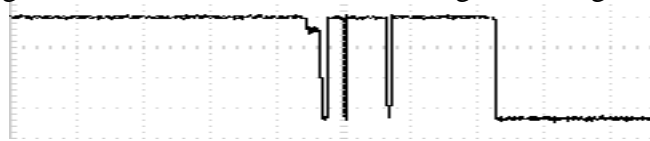
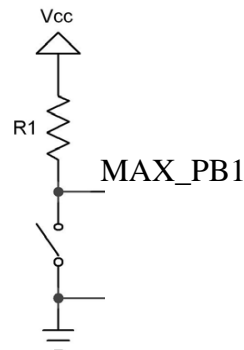


Fig. 3. High-to-low switch transitory oscillations

Debouncing is the process of removing those bounces and generating a single pulse (preferably synchronous with the system's clock), as soon as a steady stable contact is made.

Altera suggests to using module *debounce.tdf* that describes a debouncing circuit in a *Hardware Description Language*. Implementing it on your CPLD will generate a single **pulse** after a delay which is triggered by the rising-edge of any oscillation of the switch MAX_PB1 (i.e., at the end of pressing MAX_PB1); the delay duration has to be greater than the time between two parasitic pulses (Fig. 4).

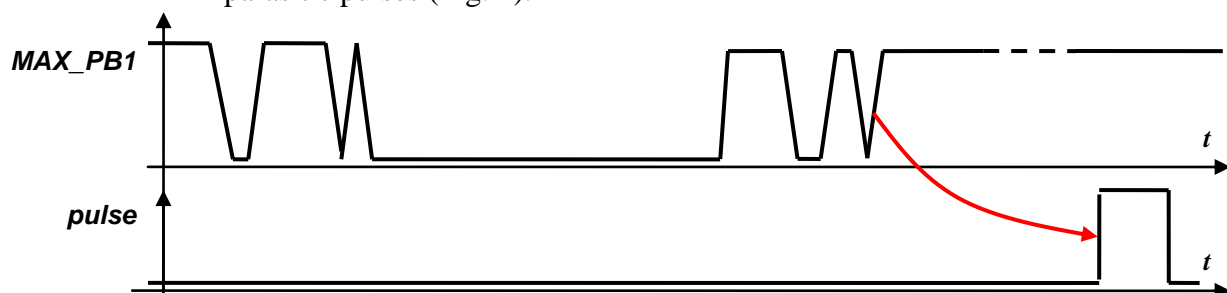


Fig. 4. Time diagram of single pulse generator *debounce* module.

debounce.tdf can be downloaded from the course *Virtual Campus* page before coming to the lab, or in CBY-B302 from the directory `c:\max2work\ahdl` of your station's computer (given that, for security reasons the browsers running in the lab doesn't allow to download anything on the PCs). Copy the *debounce* file to the working folder of your project (don't forget to remove its **read-only** property!), then do **Project >> Add/ Remove Files in Project...** and select *debounce.tdf*; click **Add** and then **OK**.

Open the file, do a compilation, and eventually create a default symbol for the debouncer: **File >> Create/ Update >> Create Symbol Files For Current File**. Alternatively, at the same effect, you can similarly use *debounce.vhd* given on the Virtual Campus, as well.

2. Now you can create another test circuit for the counter (as was shown in Fig. 2) with the *Quartus graphic editor (test.bdf)*. Save the file and set the project to the current file with **Set as Top Level Entity**. The *counter* and *debounce Symbols* created above can be inserted into your schematics by going **Edit >> Insert Symbol** and looking for them in the **Project** directory of the **Symbol Libraries**. The clock input of the debouncer (*clk*) has to be connected to the system clock (GCLK1 - from the on-board oscillator of 25.175 MHz) that is directly connected to the *FPGA PIN 1* through the PCB (do not connect any extra wire to this pin!!!). The "*key_pressed*" input of the debounce module is connected to the on-board pushbutton MAX_PB1. "*pulse*" is the debounced output that is used as a clock for the counter (CCLK). Remember to compile your file before assigning the pins!
3. Note that a LED is illuminated when current goes through it, i.e., when logic 0 is applied to the female header associated with the LED (Fig. 2). As we want to have an illuminated LED when the counter's corresponding output is High, the flip-flops' outputs should be inverted before sending them to LEDs.

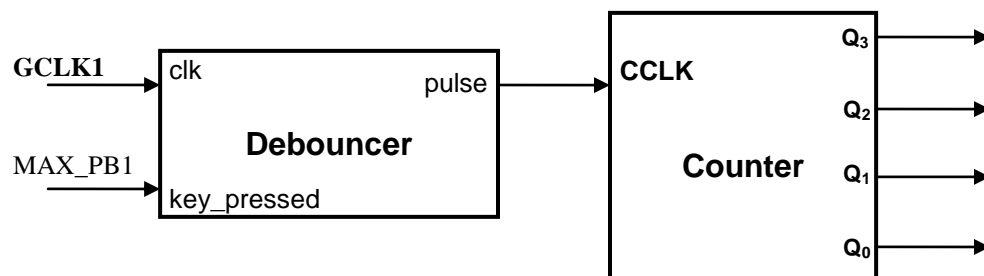


Fig 5 Test circuit

4. Assign the EPM7128SLC84-7 device number to your design (Assign/Device) and then assign pin numbers as shown in Table 3. Do **Assignments → Assignment Editor**; under **Category** select **Pin**. Double-click on the entry <<new>> which is highlighted in blue in the column labeled **To**. The drop-down list of the circuit's signals (**GCLK1, MAX_PB1, LED1, LED2, LED3, LED4**) will appear; click on **GCLK1** as the first pin to be assigned.

Double-click on the box to the right of this **GCLK1** entry, in the column labeled **Location**, to drop-down the list of the device's pins. Scroll down and select PIN_83. Instead of scrolling down the menu to find the desired pin, you can just type the pin's number (83) in the **Location** box; redo the process for the rest of the pins.

Compile your project with the assigned pins.

Function	Device	CPLD Pin
clk	GCLK1	83
key_pressed	MAX_PB1	11
Q ₃	LED1	80
Q ₂	LED2	81
Q ₁	LED3	4
Q ₀	LED4	5

You don't have to attach any wire to pin 83 since it is already connected to the local oscillator!!!

Table 3. General Pin Assignment.

5. Connect the LEDs and the push button to the CPLD's pins as shown in Table 2. Do not attach any wire to pin 83 since it is already connected to the local oscillator through the PCB.
6. Select the **Programmer** option in the main window of *Quartus II* and download the resulted configuration files to the Altera UP2 board through the **Byteblaster(MV)** cable (over LPT1) to the CPLD's JTAG interface. Don't forget to turn on the power to your board and check **Program/Configure** before you click **Start**.
7. Find experimentally the count table of your synchronous counter by pressing the **MAX_PB1** pushbutton until you rollover a full counting sequence. Verify that the output of your synchronous counter matches the corresponding the state diagram you were initially given.
8. In case your counter goes several counts when you press only once MAX_PB1, the *debouncer's* time constant has to be increased. This can be achieved by slowing down the *debouncer's* clock; to this effect insert a binary counter between GCLK1 and *debouncer's* "clk" (e.g., 74193 from `../quartus/libraries/others/maxplus2` in the **Symbol Libraries**). Link GCLK1 to the counter's "UP" input, and insert an inverter between its "CON" output and the debouncer's "clk" input. Normally such a 4-bit counter like 74193 would make it; verify experimentally if this is the case with your circuit.
9. Demonstrate the operation of your circuit to your instructor.