StopWatch Project

You will construct a simple stopwatch. A pushbutton on the DE-10 board, KEY (3) will be used as a control button and the 7-segmant displays will show the time. For DE-10 SoC board, a debounce circuit has already been implemented on board for all KEYS. We will use only four 7-segment displays for this design. As such, we assume that the accuracy of the stopwatch is 10ms such that the two high digits indicate the seconds and the lower 100ms are the 100ms and 10ms, respectively. To count at the 10ms interval, we need to generate a 100Hz clock. Therefor, the first component shown in the figure below is the "Clock-Gen" for the generation of this 100Hz clock. The counting will be done in decimal and therefore, we will design a 4-digit BCD counter as shown in the figure below. Note that the "clock" and "reset" of the "BCD_counter" are different from those for the "Clock_Gen" and the "Controller".

The "Controller" in the Figure is a FSM which detects the sequence of user input via the pushbutton and determine the action of the stopwatch. In the following, we will show the design of these individual blocks before finally demonstrating how to put everything together and describe them in VHDL codes.

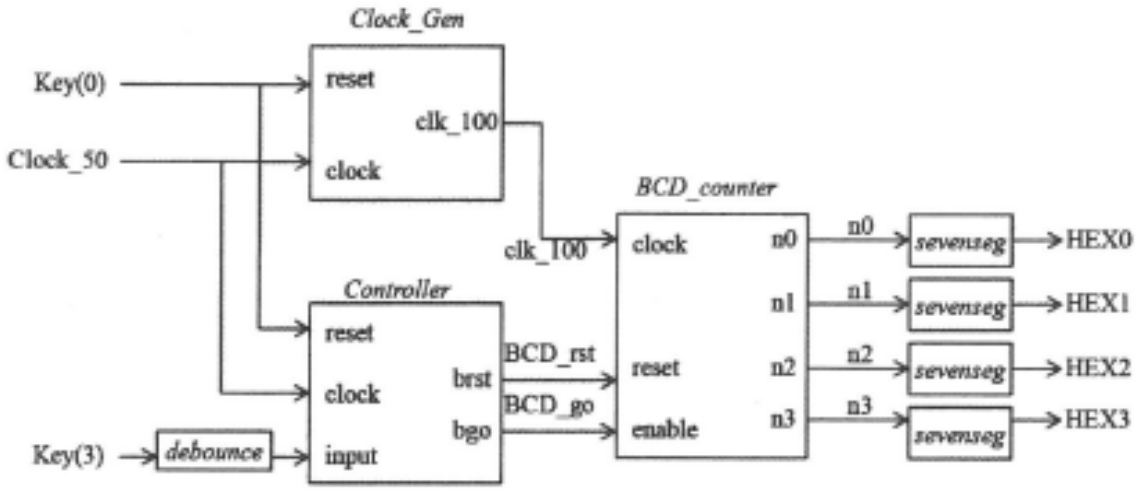

Fig. 1: Functional block diagram of the stopwatch design

Counter circuit is naturally a clock frequency divider or clock signal generator. For example, the four outputs of a 4-bit counter, as shown in Figure 4.11, can each be used as a clock signal. If the frequency of input "clock" is f MHz, then count(0) is a clock signal at *fl* 2 MHz, count(l) is at f/4 MHz, count(2) is at f/8 MHz, and count(3) is a clock at f/16 MHz.

For the stopwatch design, we need a frequency generator with a fixed frequency output to keep time. Here we will use a modulo-N counter for that purpose. A modulo-N counter is a counter that counts from O to N-1. Usually, a pulse signal is generated when the count goes from N-1 to 0. Such pulse signal will occur only once every N cycles and thus at f/N MHz.

The controller

In this design, the user operates the stopwatch with just one pushbutton: KEY(3). After initialization, the stopwatch display should read "0000". To initiate the stopwatch, the user will press and hold the pushbutton. Once the pushbutton is released, the counting stops and the stopwatch stays at its final time count. After that, the user may clear the display by pressing the pushbutton and releasing it. The stopwatch essentially goes back to its initial state after the display is cleared. KEY(0) is used to emulate the power-up reset. It is **easy** to get confuse when we are using other pushbutton as a user input device. Nevertheless, KEY(0) is not part of the input device and does not involve in circuit behavior. From the original design shown in Figure 1, we see that the main function of the "controller" is to produce two control signals for the "BCD_counter". The signal "brst" will be used by the "BCD_counter" as its "reset" signal. The signal "bgo" will be used by the "BCD_counter" as its counting enable signal. **We** may summarize the functionalities of these control signals as in Table 1.

Table 1: Functions of control signals: “brst” and “bgo”

brst	bgo	BCD_counter
0	0	No counting; display shows the last time count
0	1	Counting; display shows the running time count.
1	0	At reset, display = x"0000"
1	1	At reset, display = x"0000"

Based on the design of "BCD_counter" and its definitions of the two control signals as shown in Table 1, we may construct a FSM implementing the given design specifications. The FSM should be initialized so that "brst" = '1' and "bgo" = '0'. The stopwatch display is x"0000" and no counting. When the user presses KEY(3), or "input" = '0', we should immediately provide "brst"= '0' and "bgo" = '1'. This will take "BCD_counter" out of the reset state and start counting. When the FSM

detects that KEY(3) has been released, or "input" = '1', the control signals should change to "brst" = '0' and "bgo" = '0'. This stops the "BCD_counter" from counting but did not reset it. The BCD displays will remain at the last time count value just when KEY(3) was released. Finally, when the user presses KEY(3) again, the FSM changes the control signals back to "brst" = '1' and "bgo" = '0', same as just after the initialization. Based on the behavioral description in Table 1, you should get:

1. Flowchart of the Stopwatch controller.
2. State diagram of the Stopwatch controller.

The controller receives only three inputs: ‘clock’, ‘reset’, and ‘input’, and produces two outputs: ‘brst’ and ‘bgo’, will work on the “BCD_Counter”, and thus provides the functional information.