UNIVERSITI TEKNOLOGI MALAYSIA

Simple Input/Output with CPLD

Muhammad Mun'im Ahmad Zabidi, Musa Mohd Mokji, Izam Kamisian, Norhafizah Ramli

Abstract
Reconfigurable devices such as Complex Programmable Logice Device (CPLD) and Field Programmable Gate Array (FPGA) work much faster than humans. Interfacing a fast digital circuit require special considerations when input/output involves human. This documents explores several considerations when designing human interfaces.

[1] *Department of Electronic and Computer Engineering, School of Electrical Engineering, Faculty of Engineering*
*__Corresponding author__: munim@utm.my

1. Basic Digital Input

A digital input pin can take one of these states: high, low and floating or high impedance. When an input is driven above the high threshold, it is a logic 1. When the input is driven below the low threshold, the input is logic 0. When the input not driven or is in high impedance state, the input level in a floating state. The floating state is may cause your circuit to behave erratically, so it has to be prevented.

To ensure that input values are always in a known state, pull up and pull-down resistors are used. The key function of pull-up and pull-down resistors is that the pull up resistor pulls the signal to high level unless it is driven low; and, a pull-down resistor pulls the signal to low level unless it is driven high.

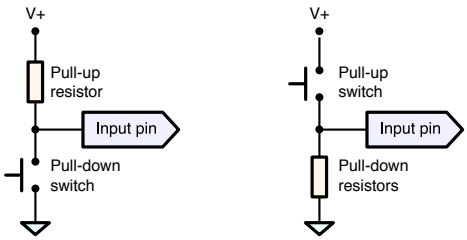

Figure 1. *Location of pull-up and pull-down resistors in relation to the input switch.*

With a pull-up resistor and with the button unpressed you make a logic state ON and with the button pressed you make a logic OFF. In Fig. 2, you apply the resistor to resist current flow. When the switch is locked, the current will take the path of least resistance to the CPLD pin, giving a signal as you want to do. You need to be connected to the ground as a reference point, and the resistor also avoids short circuit power-to-ground.

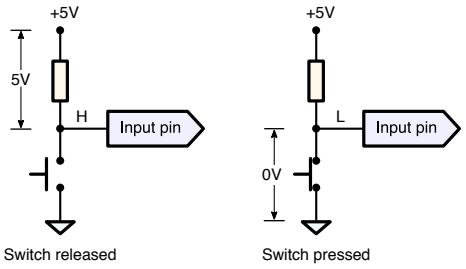

Figure 2. *How pull-up resistor works.*

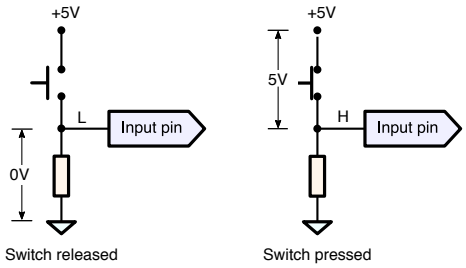

Figure 3. *How pull-down resistor works.*

What should be the value of the resistor?

So lets assume you want to limit the current to 1mA. Assuming Vcc = 5V, using Ohms law:

$$\begin{aligned} R &= \frac{V}{I} \\ &= \frac{5000mV}{1mA} \\ &= 5000\Omega \\ &= 5k \end{aligned}$$

so a resistor of 4k7 will be fine. Even 10kΩ resistor is fine since this will supply 0.5mA to the input port.

With a pull-down resistor and a pressed button you make an ON logic state and OFF logic state when its unpressed. Fig. 3 shows how the pull-down resistor works. The calculation for the resistor is similar.

2. Basic Digital Output

Digital display is the easiest regulation that can be used over an electrical interface. In this scenario, you'd either turn something off or turn it on. Fig. 4 is a digital output that controls the LED:

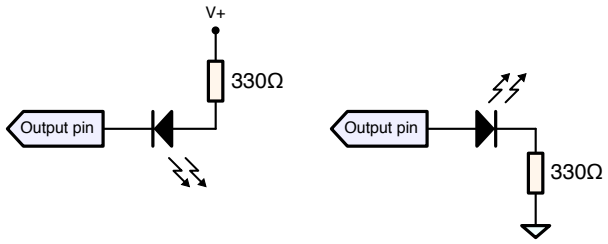

Figure 4. *Simple digital output.*

Digital outputs are also used to power other electrical equipment, such as transistors or relays. More information on these elements will come later.

3. Prescalers

Digital circuits work much faster than humans. A low cost reconfigurable device such as the Altera MAX II CPLD works at 50 MHz so their outputs must be slowed down significantly to have any meaning to humans.

A prescaler is a circuit which uses the system clock to derive a slower frequency by integer division. The slow clock is then used as the timing reference for input/output operations. Fig. 5 illustrates an accumulator based prescaler which generates a 1 Hz slow clock from a 50 MHz system clock. The basis for the prescaler is the 25-bit counter which counts from 25 million down to 0.

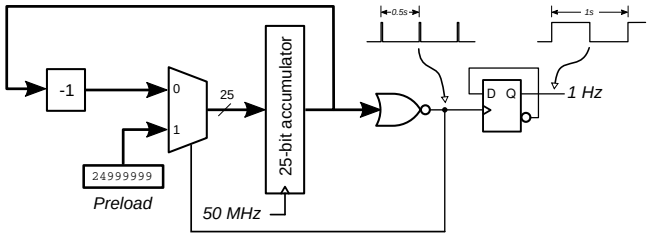

Figure 5. *A 50,000,000:1 prescaler.*

When the counter reaches 0, the NOR output becomes high for 1 cycle. Two things happen. One, toggles the output of the D flip-flop. This output of the D flip-flop is then becomes 1 Hz signal with a 50% duty cycle. Two, the counter is reloaded with the constant 24,999,999.

Using Quartus, we can use the built-in **lpm_counter** module from the Library of Parameterized Modules. To create the prescaler in the Block Diagram Editor, follow the steps below.

1. Click anywhere in the editor workspace. The new symbol dialog box, enter **lpm_counter** in the text box then click OK.

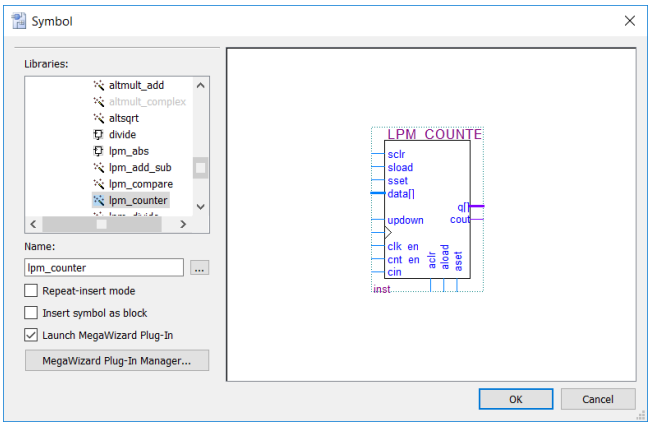

Figure 6. *In the New Symbol dialog, enter* ***lpm_counter****.*

2. In Page 2c of the MegaWizard, click Next >.

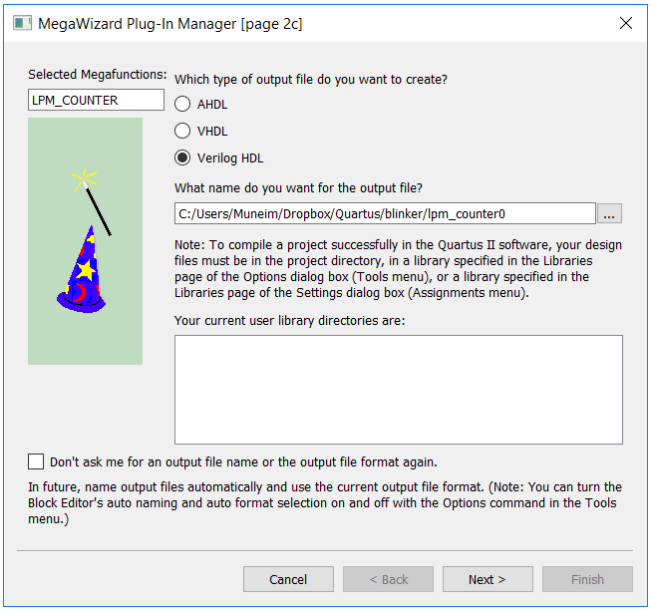

Figure 7. *Page 2c of MegaWizard.*

3. In Page 3 of the MegaWizard, select size of output to **26**, and counter directon to **Down only**.

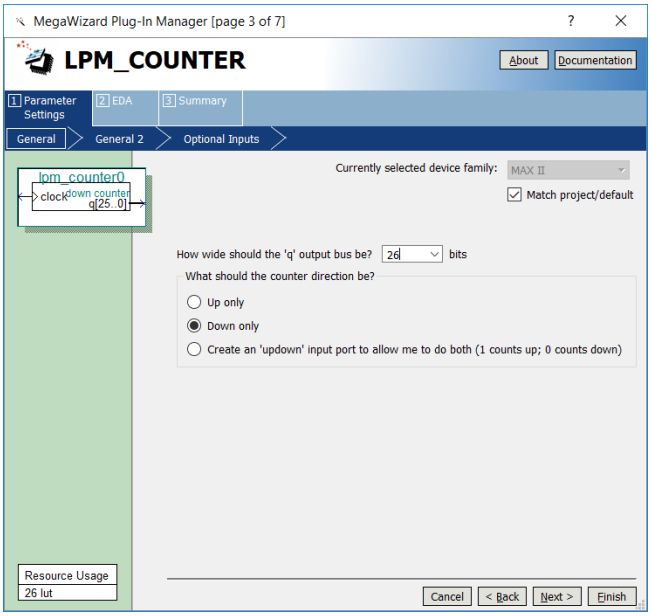

Figure 8. *Page 3 of MegaWizard.*

4. In Page 4 of the MegaWizard, add a **Carry-out** output port.

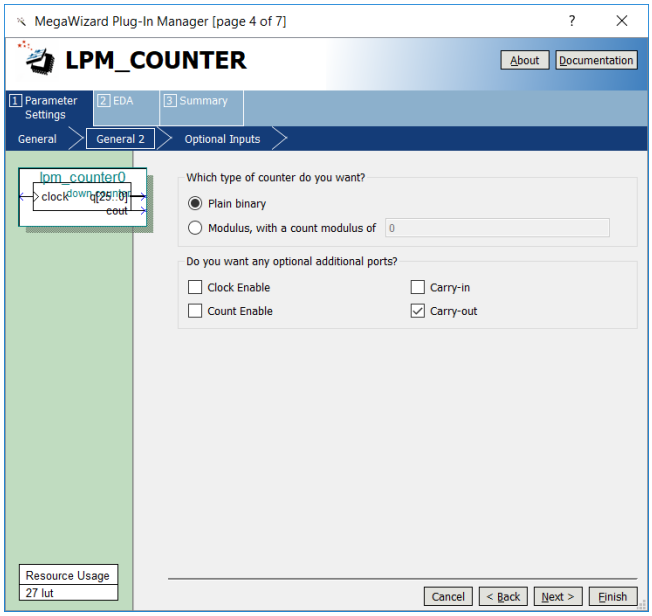

Figure 9. *Page 4 of MegaWizard.*

5. In Page 5 of the MegaWizard, add a **Synchronous Load** output port. Click Finish.

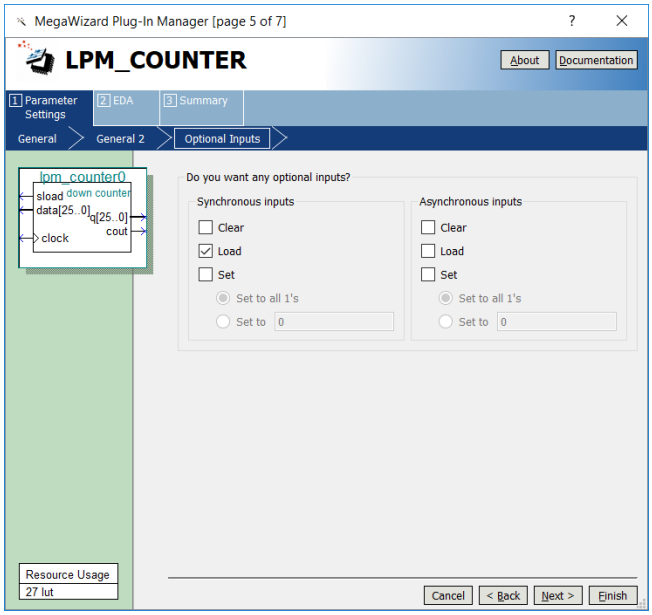

Figure 10. *Page 5 of MegaWizard.*

6. In Page 6 of the MegaWizard, just click Finish.

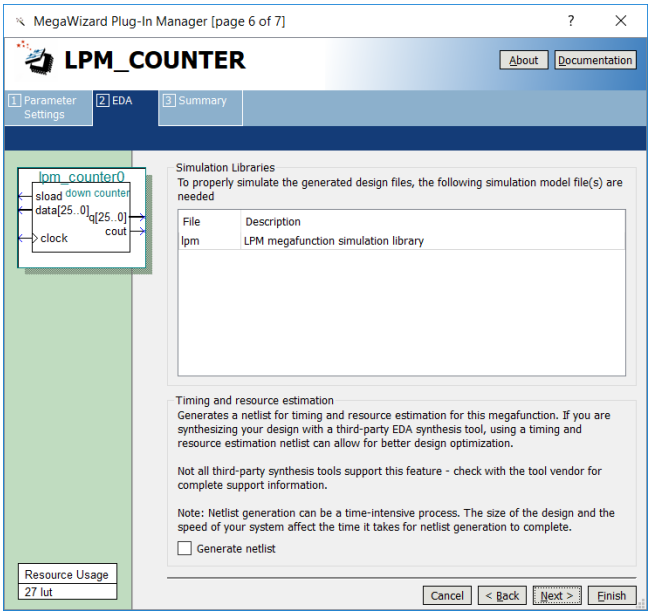

Figure 11. *Page 6 of MegaWizard.*

7. In Page 7 of the MegaWizard, click Finish again.

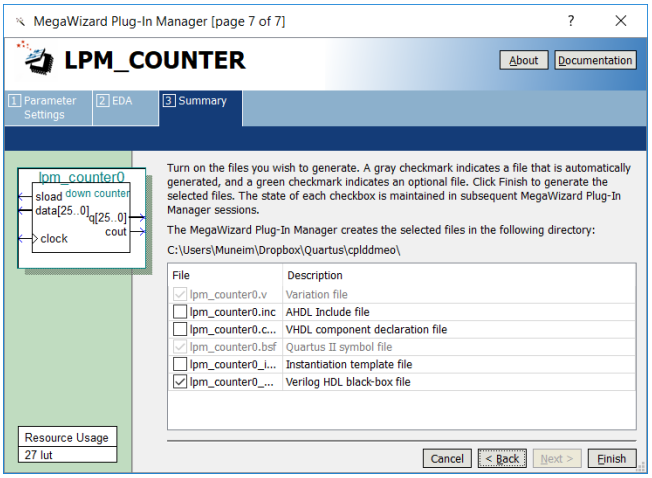

Figure 12. *Page 7 of MegaWizard.*

8. Place the generated counter symbol anywhere on the workspace.

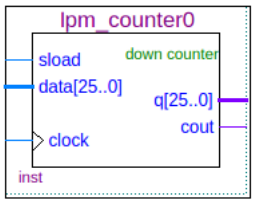

Figure 13. *Generated lpm_counter symbol.*

9. Double-click anywhere to get the New Symbol dialog, then enter **lpm_constant**.

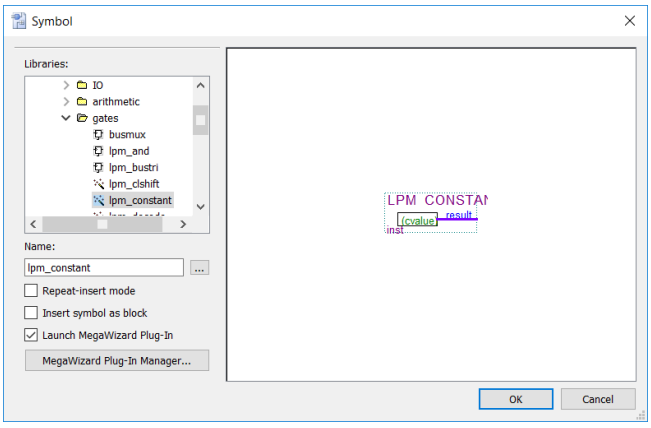

Figure 14. *In the New Symbol dialog, enter* ***lpm_constant****.*

10. In Page 2c of the MegaWizard, click Next >.

11. In Page 3 of the MegaWizard, select output width of 26 bit and constant value of 24999999 (25 million minus 1) decimal. Then click Finish.

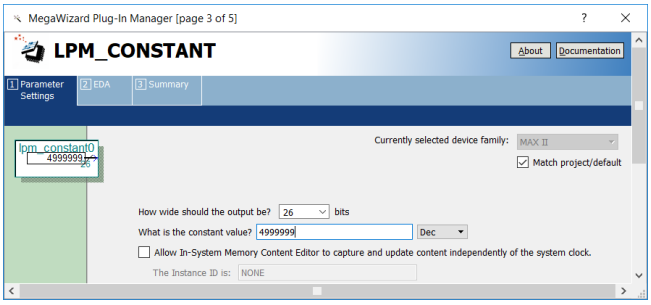

Figure 15. ***lpm_constant*** *parameter settings.*

12. In Page 7 of the MegaWizard, click Finish again.

13. Double-click anywhere to get the New Symbol dialog, then enter **tff**.

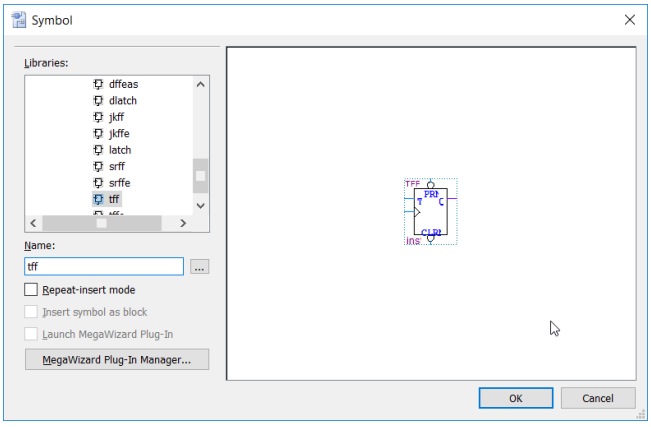

Figure 16. *Insert a TFF.*

14. Click OK and place the T flip-flop anywhere on the workspace.

15. Make the connections as shown in Fig. 16.

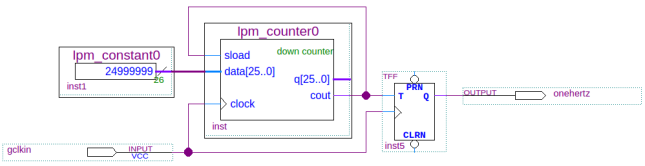

Figure 17. *Prescaler module to generate 1 Hz time base.*

16. Save the design. Compile.

17. For design reuse, convert to Symbol File. From the top menu choose **File ➥ Create/Update ➥ Create Symbol Files from Current File**. Give it is descriptive short name. Recommended: **prescale1hz**.

4. Timer

A timer is a circuit which generates an output for a specified duration, usually in terms of seconds or fraction of seconds. Fig. 19 illustrates the general idea of a timer which produces a pulse length of one second.

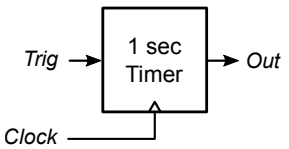

Figure 18. *Top level diagram of 1 second timer.*

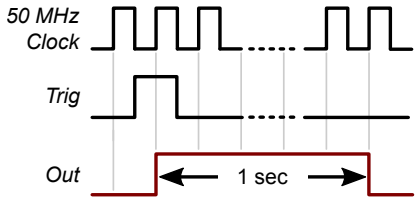

Figure 19. *Timer with one second pulse length.*

Fig. 20 depicts how the one-second timer may be implemented. A 26-bit accumulator-based counter is preloaded with 50,000,000 when the triggering signal, *Trig*, is received. The same *Trig* signal sets the output of the SR latch to high. The counter is then decremented 50 million times until it reaches zero. The accumulator condition of zero is detected by the NOR gate which resets the output of the SR latch.

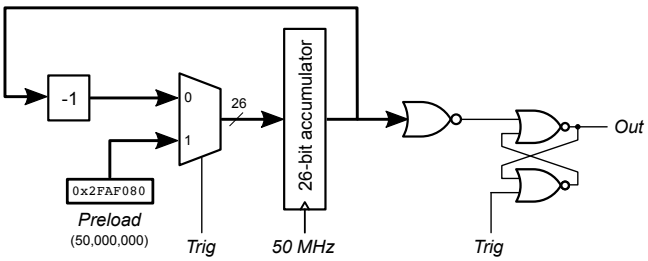

Figure 20. *One-second timer schematic.*

If the 26-bit accumulator is reloaded and restarted every time it reaches zero, the NOR gate would generate a pulse every second. If the pulses is used to enable the movement of another counter, a timer with a selectable pulse length is possible. This is the idea suggested in Fig. 23.

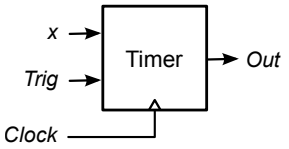

Figure 21. *Top level diagram.*

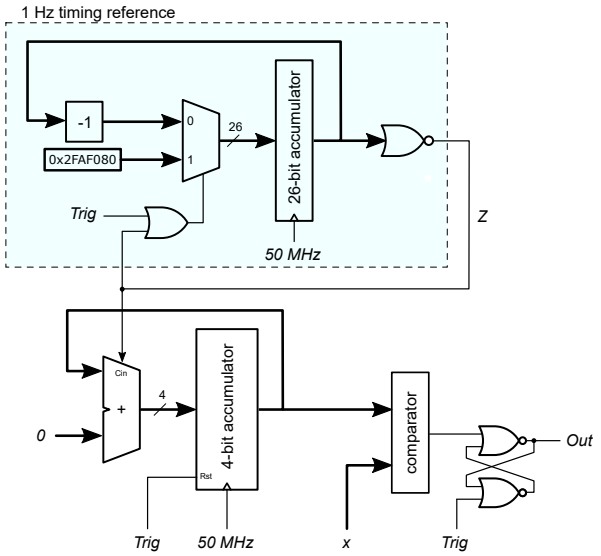

Figure 24. *Programmable timer.*

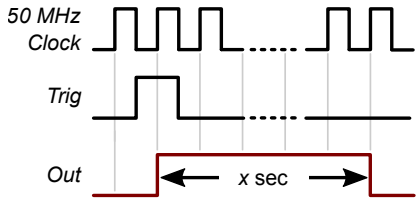

Figure 22. *Timing diagram.*

Figure 23. *Timer with selectable pulse length.*

The timer consists of two major blocks. The top block it the 1 Hz timing reference which generates a 1 Hz pulse train labeled Z. This pulse appears whenever the accumulator contains zero. This circuit is identical to the one second timer discussed with the exception of the SR latch.

The bottom block contains an upcounter based on a resettable 4-bit accumulator. When *Trig* is received, the accumulator is cleared. The accumulator receives its input from an adder which adds 1 to its input when Z is high. This results in the 4-bit accumulator being incremented every second. The output of the accumulator is fed to a comparator which outputs 1 when the accumulator equals x, which is the target pulse length. The final part of the puzzle is the SR latch which goes high when *Trig* is received, and goes low when the comparator outputs 1. The sum effect of the components produces the timing diagram shown in Fig. 24.

The length of the output pulse can be easily set to anywhere from 1 to 15 clock cycles by changing the x input. This input can be hardwired meaning it is fixed to a certain value, or x can be made 'programmable' by deriving the preload value dynamically from another circuit.

5. Key Bounce

Users often interface with digital circuits using pushbutton switches. Contact bounce is a common problem with mechanical switches. When a switch or button is toggled, contacts have to physically move from one position to another. When two contacts of the switch hit together, they mechanically bounce back and forth and settle down only after a small time delay (about 5-10ms).

Because a digital circuit is so fast, it will record multiple signal assertions from the single button push. This problem is called **key bouncing** as depicted in Figure 25. In finite state machine, key bounce is a problem because a single key push is counted as many and causes the circuit to behave erratically.

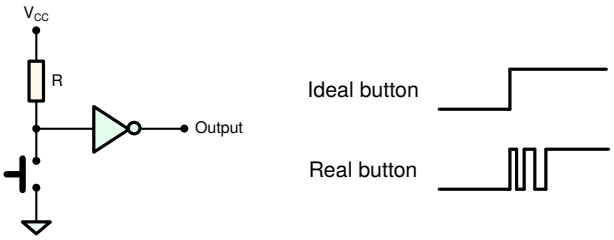

Figure 25. *Key bounce problem.*

The solution to this problem is called **debouncing**. Debouncing can be done in many ways. One may, for instance, use special input circuitry on the switch such as the one shown in Fig. 26. However, this solution is not cost effective as double-throw switches are not easy to find. Another solution is to use analog components (capacitor and diodes) to remove the glitches.

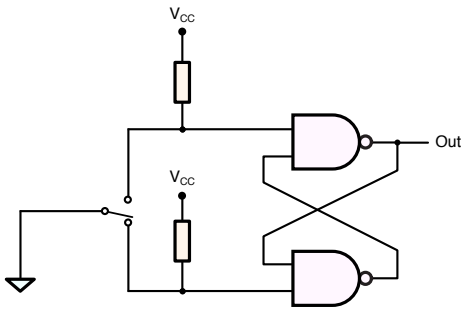

Figure 26. *A debouncing circuit based on $\overline{S}\overline{R}$ latch.*

It may be attrative to perform the debouncing completely by digital means. The method illustrated in Fig. 27 uses a slow clock to sample the pushbutton. When not pressed, the switch gives a high logic value. A long keypress generates the sequence 111100000... with the transition 100 occuring when the switch is pressed. The D flip-flops detect the 100 sequence causing the AND gate to generate a 10 ms long pulse. A level-to-pulse converter then shortens the pulse to exactly one period of the system clock.

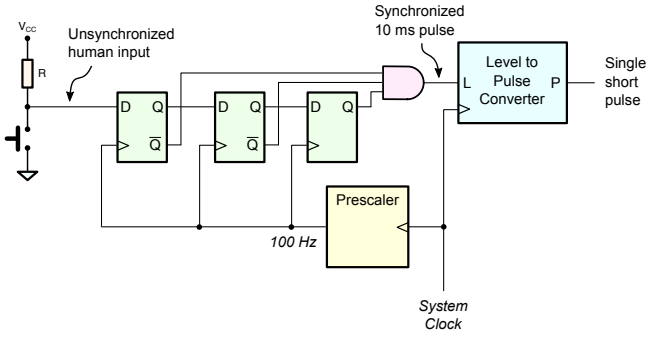

Figure 27. *Debouncing using a slow clock.*

A 100 Hz slow clock is generated by a prescaler shown later in this application note. The slow clock causes the input to be sampled once every 10 ms, thus any key bounce is assumed to be over from one sample to the next. As shown in Fig. 28, the AND gate produces a intermediate 10 ms pulse. The AND gate is not strictly required as a similar final output can be obtained by connecting QN of the second FF to the level-to-pulse converter.

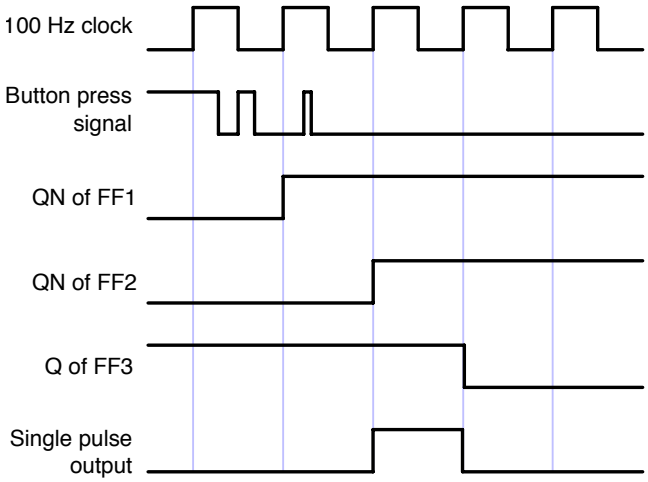

Figure 28. *Timing diagram for debouncing circuit with slow clock.*

Another digital method to is deactivate the switch for a specified length of time after the first contact is made. This method enables faster response as the input is processed as soon as it received; it does not wait for the next edge of a slow clock. A finite state machine can be designed to do nothing but wait for the input pin to go low and then it "pauses" for 10 ms before continuing. Any switch bouncing is assumed to have died down once pause interval is over. This method requires the aid of a timer circuit. The drawback of this method is the need for an independent timer for each switch, which requires considerable resources when many switches are present.

Acknowledgments

Thanks to the supervisors of digital lab 2018/2019-2 for constructive comments.

References

[1] Munim Zabidi, Izam Kamisian, and Ismahani Ismail. *The Art of Digital Design*. 2019.

[2] *Introduction to the Quartus® II Software*. Version 10. Altera. 2010. URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/manual/intro_to_quartus2.pdf.

[3] *My First FPGA Design Tutorial*. Altera. July 2008. URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/tt/tt_my_first_fpga.pdf.