Experiment 8. Clock

Masud ul Hasan · Ahmad Khayyat – Version 151, 9 April 2015

Table of Contents

1. Objectives

- Learn about the clock signal and clock frequency
- Generate slower clock signals from a faster one
- Learn how to use oscilloscopes

2. Material Required

- An FPGA prototyping board.
- Design and simulation software tools.
- An oscilloscope.

3. Background

3.1. The Clock Signal

Sequential circuits are controlled by a periodic control signal, called the *clock*. The clock determines when memory elements of a digital circuit change their states. The state of a memory element may change either at the rising edge or at the falling edge of the clock signal, as illustrated in the clock signal figure. The clock signal is generated by a device called the *oscillator*.

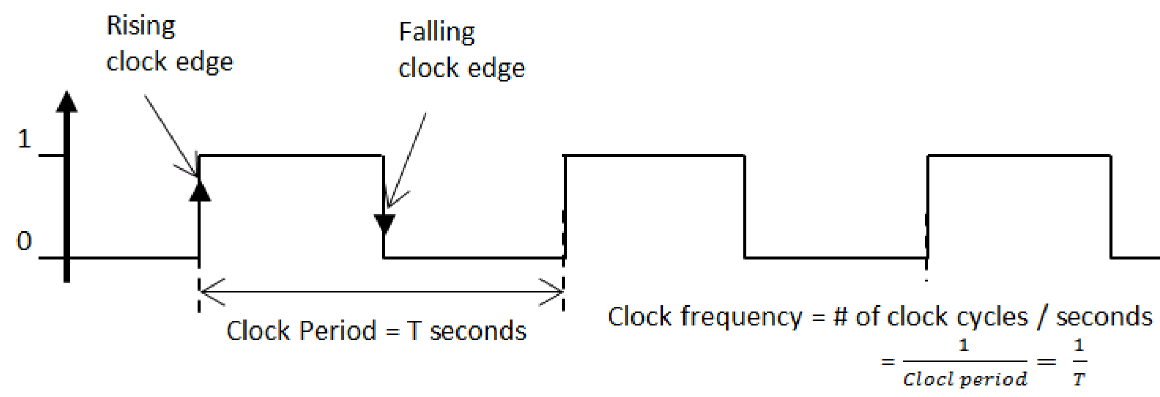

Figure 1. A Clock Signal

The Xilinx Nexys 3 FPGA board includes a clock oscillator with a frequency of 100 MHz (10 ns period). This clock is too fast for a human observer. Thus, a circuit is needed to reduce the clock frequency to lower frequencies. In this experiment, you will design a *clock division circuit* that reduces this frequency to different slower frequencies. You will observe the clock signal using an electronic test instrument called the *oscilloscope*.

3.2. Using the Oscilloscope

A typical oscilloscope is shown in the oscilloscope figure. A probe, connected to the oscilloscope from one end, allows you to monitor a signal by placing its other end on a terminal. The signal at that terminal is then visualized on the oscilloscope's screen. In this experiment, you will monitor an output pin on your board to see the 100 MHz clock signal of the on-board oscillator.

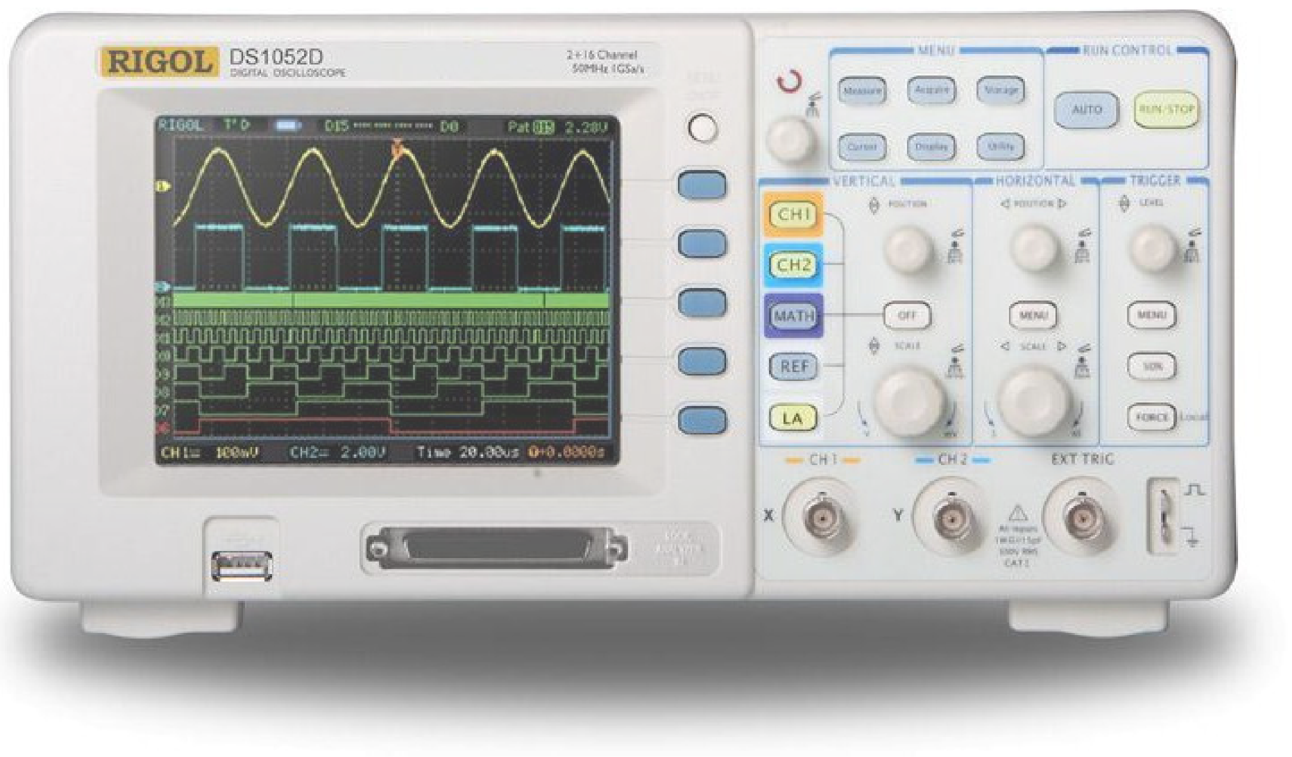

Figure 2. An Oscilloscope

To minotor a signal in your design using the oscilloscope, you need to expose the signal on a terminal that can be probed. A good candidate for such terminals is the Pmod connectors on the FPGA board. Pmod, or the *Peripheral Module* interface, is a standard defined by Digilent Inc. for peripherals used with FPGAs or microcontrollers. The Pmod connectors on our FPGA board are shown in the Pmod Connectors figure.

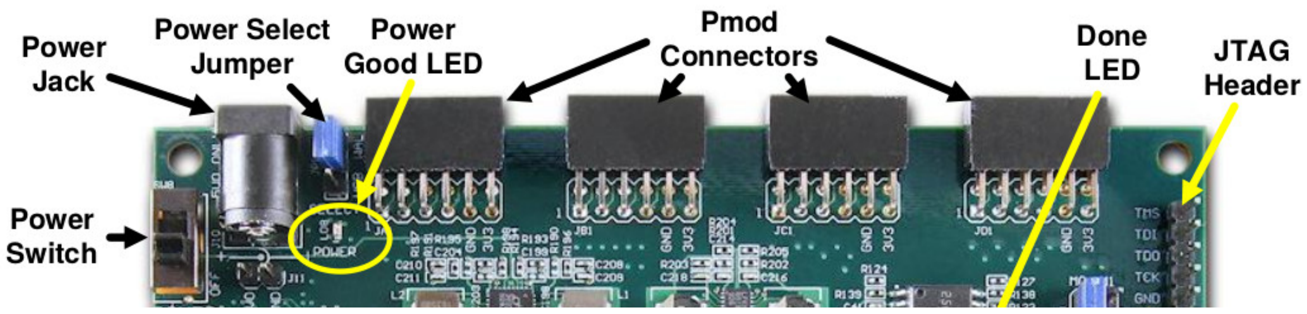

Figure 3. Pmod Connectors

Each 12-pin Pmod connector provides two 3.3V VCC signals (pins 6 and 12), two Ground signals (pins 5 and 11), and eight logic signals. To use a Pmod connector to monitor a signal, connect your signal to one of the data pins.

To learn about the pin configuration of the Pmod interface, and the required pin assignments to connect the Pmod pins to your design, refer to the *Pmod Connectors* section of the Nexys3 Board Reference Manual [nexys3].

3.3. Frequency Division

Consider a 2-bit counter (also known as a mod-4 counter). The counting sequence is shown in the Clock, Q_0, and Q_1 Signals figure. In the figure, the period of the Q_0 signal is double the period of the clock signal, i.e. $f_{Q_0} = \frac{1}{2T} = \frac{f}{2}$, where T and f are the period and frequency of the clock signal, respectively. Similarly, the period of the Q_1 signal is four times the period of the clock signal, i.e. $f_{Q_1} = \frac{f}{4}$. Thus, we can effectively use Q_1 as a clock that is four-times slower than the original clock signal, as shown in the Clock, Q_0, and Q_1 Signals figure.

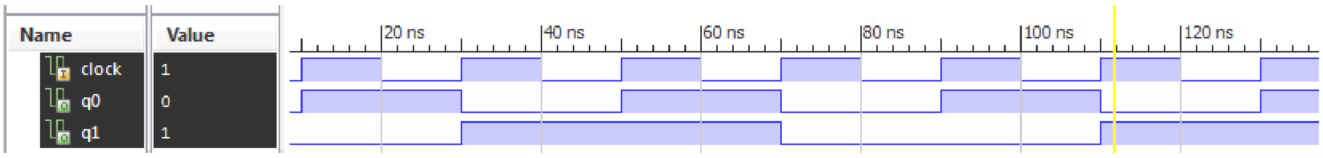

Figure 4. Clock, Q_0, and Q_1 Signals

In general, if we use the original clock to drive a counter, then the *n*-th bit of the counter will act as a clock signal with a period that is $2^{(n+1)}$ times the original clock's period. That is,

$$\text{Desired frequency} = \frac{\text{Original frequency}}{2^{n+1}}$$

Accuracy of Generated Clock Frequency

Because we obtain a new clock frequency by dividing the frequency of an existing clock by 2, we do not have full control over the exact frequency of the generated clock. For example, if your original clock is 4 Hz, you cannot use this technique to obtain a 3-Hz clock.

Nonetheless, this technique is effective for generating approximate frequencies from relatively higher-frequency clock signals.

You will learn how to generate pulses with precise frequencies in the next experiment. In the meantime, can you think of a way?

Exercise

- How wide of a counter do we need to generate a clock with an approximate frequency of 1.5 Hz from the 100-MHz clock on the FPGA board?
- If bit *n* of that counter provides a 1.5 Hz clock, how will bit *n* – 1 behave?

3.4. Implementing a Parameterized Counter

Consider a binary counter component with the following parameterized Verilog implementation:

Counter Module with Synchronous Clear and Parameterized Width

```
module counter #(parameter WIDTH=16) (
    input  wire             clock, reset_n, enable,
    output wire [WIDTH-1:0] count);

    reg [WIDTH-1:0] count_reg = 0;
    assign count = count_reg;

    always @(posedge clock)
        if (!reset_n)
            count_reg <= 0;
        else if (enable)
            count_reg <= count_reg + 1;
endmodule
```

Default Port Type in Verilog

In Verilog, the default port type is `wire`. Therefore, it is common to omit the type declaration when the desired type is `wire`.

For example, in the counter module definition above, `wire` in the `input` and `output` port definitions is optional.

It is, however, good practice to explicitly specify the type to make your code more readable and less ambiguous.

Upon instantiating this module, the counter's width can be overridden to create an arbitrarily-wide counter:

Instantiating a 20-bit Counter

```
counter #(.WIDTH(20)) count_20 (
    .clock   (clock),
    .reset_n (reset_n),
    .enable  (count_enable),
    .count   (count_value));
```

Verilog code for commonly used functions and language constructs can be obtained using the Xilinx ISE's *Language Templates* facility, in the *Edit* > *Language Templates* menu item.

4. Tasks

4.1. Observe the Main Clock Signal

1. Create a minimal top-level Verilog module in which the main clock input is connected to a Pmod pin.

   Review the Using the Oscilloscope section for information about the Pmod connectors.

2. Use the oscilloscope to monitor the Pmod pin configured to output the clock signal.
3. Describe the shape of the signal and report the clock's frequency as observed on the oscilloscope.

4.2. Build a Digital Frequency Divider

1. Design a frequency divider Verilog module that uses the 100-MHz input clock to generate a clock signal whose frequency is approximately 1 Hz.

   In addition to the 1-Hz clock signal, the module should also output any signals with intermediate frequencies that are required to generate the 1-Hz clock.

2. Simulate your frequency divider using a 100-MHz input clock. Verify that the outputs are slower clock signals.
   a. What is the frequency of the fastest output signal?
   b. What is the relationship between the frequencies of output signals?
3. Implement your frequency divider on the FPGA board, and connect the original clock signal as well as the fastest three generated clock signals to four different Pmod pins.

   Use the two probes of the oscilloscope to observe and compare the original clock signal to each of the three generated clock signals.

The board's clock signal is available at pin `V10` on the FPGA. Use it as your input clock signal.

4.3. Build a Frequency Selector

1. Design a Verilog module that does the following:
   a. Uses your frequency divider to generate four clock signals with frequencies of 1 Hz, 2 Hz, 4 Hz, and 8 Hz.
   b. Accepts a two-bit input that selects one of the four frequencies.
   c. Outputs one of the four generated clock signals for each input combination.
2. Implement your design on the FPGA board, connecting the two inputs to switches, and the output clock to an LED.
3. Verify that your design is working correctly on the FPGA board.

5. Grading Sheet

Task	Points
Observe the 100-MHz clock using the Oscilloscope	20
Compare divider outputs to the original clock (simulator and oscilloscope)	30
Implement a frequency selector	30
Discussion	20

6. Resources

[nexys3]

Digilent Inc. *Nexys3 Board Reference Manual.* April 3, 2013. Doc: 502-182. https://www.digilentinc.com/Data/Products/NEXYS3/Nexys3_rm.pdf

Version 151
Last updated 2016-04-09 20:34:49 AST