UNITED STATES
AIR FORCE
ACADEMY

ECE 281

Lesson 22 Notes

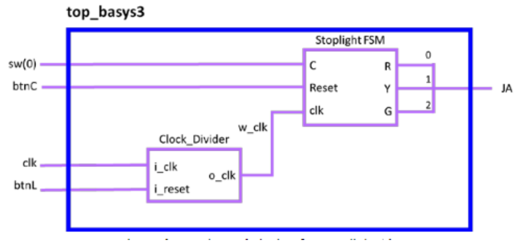
Objectives:

1. Understand how to correctly use and interpret the VHDL constructs within a process
2. Demonstrate the ability to correctly write or analyze a process to describe basic sequential logic such as a flip flow, latch and register
3. Understand the difference between asynchronous and synchronous resets and how to describe them in VHDL

Review

Up to this point, we have primarily built combinational logic inside of the FPGA. However, during today's ICE, we are going to start working with sequential logic in order to implement a finite state machine. The particular finite state machine that we will build is a stoplight. Before we get started, let's review the major structural components in our VHDL code.

First of all, what do we do in the **entity**:



```
entity                    is

        port(





);
```

What do we do in the **architecture**:

Now, let's introduce a few new concepts that will be required for your sequential logic implementation. We will handle this with a process. The process is largely similar to previous coding you may have done, but in this class we are going to insist on the following:

1.
2.
3.

Before we go any further, let's review the difference between asynchronous and synchronous logic:

Synchronous – the state of the device changes only

Asynchronous – the state of the device can change at

Let's first use a process to describe **Asynchronous logic.** Assume Q is a 4-bit vector. If the reset flag is set to 'high', we want to reset Q to all zeros. Otherwise, we want to set Q to some other value stored in the register 'D'.

```
async_example : process (          )

begin










end process;
```

Now let's use a process to describe **Synchronous logic** with similar results**.** Assume Q is a 4-bit vector. If the reset flag is set to 'high', we want to reset Q to all zeros. Otherwise, we want to set Q to some other value stored in the register 'D'.

```
sync_example : process (                  )

begin










end process;
```