Jose Sotelo

CECS 460: System on Chip Design

Project 1 – Seven Segment Counter Display with TramelBlaze Processor

February 8, 2018

Introduction:

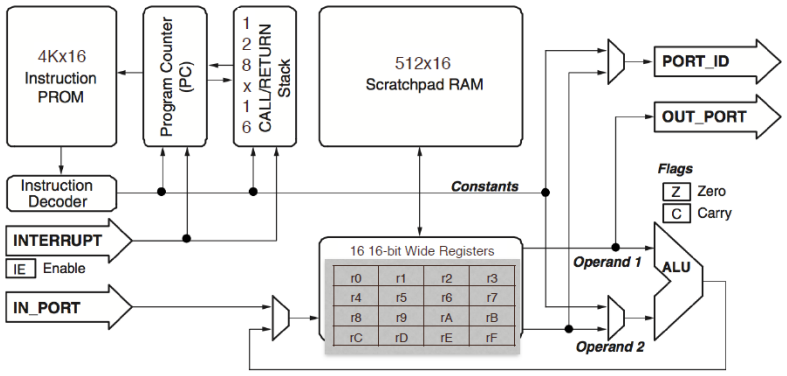
The purpose for this project is to review topic covered in 201, 301 and 360. We will be showing proper design practices and techniques to design a synchronous counter and implement a processor, the TramelBlaze to display values on the 7-Segement display. We will be using the TramelBlaze, a 16-bit processor core design to emulate the 8-bit Xilinx PicoBlaze to fetch and execute instructions. This project consists of implementing modules design in the previous semester which are: Asynchronous in Synchronous out circuit (AISO), debounce Finite State Machine circuit, Pulse Edged Detector (PED), a 7-Segment Display and replace the 16-bit counter with a 16-bit loadable register.

Project Description:

For this project to function properly, it requires momentary switches for reset and count and a slide switch to increment and decrement the count. The AISO circuit is used to generate a synchronous reset to all the modules and the debounce circuit is used to filter unwanted bounces in a mechanical switch. The positive edge detector is used to generate a one clock period wide pulse to the SR Flip Flop. The SR flop is used to store a 1-bit value when an interrupt is issued by the TramelBlaze processor. A 16 bit loadable register is used to store the current addresses from the processor which then outputs the current count to a seven segment display.

Before we begin in using the processor, we need to understand the architecture of the TramelBlaze. The TramelBlaze contains 16 16-bit registers, I/O ports and three memories: the instructions ROM, the call/return stack and the scratch pad RAM. The 16x4k instructions ROM is used to execute the code saved in ROM. The 512x16 ScratchPad RAM is available for intermediate values which are accessed by the fetch/store instructions. The 16x128 Call/Return

Stack is used to keep track of returned addresses for subroutines and interrupt handling. Below, there is an illustration of the architecture of the TramelBlaze.



The TramelBlaze processor also contains a single interrupt which is used to send a signal to the processor to indicate that an event needs immediate attention. When the interrupt is enabled and the pin transitions high, the current flow is suspended, and the processor jumps to the pre-defined interrupt address in assembly code. To be able fetch and execute instructions, we need to program the TramelBlaze using assembly language. We will be using a template to write our code according to the PicoBlaze syntax. We will be using arithmetic, test and compare, and control flow instructions to program the TramelBlaze processor. When developing our design, there are two approaches for utilizing the TramelBlaze in a design: simulation and hardware. When a digital design is destined to run on hardware the ROM and RAMs (tb_rom, call stack RAM, and scratchpad RAM) must be created. Due to timing consumption in creating the ROM and RAMs, we could use the TramelBlaze's simulation module to run our design immediately without having the build the memories every time.