Lab 8: Registers

October 26, 2009

In this lab we are going to investigate and build several sequential circuits. The behavior of the sequential systems depend not only on the current values of the input variables, but also on the sequence of input values that occurred in the past. Such systems have some kind of storage of memory elements. In this lab we are going to design couple types of registers.

Contents

1 Prelab

1.1 Registers

In this lab we will investigate two types of registers.

1.1.1 Simple Latch

One type of register is a simple latch as shown in Figure 1. When `LOAD` is high, the output data `D_OUT` will not change. When `LOAD` is low, the input data `D_IN` should be latched into the register on the rising edge of `CLOCK`.

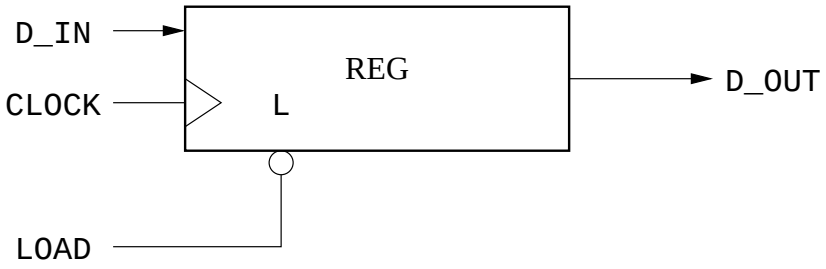

Figure 1: Register using simple latch

1.1.2 Program Counter

The second type of register is called a program counter (PC). This keeps track of which instruction in memory to execute. Usually programs are executed sequentially, so after executing the instruction at address, say, 0×0123, the program will then execute the instruction at address 0×0124. In this case PC needs to increment after each instruction is executed. Sometimes the program needs to execute code in a different area of memory - flow control statements such as `for` and `while` do this. In this case, the PC needs to be loaded with a new address. In order for the program to start, you will need to reset the program counter to zero to start execution at the first instruction of the program. Figure 2 shows what the PC looks like.

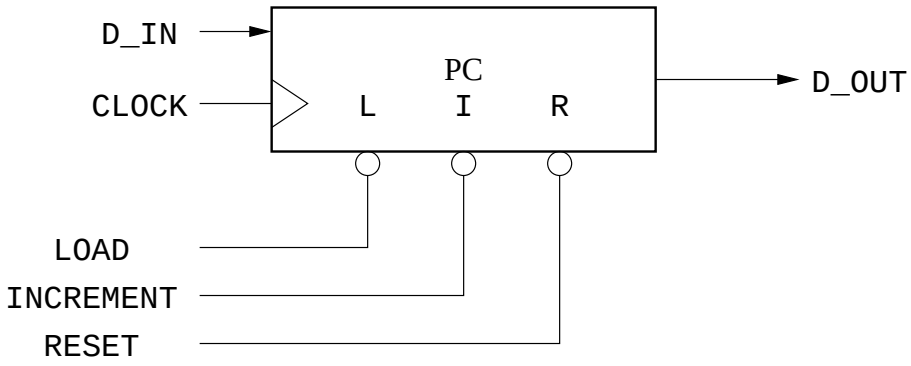

Figure 2: PC register

Normally, `INCREMENT`, `LOAD`, and `RESET` will be high. When `INCREMENT` is low, the PC should increment `D_OUT` to `D_OUT`+1 on the rising edge of `CLOCK`. When `LOAD` is low, the input data `D_IN` should be latched into the register on the rising edge of `CLOCK`. The system which controls PC will ensure that `LOAD` and `INCREMENT` are never low at the same time. (In your program, you should have PC do something sensible, like latch `D_IN`, if both happen to be low simultaneously.) When `RESET` is low, PC should immediately reset to 0×00; it shouldn't wait for a clock edge. This is normally called a synchronous counter with synchronous load and asynchronous reset.

1.2 Prelab Questions

1. Design an eight-bit synchronous latch in Verilog.

2. Design an eight bit PC as described above in Verilog.

3. Write a program which uses the above two designs as functions to test that they work.

2 Lab

In this part you will implement five different 8-bit registers: `PC` (program counter), `MAR` (Memory Addressing Register), `OUT` (Output), `ACCA` (Accumulator A), and `INST` (Instruction Register). In addition, you will design two single one-bit registers, `C` and `Z` (What simple circuit elements are `C` and `Z`?).

`MAR`, `OUT`, `ACCA`, and `INST`, are all 8-bit registers with synchronous parallel load. These registers all have a clock input, an 8-bit data input, and an active low load/enable input, as well as an 8-bit output.

For example, when `MAR_L` (Memory Addressing Register Load) is `VCC` the `MAR` register is not enabled. When `MAR_L` goes low the `MAR` register is enabled and on the next clock pulse the 8-bit data on the input line is loaded into the `MAR` register.

The `PC` is an 8-bit register with synchronous parallel load capability, synchronous count, and an asynchronous reset. The `PC` has a clock input, an 8-bit data input, and 3 additional inputs: `PC_L`, `PC_I`, and `RESET`. These three inputs are all active low. `PC_L` loads the program counter, `PC_I` increments the program counter by 1, and `RESET` resets the program counter to 0.

Implement these registers (synchronous load and synchronous load/count) as Altera functions. Include each one in a higher-level design file. Use a DIP switch for the input data, and switches on the evaluation board for `LOAD`, `INC`, and `CLOCK`. Verify that the load function works correctly for the parallel load register, and that the load and increment functions work correctly for the load/increment register.