Designing an Easily Modified LED Sequencer with No Processor Intervention

By Andrew Siska, Applications Engineer Senior Staff and Meng He, Product Marketing Engineer Senior, Cypress Semiconductor Corp.

As LED technology continues to be used in more and more applications, it is also becoming quite common for developers to seek ways to minimize cost and design complexity by implementing an LED or other device sequencer on a system-on-chip (SOC) platform. SOC devices integrate the MCU capacity and various digital peripherals required to support a complete LED subsystem using a single chip. This article describes a simple 8 LED sequencer design based on the latest SOC technology. However, the highlight in this design is that there is no interaction from the device's microprocessor. Instead of using the traditional passive digital peripherals with MCU processor intervention, this design is entirely based on the intelligently distributed processing functions in the SOC digital system. This frees the central CPU from managing the sequencer, eliminating any drain on horsepower and improving design efficiency.

This design approach can also be easily expanded to include timers, various lengths and patterns of sequence, and devices other than LEDs that are required to be powered up or down in a particular manner. Additional features were added to the design for demonstration purposes:

- terminal count out for a 7-bit counter (TC)
- direction output indicating whether the device is powering up or down
- 8-bit output for the sequenced devices
- clock input for the Verilog state machine
- bus-clock for one 8-bit ALU (bit-slice) processor.

The development tool used in this article is PSoC Creator, the development IDE for Programmable System On Chip (PSoC) from Cypress Semiconductor.

Schematic Design

The first step in developing the design is creating a Verilog symbol defining the inputs, outputs, and their associated bit widths (see Figure 1). Once the upper level Verilog module (schematic symbol) has been created, it is used to generate the Verilog source file containing all of the defined pins in the module. Functional Verilog code does not have to be developed at this step.

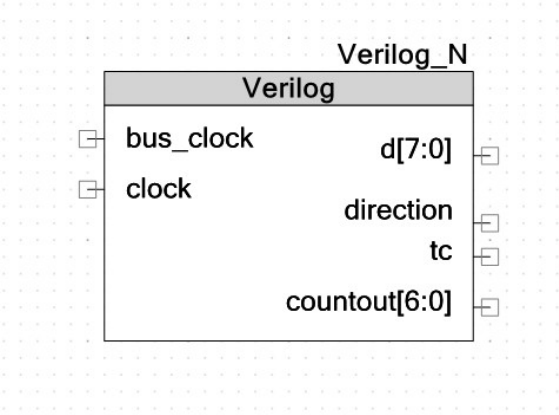

Figure 1: Verilog Symbol

Published in ECN Magazine (http://www.ecnmag.com/) November 2010

The Verilog symbol just created can now be placed onto a high-level schematic where each of the inputs and outputs can be connected to clock sources, I/O pins, status and control registers, etc. The high-level schematic in Figure 2 shows the mapping used for the 8-LED sequencer.

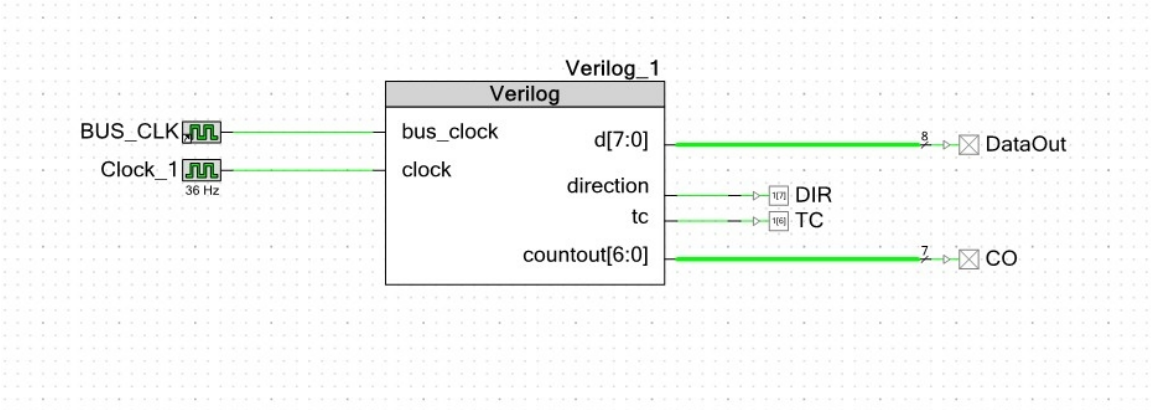

Figure 2: Example High-Level Schematic

At this point, the Verilog symbol has been created, placed within the high-level schematic, and connected to the device's I/Os and clocks. Verilog code can now be generated to perform the function – in this case, blinking the LEDs. To provide logic capabilities to manage sequencing, a simple data path can be introduced into the design.

This data path contains an 8-bit ALU with a reduced instruction set, two data registers, two accumulators, shift and compare logic, and a four deep 8-bit FIFO. To keep the design simple, only two ALU instructions were used: setting an accumulator to 0, and incrementing an accumulator each time a power-up or down sequence is performed. For more complex sequencing designs, developers can combine multiple ALUs to form a 16-bit or 24-bit processor. Such processors are similar to the bit-slice processors that were popular back in the 70's and early 80's and can provide sufficient processing capabilities for sequencing subsystems.

A picture of the Datapath Configuration tool appears below. Note the comments for the first two lines of the CFGRAM (Configuration RAM): "A0 <- 0", which clears accumulator 0, and "A0 <- A0 +1", which increments the value in A0.

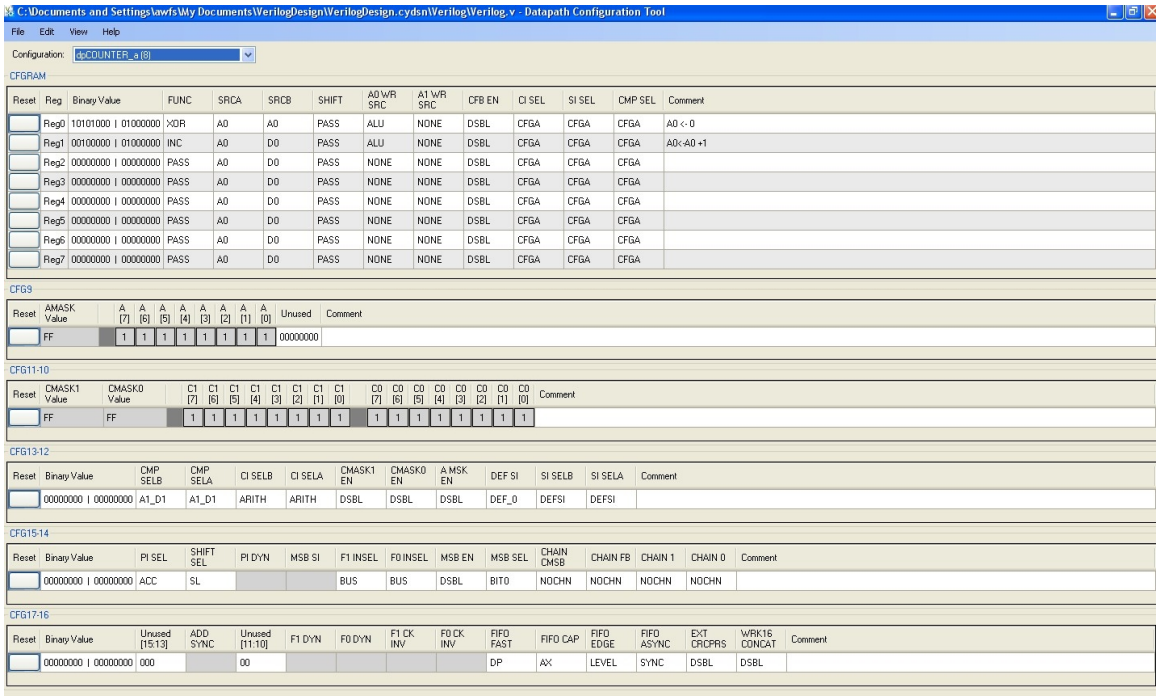

Figure 3: Data Path Configuration Tool

The system on chip (SOC) technology revives bit-slice in a programmable fashion to serve the purpose of offloading the main CPU by intelligently assigning processing tasks to other on-chip programmable hardware. With such an approach, it is possible to develop a standard state machine. The difference is that normally arithmetic functions consume a large number of logic gates. This is no longer a concern because these functions are implemented in the standard ALU contained in the data path logic and/or controlled by the PLD-based state machine.

This design runs independently of the main CPU. The primary application can control the sequencer through an API which modifies execution parameters and, once the sequencer is initiated, the CPU is no longer required. Additionally, this type of implementation is inherently efficient, utilizing fewer transistors than methods using the CPU, resulting in better overall system power consumption and more available overhead for other advanced features.

While this article discusses the design of an LED sequencer, the same design approach can be used for a variety of frequently-executed tasks to offload the main CPU by exploiting the full capabilities of an SOC's integrated architecture. In a world where engineers are constantly under pressure to increase performance, lower power consumption, and reduce system cost, having a another tool like this in their system design toolbox can help engineers continue to perform the miracles that have come to be expected of them.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone: 408-943-2600
Fax: 408-943-4730
http://www.cypress.com