

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.

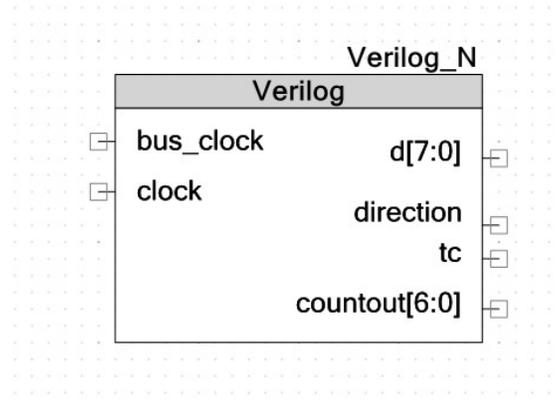


Figure 1: Verilog Symbol

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.

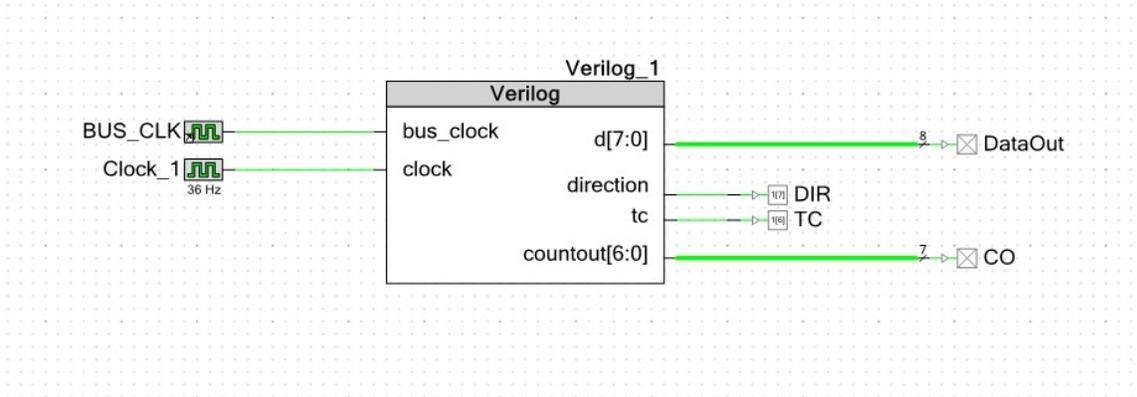


Figure 2: Example High-Level Schematic

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 CFGGRAM (Configuration RAM): "A0 <- 0", which clears accumulator 0, and "A0 <- A0 + 1", which increments the value in A0.

C:\Documents and Settings\lawf\My Documents\VerilogDesign\VerilogDesign.cysdn\Verilog\Verilog.v - Datapath Configuration Tool

File Edit View Help

Configuration: LCOUNTER_3 (8)

CFG0RAM

Reset	Reg	Binary Value	FUNC	SRCA	SRCB	SHIFT	A0 WR SRC	A1 WR SRC	CFB EN	CI SEL	SI SEL	CMP SEL	Comment
<input type="checkbox"/>	Reg0	10101000 01000000	>XDR	A0	A0	PASS	ALU	NONE	D5BL	CFGA	CFGA	CFGA	A0 < 0
<input type="checkbox"/>	Reg1	00100000 01000000	INC	A0	D0	PASS	ALU	NONE	D5BL	CFGA	CFGA	CFGA	A0 < A0 +1
<input type="checkbox"/>	Reg2	00000000 00000000	PASS	A0	D0	PASS	NONE	NONE	D5BL	CFGA	CFGA	CFGA	
<input type="checkbox"/>	Reg3	00000000 00000000	PASS	A0	D0	PASS	NONE	NONE	D5BL	CFGA	CFGA	CFGA	
<input type="checkbox"/>	Reg4	00000000 00000000	PASS	A0	D0	PASS	NONE	NONE	D5BL	CFGA	CFGA	CFGA	
<input type="checkbox"/>	Reg5	00000000 00000000	PASS	A0	D0	PASS	NONE	NONE	D5BL	CFGA	CFGA	CFGA	
<input type="checkbox"/>	Reg6	00000000 00000000	PASS	A0	D0	PASS	NONE	NONE	D5BL	CFGA	CFGA	CFGA	
<input type="checkbox"/>	Reg7	00000000 00000000	PASS	A0	D0	PASS	NONE	NONE	D5BL	CFGA	CFGA	CFGA	

CFG9

Reset	AMASK Value	A [7]	A [6]	A [5]	A [4]	A [3]	A [2]	A [1]	A [0]	Unused	Comment
<input type="checkbox"/>	FF	1	1	1	1	1	1	1	1	00000000	

CFG11-10

Reset	CMASK1 Value	CMASK0 Value	C1 [7]	C1 [6]	C1 [5]	C1 [4]	C1 [3]	C1 [2]	C1 [1]	C1 [0]	C0 [7]	C0 [6]	C0 [5]	C0 [4]	C0 [3]	C0 [2]	C0 [1]	C0 [0]	Comment
<input type="checkbox"/>	FF	FF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

CFG13-12

Reset	Binary Value	CMP SELB	CMP SELA	CI SELB	CI SELA	CMASK1 EN	CMASK0 EN	A MSK EN	DEF S1	SI SELB	SI SELA	Comment
<input type="checkbox"/>	00000000 00000000	A1_D1	A1_D1	ARITH	ARITH	D5BL	D5BL	D5BL	DEF_0	DEFS1	DEFS1	

CFG15-14

Reset	Binary Value	PI SEL	SHIFT SEL	PI DYN	MSB SI	F1 INSEL	F0 INSEL	MSB EN	MSB SEL	CHAIN CMSB	CHAIN FB	CHAIN 1	CHAIN 0	Comment
<input type="checkbox"/>	00000000 00000000	ACC	SL			BUS	BUS	D5BL	BIT0	NOCHN	NOCHN	NOCHN	NOCHN	

CFG17-16

Reset	Binary Value	Unused [15:3]	ADD SYNC	Unused [11:10]	F1 DYN	F0 DYN	F1 CK INV	F0 CK INV	FIFO FAST	FIFO CAP	FIFO EDGE	FIFO ASYNC	EXT ERCPRS	WRK18 CONCAT	Comment
<input type="checkbox"/>	00000000 00000000	000		00					DP	AK	LEVEL	SYNC	D5BL	D5BL	

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>

© Cypress Semiconductor Corporation, 2007. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC Designer™, Programmable System-on-Chip™, and PSoC Express™ are trademarks and PSoC® is a registered trademark of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

This Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and/or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.