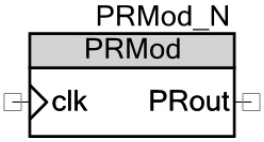
Pseudo Random Modulated Density Generator

PRMod v1.0

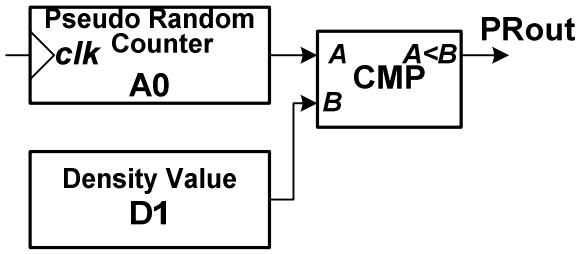


Features

- Synchronous operation
- Positive-edge triggered clock
- Produces a pseudo random modulated density stream
- Parameterized density value
- Simple to deconstruct

General Description

A pseudo random modulator is similar to a pulse width modulator except that a random number generator is used instead of a down counter. It has a frequency half way between a DSMod and PSMod. The output is HIGH whenever the counter is less than the density value. You can think about it as the output having a weighted probability of being HIGH. The randomness significantly reduces switching harmonics. This implementation of a CRC circuit produces a pseudo random series (0- 254) that repeats every 255 counts. The output is HIGH whenever the counter value (A0) is less than the density value (D1).



For a period of 255 cycles the output will be HIGH for of a total number of cycles equal to the **Density** parameter. However, the distribution of these within the 255 cycles will be (pseudo) random.

This component was built as a teaching tool. This classic component's operation is well understood and this datasheet's function is to help understand how the component was built through its deconstruction. It is recommended that you review the DSMod component before attempting this one.

Pin Description

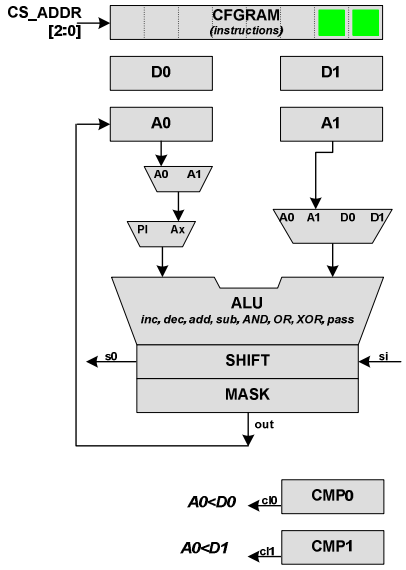
Pin	Type	Function
clk	input	clock input
PRout	output	pseudo random density stream

Function Table

Operating Modes	Input	Register		Output
	clk	MSbit	A0	PRout
count	↑	0	LeftCircularRotate(A0)	A0 < D1
count	↑	1	LeftCircularRotate(A0^0x38)	A0 < D1

Deconstructing the Component

This component's purpose is to show how to route a serial out (SO) and serial in (SI). Also this example shows how to control two different instructions from the output of a datapath comparator CMP0 and bring out an output from the other CMP1. For more information about datapaths, first review the DSMod. This component uses the following pieces of a datapath.



There are two instructions that are controlled by the state of the CMP0 Counter. If A0<128 then A0 is passed through the ALU, circularly left shifted and fed back to A0. If not then A0 is XORed with 0x38 before being circularly left shifted and fed back to the ALU. These operation form a pseudo random number generator with a range of 0 - 254.

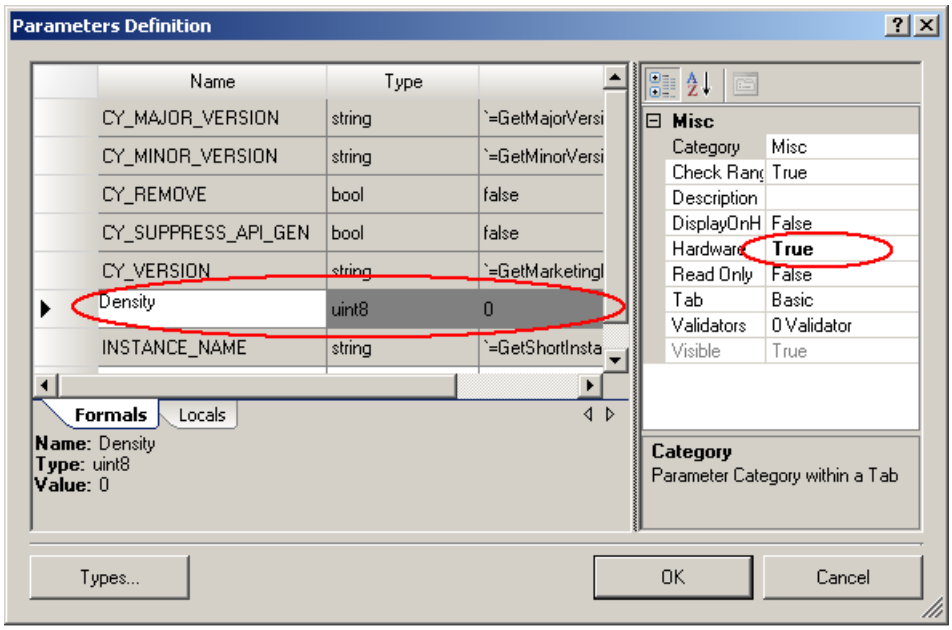
Using PSoC Creator, open the PRMod example project to see the project schematic (*TopDesign.cysch*). It has a PRMod component connected to input switches, output LEDs, and a clock.

In the Workspace Explorer, click the **Components** tab. Then, right-click on the project and select **Import Component**. Navigate to where the CYCC_SimpleComps library project is, and select the CYCC_PRMod_v1_0 component. Click **OK** and the following files are shown for the component:

- Symbol file (cysym)
- Datasheet (pdf)
- Verilog File (v)

Open the symbol file to find a symbol with one input and one output. It looks like the symbol shown on the first page.

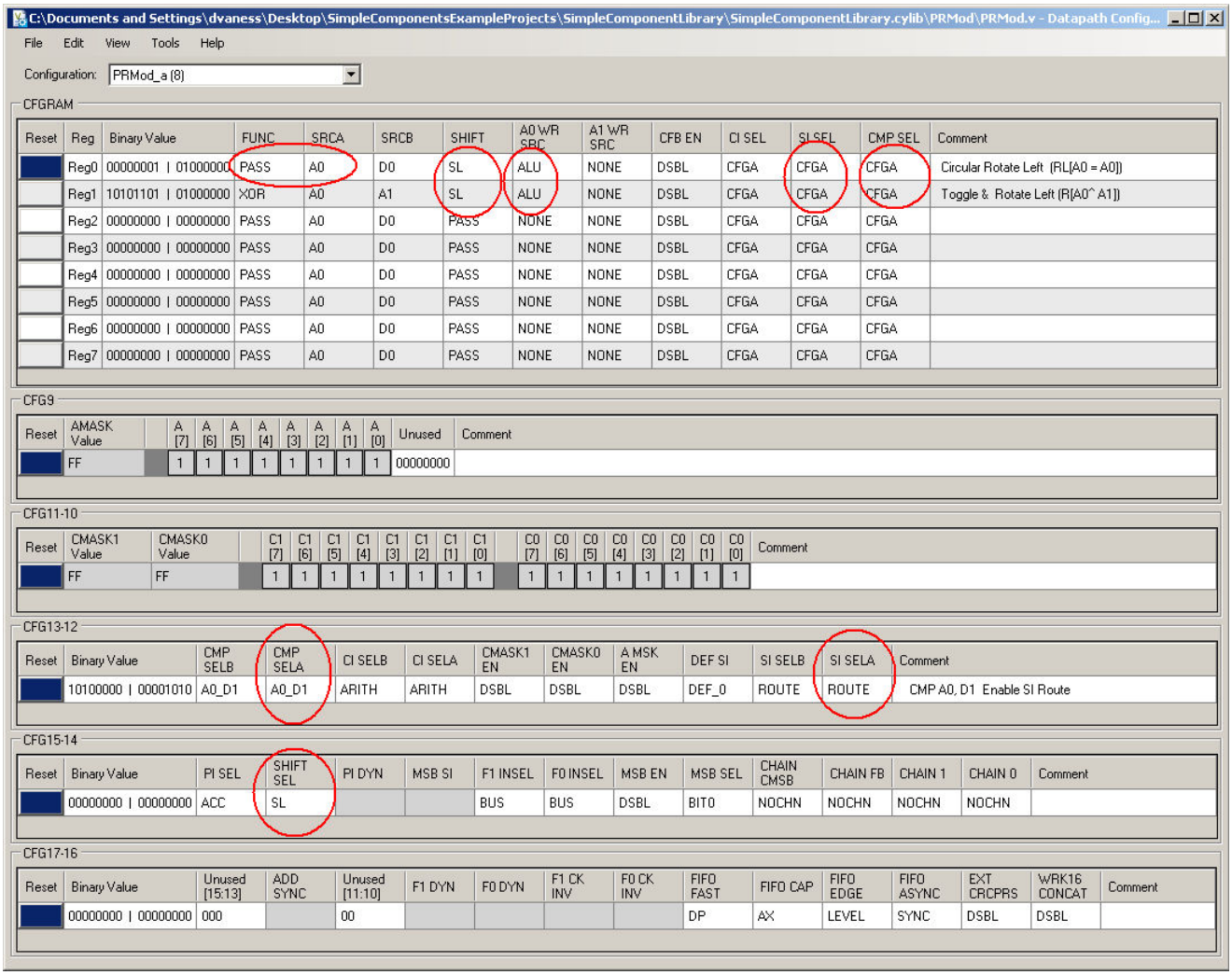
While viewing the symbol file, right-click on the canvas and select **Symbol Parameters**.



Note that **Density** is included as a parameter. It is an unsigned 8-bit value that has a default value of zero. Its hardware flag has been enabled.

Open the Verilog file and notice that at lines 24-25, these definitions were passed from the symbol to this Verilog file when it was created. Lines 32-33 are some intermediate signals. Line 28 is where the symbol Density parameter got passed. The first 18 lines of this header list register usage and the datapath instruction definitions.

What follows is the datapath module definition. It was created and inserted by the Datapath Configuration Tool. This information is backward compatible so opening up the Datapath Configuration Tool for this Verilog file results in the following.



C:\Documents and Settings\dvaness\Desktop\SimpleComponentsExampleProjects\SimpleComponentLibrary\SimpleComponentLibrary.cylib\PRMod\PRMod.v - Datapath Config...

File Edit View Tools Help

Configuration: PRMod_a (8)

CFGRAM

Reset	Reg	Binary Value	FUNC	SRCA	SRCB	SHIFT	A0 WR SRC	A1 WR SRC	CFB EN	CI SEL	SI SEL	CMP SEL	Comment
	Reg0	00000001 | 01000000	PASS	A0	D0	SL	ALU	NONE	DSBL	CFGA	CFGA	CFGA	Circular Rotate Left (RL[A0 = A0])
	Reg1	10101101 | 01000000	XOR	A0	A1	SL	ALU	NONE	DSBL	CFGA	CFGA	CFGA	Toggle & Rotate Left (R[A0^ A1])
	Reg2	00000000 | 00000000	PASS	A0	D0	PASS	NONE	NONE	DSBL	CFGA	CFGA	CFGA	
	Reg3	00000000 | 00000000	PASS	A0	D0	PASS	NONE	NONE	DSBL	CFGA	CFGA	CFGA	
	Reg4	00000000 | 00000000	PASS	A0	D0	PASS	NONE	NONE	DSBL	CFGA	CFGA	CFGA	
	Reg5	00000000 | 00000000	PASS	A0	D0	PASS	NONE	NONE	DSBL	CFGA	CFGA	CFGA	
	Reg6	00000000 | 00000000	PASS	A0	D0	PASS	NONE	NONE	DSBL	CFGA	CFGA	CFGA	
	Reg7	00000000 | 00000000	PASS	A0	D0	PASS	NONE	NONE	DSBL	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
	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
	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 SI	SI SELB	SI SELA	Comment
	10100000 | 00001010	A0_D1	A0_D1	ARITH	ARITH	DSBL	DSBL	DSBL	DEF_0	ROUTE	ROUTE	CMP A0, D1 Enable SI Route

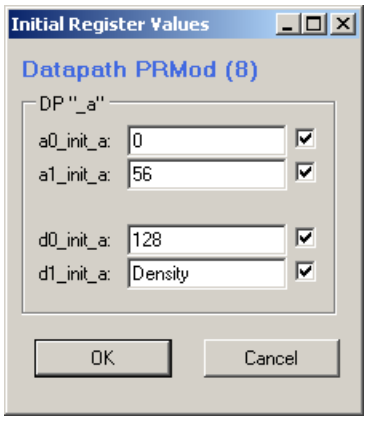
CFG15-14

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
	00000000 | 00000000	ACC	SL			BUS	BUS	DSBL	BIT0	NOCHN	NOCHN	NOCHN	NOCHN	

CFG17-16

Reset	Binary Value	Unused [15:13]	ADD SYNC	Unused [11:10]	F1 DYN	F0 DYN	F1 CK INV	F0 CK INV	FIFO FAST	FIFO CAP	FIFO EDGE	FIFO ASYNC	EXT CRCPRS	WRK16 CONCAT	Comment
	00000000 | 00000000	000		00					DP	AX	LEVEL	SYNC	DSBL	DSBL	

Note that SHIFT_SEL is set for left shift operations and CMP_SELA is set to allow a serial input to be routed into the shifter. One instruction XORs A0 with 0x38 (A1) while the other just passes A0 through. CMP_SELA is set so A0 and D1 are compared. The four registers have been initialized as shown below.



When saved, this tool inserts the updated configuration data back into the Verilog file along with an instance of the datapath interface. Just pass the correct signals in to and out of it and you are done. This counter parameter list may look ominous but only a few parameters have to be entered as shown below.

```
/* input          */ .clk(clk),                  // Clock input
/* input  [02:00] */ .cs_addr({2'b00, MSBisLow}), // Logic to control instructions
/* input          */ .route_si(MSbit),           // Routed from SO
/* output         */ .cl0(MSBisLow),             // Get CMP0 flag
/* output         */ .cl1(PRout),                // Weighted density output
/* output         */ .so(MSbit),                 // Routed to SI
```

These parameters do the following:

- Connect a clock signal to the datapath clock circuitry
- Route the so to the si
- Two datapath instructions are controlled by a comparator flag
- Route the so to the si for a circular left shift
- Bring the CMP1 flag to an output

Support

PSoC Creator Community Components are developed and supported by the Cypress Developer Community. Go to www.cypress.com/CommunityComponents to discuss this and other Community Components.

© Cypress Semiconductor Corporation, 2013. 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® is a registered trademark, and PSoC Creator™ and Programmable System-on-Chip™ are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

This work is licensed under a Creative Commons Attribution 3.0 Unported License.

http://creativecommons.org/licenses/by/3.0/deed.en_US

You are free to:

- Share — to copy, distribute and transmit the work
- Remix — to adapt the work
- Make commercial use of the work