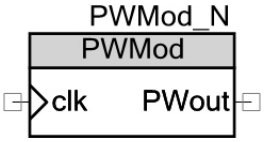
Pulse Width Modulated Density Generator

PWMod v1.0

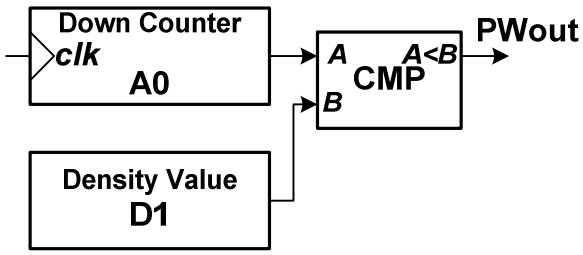


Features

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

General Description

A pulse width modulator produces the lowest possible output frequency for a given density. This implementation is a down counter and an 8-bit wide digital comparator. The output is HIGH whenever the counter value (A0) is less than the density value (D1).



For a period of 256 cycles the output will be HIGH for a continuous number of cycles equal to the density value.

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

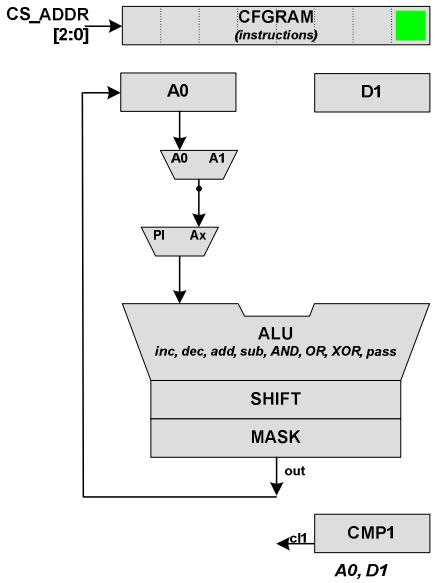
Pin	Type	Function
clk	input	clock input
DSout	output	pulse width density stream

Function Table

Operating Modes	Input	Register	Output
	clk	A0	PWout
count	↑	A0-1	A0 < D1

Deconstructing the Component

This component's purpose is to show how to connect an output from one of the datapath compare registers. For more information about datapaths, first review the DSMod component. This component uses the following pieces of a datapath.



The single operation to be performed is that on each clock cycle, A0 is fed to the ALU were it is decremented and fed back to A0. CMP1 is used to compare A0 and D0. The comparator's output is HIGH whenever A0 is less than D0. This flag (cl1) is brought out.

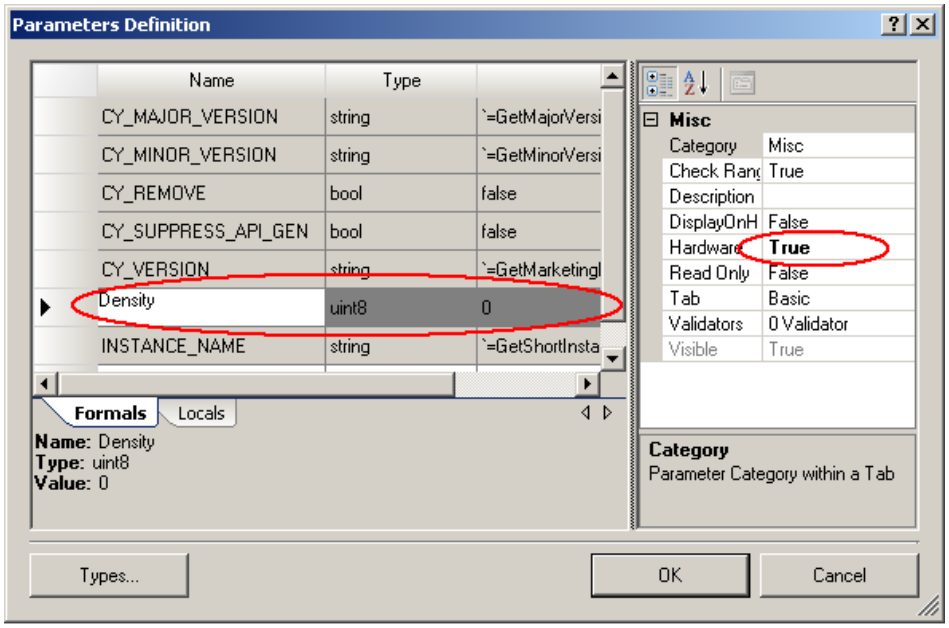
Using PSoC Creator, open the PWMod example project to see the project schematic (*TopDesign.cysch*). It has a PWMod component connected to an output LED and a clock.

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_PWMod_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 and you will find a symbol with one input and one output. It looks like the symbol shown on the first page.

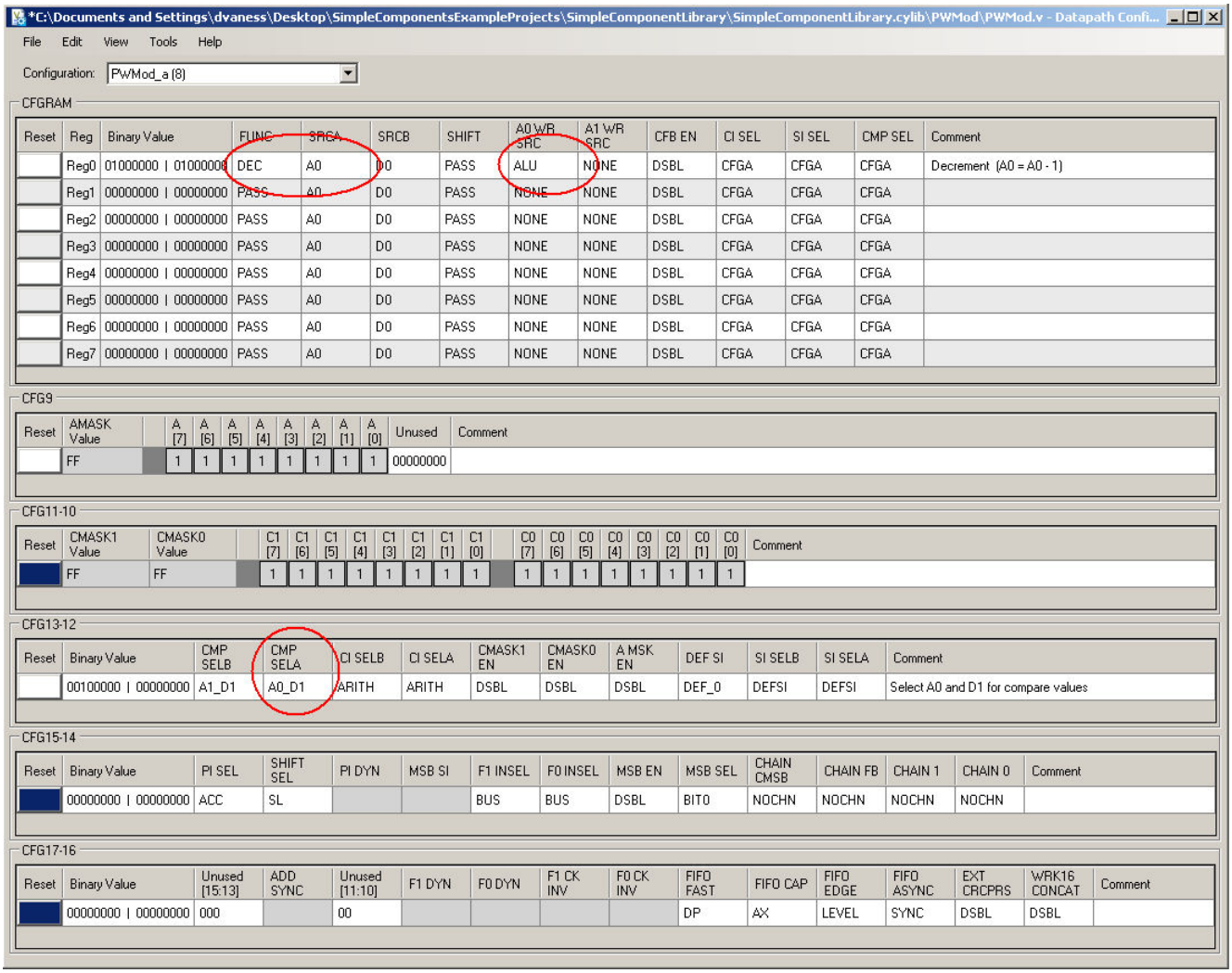
While viewing the symbol file, right-click on the canvas and select **Symbol Parameters**.



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 22-23, these definitions were passed from the symbol to this Verilog file when it was created. Line 25 is where the symbol Density parameter got passed. The first 16 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\PWMod\PWMod.v - Datapath Confi...

File Edit View Tools Help

Configuration: PWMod_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	01000000 | 01000000	DEC	A0	D0	PASS	ALU	NONE	DSBL	CFGA	CFGA	CFGA	Decrement (A0 = A0 - 1)
	Reg1	00000000 | 00000000	PASS	A0	D0	PASS	NONE	NONE	DSBL	CFGA	CFGA	CFGA	
	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
	00100000 | 00000000	A1_D1	A0_D1	ARITH	ARITH	DSBL	DSBL	DSBL	DEF_0	DEFSI	DEFSI	Select A0 and D1 for compare values

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	

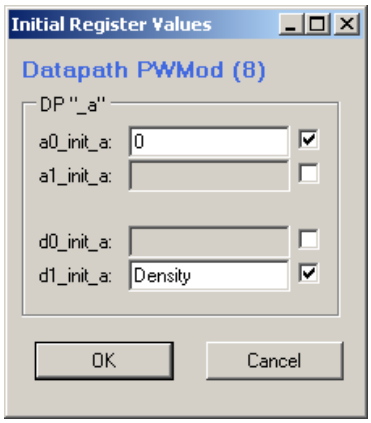
CFG17-16

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 only one operation is needed to be implemented and that CMP_SELA is set so A0 and D1 are compared.

For the dec instruction A0 is decremented and the result is fed back to A0.

These two 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 into 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),           // Connect a clock to the datapath
    /* input [02:00]   */  .cs_addr(3'b00),     // Only Reg0 instruction is used
    /* output          */  .cl1(PWout),         // Get compare flag from datapath
```

These parameters do the following:

- Connect a clock signal to the datapath clock circuitry
- Only a single datapath instruction (REG0) is used
- 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