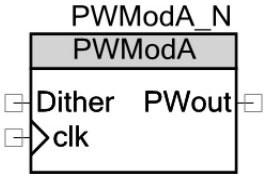
Pulse Width Modulated Density Generator with Dither

PWModA v1.0

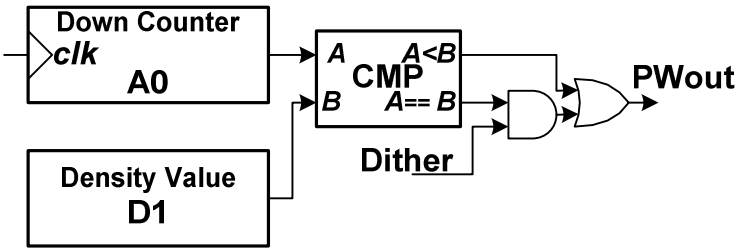


Features

- Synchronous operation
- Positive-edge triggered clock
- Produces a delta sigma modulated density stream.
- Dither to allow fractional density values
- 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 of a continuous number of cycles equal to the density value. If the Dither is HIGH then the number of continuous HIGH cycles increases by one. If the Dither input is kept HIGH one part in n, the average density value is.

$$\overline{Density} = Density + \frac{1}{n}$$

This means that fraction counts of resolution can be achieved.

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 it was built through its deconstruction. It is recommended that you review the DSMod component before attempting this one.

Pin Description

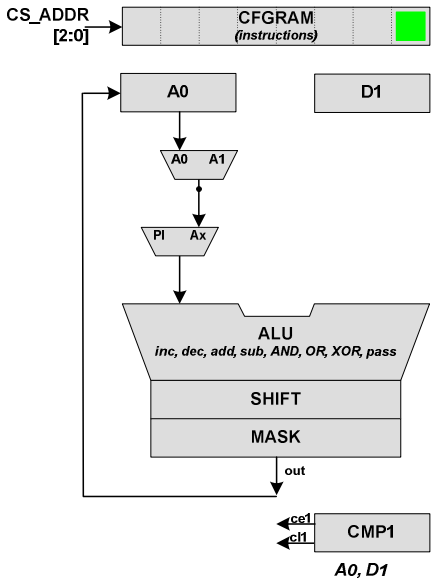
Pin	Type	Function
Dither	input	dither input
clk	input	clock input
PWout	output	pulse width density stream

Function Table

Operating Modes	Inputs		Register	Output
	clk	Dither	A0	PWout
count	↑	0	A0 - 1	A0 < D1
count	↑	1	A0 - 1	A0 <= D1

Deconstructing the Component

This component's purpose is to show how to connect an input to the ALU in a datapath. For more information about datapaths, 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. The flags (cl1 and ce1) are brought out.

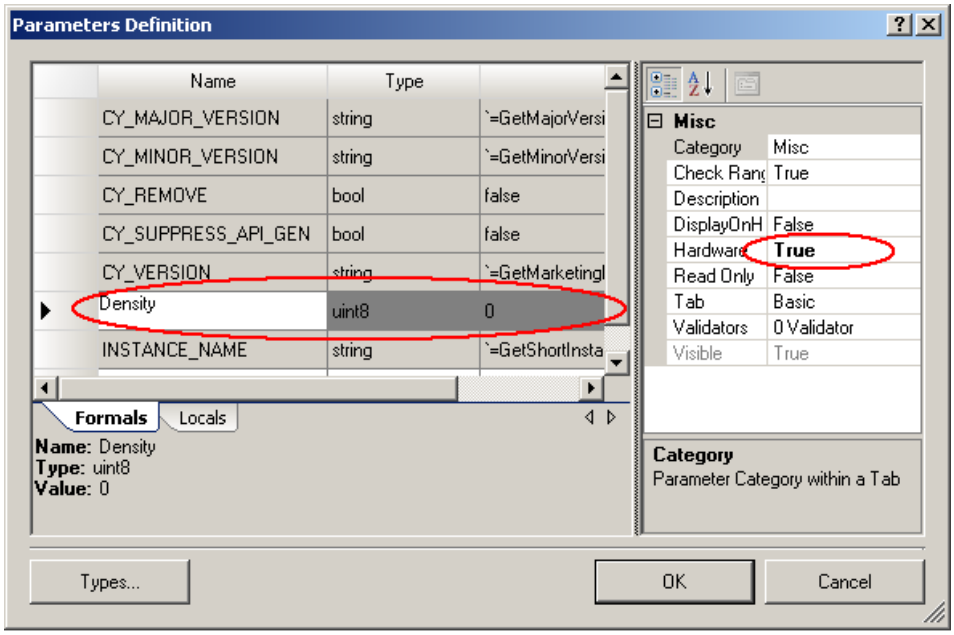
Using PSoC Creator, open the PWModA example project to see the project schematic (*TopDesign.cysch*). It has a PWModA component connected to input switches, output LEDs, and a clock.

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_PWModA_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 two inputs and one output. It looks like the symbol shown on the first page.

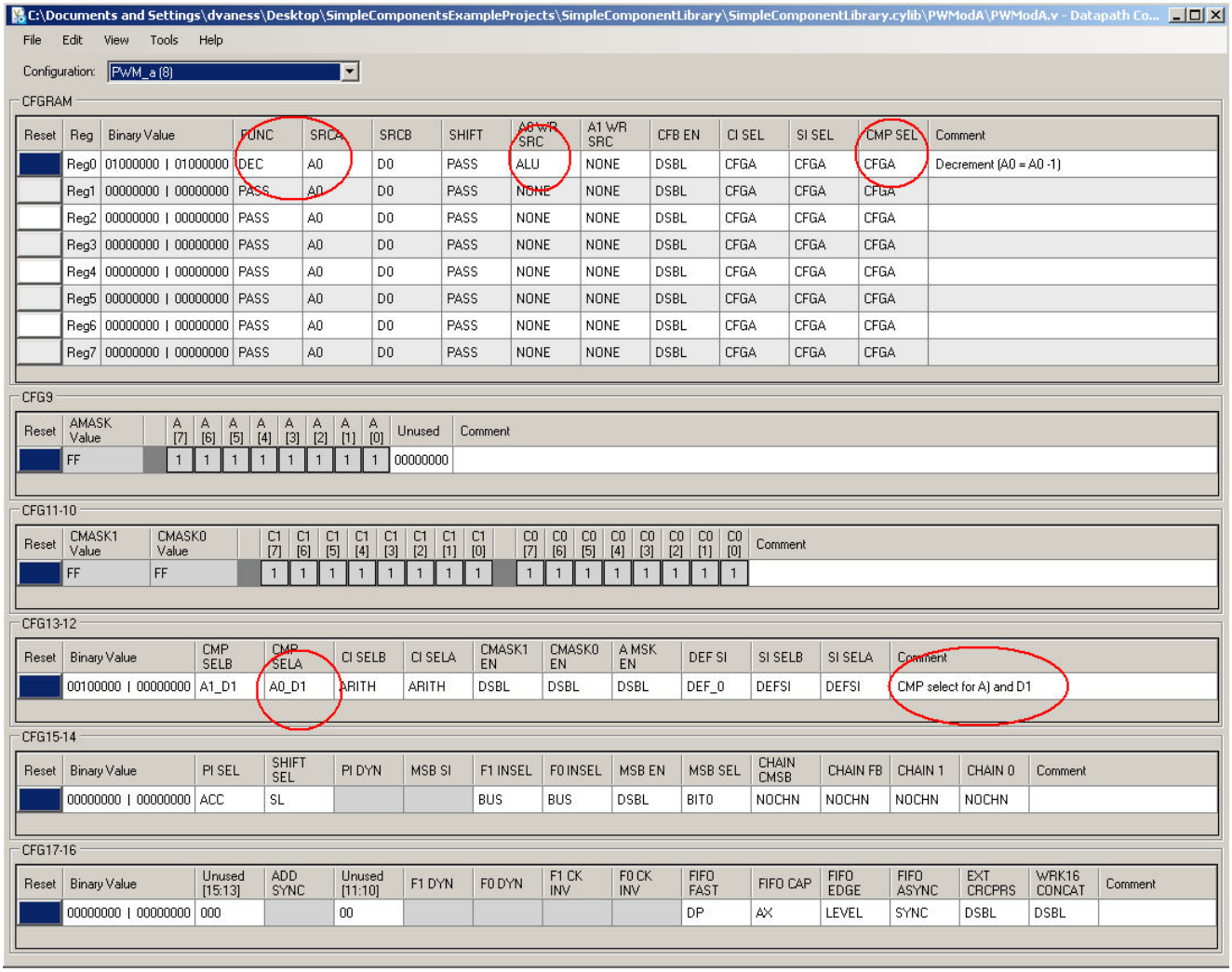
While viewing the symbol file, right-click on the canvas and select **Symbol Parameters**.



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 24-26, these definitions were passed from the symbol to this Verilog file when it was created. Lines 32-34 are the combinational logic needed to couple the two compare flags with the Dither. 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.

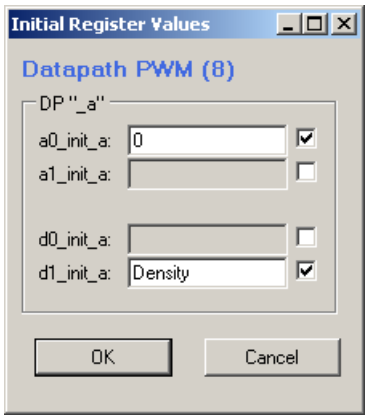
What follows is the datapath module definition. It was created and inserted by the Datapath Configuration Tool. This information is backward compatible so opening up the datapath configuration tool for this Verilog file results in the following.



Note that only one operation is needed to be implemented and that CI SELA is set to allow ci to be routed to the ALU.

For the add instruction A0, D1, ci are added 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),          // Clock for datapath
/*  input   [02:00]         */  .cs_addr(3'b0),     // Only Reg0 is used
/*  output                  */  .ce1(EqualTo),      // Bring out ce1
/*  output                  */  .cl1(LessThan),     // Bring out cl1
/*  output                  */  .z1(),
```

These parameters do the following:

- Connect a clock signal to the datapath clock circuitry
- Only a single datapath instruction (REG0) is used
- Bring out the LessThan flag from the datapath
- Bring out the EqualTo flag from the datapath DSout

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