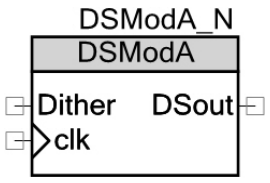
Delta Sigma Modulated Density Generator with Dither

DSModA v1.0

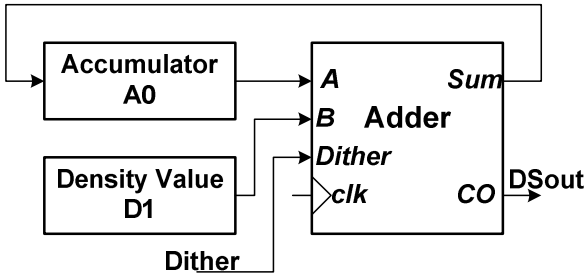


Features

- 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 delta sigma modulator produces the highest possible output frequency for a given density. This implementation is the carry output from an accumulated density value.



The adder and registers are all 8-bits wide. When Dither is LOW and the adder is operated 256 times, the number of carries equals the density value. When Dither is HIGH, for the same number of cycles, the number of carries is now one greater. 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 the component was built through its deconstruction. It is recommended that you review the DSMod component before attempting this one.

Pin Description

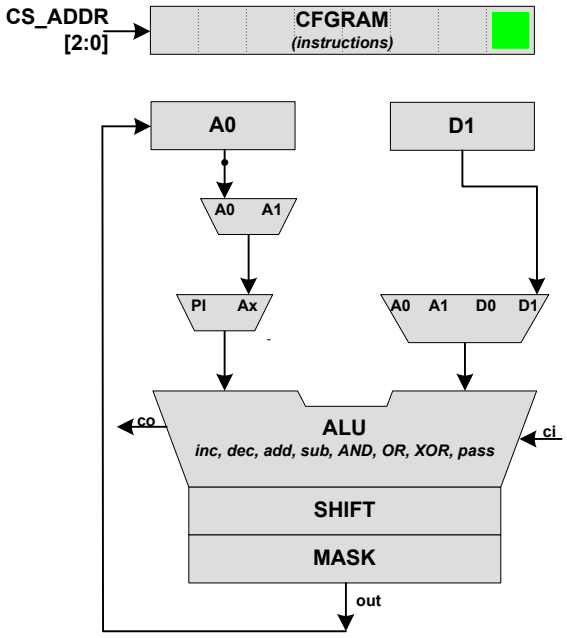
Pin	Type	Function
Dither	input	dither input
clk	input	clock input
DSout	output	delta sigma density stream

Function Table

Operating Modes	Input	Register	Output
	clk	A0	DSout
count	↑	mod(A0 + D1 + Dither, 256)	256<= A0 + D1 + Dither

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, refer to 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, D1, and ci are fed to the ALU where they are added and fed back to A0. The carry (co) flag is brought out.

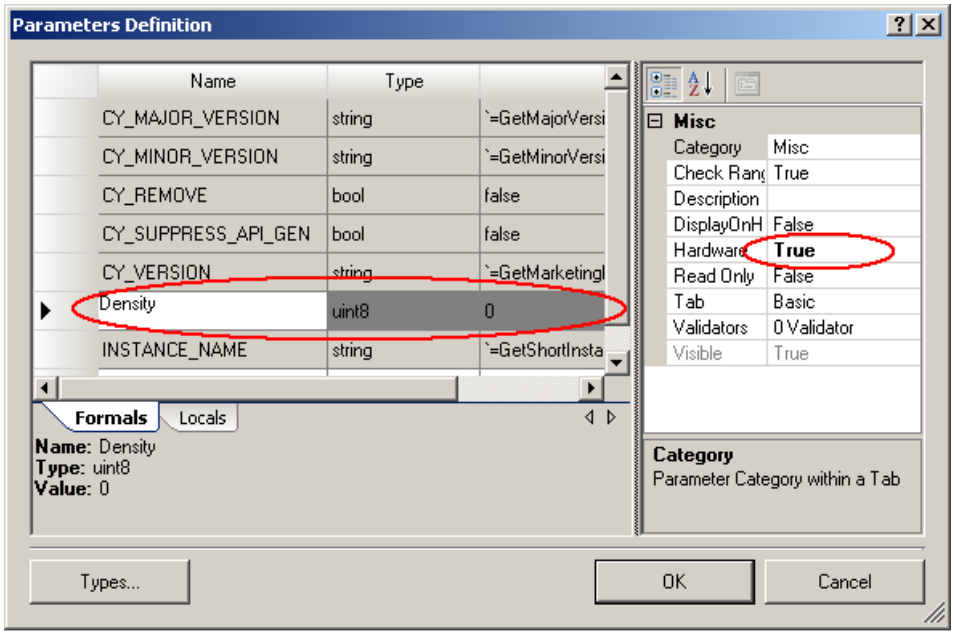
Using PSoC Creator, open the DSModA example project to see the project schematic (*TopDesign.cysch*). It has a DSModA component connected to input switches, output LEDs a clock.

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_DSModA_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.

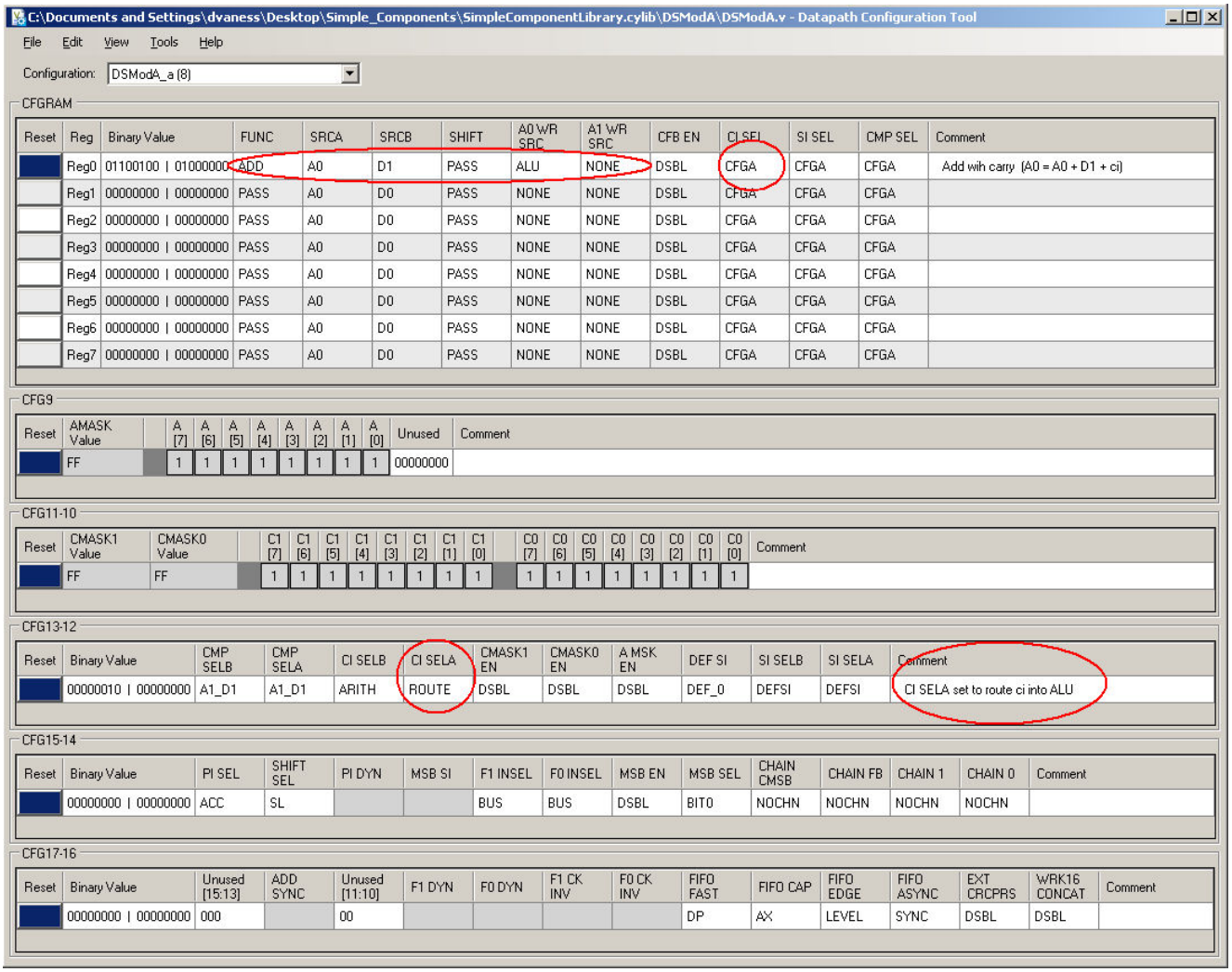
While viewing the symbol file, right-click on the canvas and select **Symbol Parameters**.



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 22-25, these definitions were passed from the symbol to this Verilog file when it was created. Line 27 is where the symbol Density parameter got passed. The first 18 lines of this header list register usage and the datapath instruction definitions.

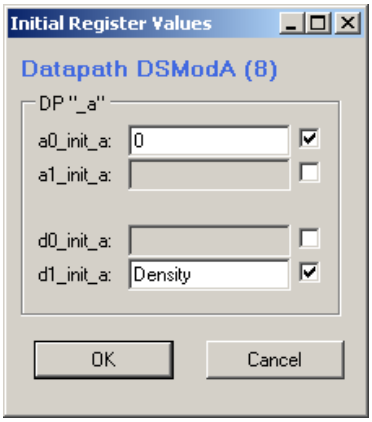
What follows is the datapath module definition. It was created and inserted by the Datapath Configuration Tool. This information is backward compatible so opening up the Datapath Configuration Tool for this Verilog file results in the following.



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 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),          // Connet a clock to the datapath
/* input [02:00] */ .cs_addr(3'b0),     // Only a single instruction (REG0) is used
/* input          */ .route_ci(Dither), // Carry input routed to ALU
/* output         */ .co_msb(DSout),    // Carry Output routed from ALU
```

These parameters do the following:

- Connect a clock signal to the datapath clock circuitry
- Only a single datapath instruction (REG0) is used
- Bring the Dither to the ci on the datapaths ALU
- Bring out the carry 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