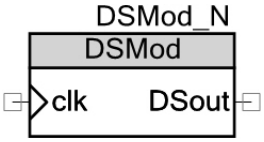
Delta Sigma Modulated Density Generator

DSMod v1.0

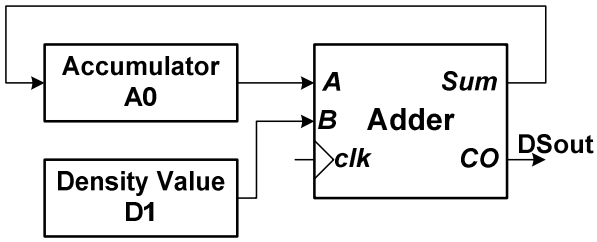


Features

- Synchronous operation
- Positive-edge triggered clock
- Produces a delta sigma modulated density stream.
- Parameterized density value.
- Simple to deconstruct

General Description

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



The adder and registers are all 8-bits wide. If the density value is set to 128, then a carry output will be generated 50% (128/256) of the time and have an average value of 50%. It will be a stream of repeating of ones and zeros, and have an output frequency of clk·50%.

If the density value is set to 64, then a carry will be generated every fourth (64/256) clock, have an average value of 25%, and the output frequency will be clk·25%.

If the density is set to 85, then over 256 cycles there will be a periodic stream of:

- 84 repetitions of 0,0,1
- 1 repetition of 0,0,1

The result is 85 of every 256 cycles are high for an average value of 33.33% and the corresponding average output frequency.

This component was built as a teaching tool. The classic component's operation is well understood and this datasheet's function is to help understand how the component was built through its deconstruction.

Pin Description

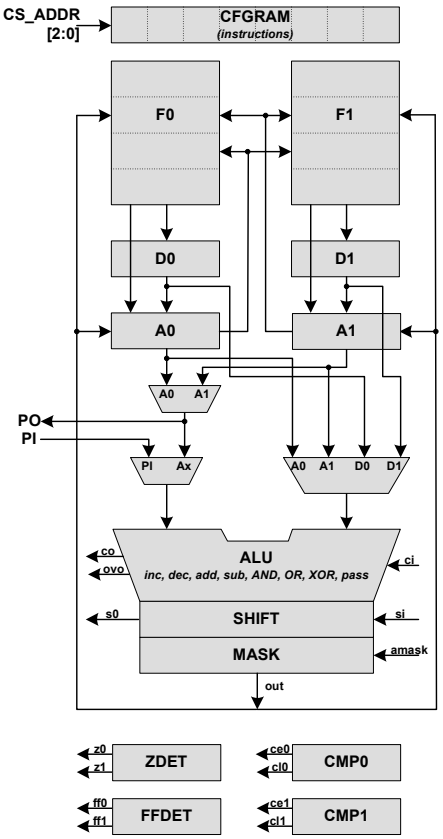
Pin	Type	Function
clk	input	clock input
DSout	output	Delta sigma density stream

Function Table

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

Datapath Primer

In digital design there is a conflict between the versatility of programmable logic and the compactness of a fixed design peripheral. Cypress' solution was to design a sequential programmable logic device. We call it a "datapath" and a simplified block diagram for single block use is shown below.

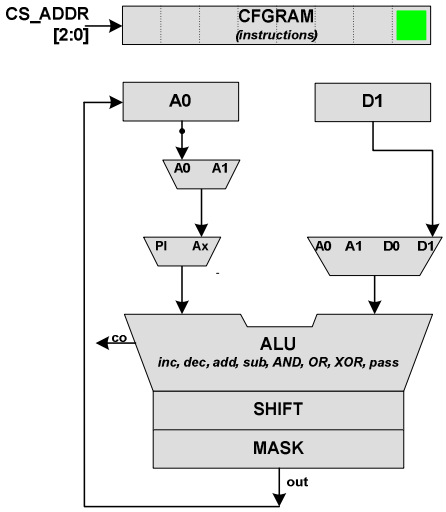


It comes with an ALU, shift register, and four 8-bit digital compare registers. There are 8 different data registers and the ability to bring parallel data into and out of this block. The flow is that data from some specific registers is fed to the ALU/SHIFT where it is manipulated and fed back to the appropriate registers. The MASK is used to allow data widths less than 8-bits (6-bit UART as an example). This process is controlled by one of the eight instructions.

These instructions are configurable. Three inputs control which instruction is being processed. Couple this with the programmable logic and you have you own little pico-processor. The datapath configuration tool is used to build up to eight different routings (instructions). By reviewing this series of simple components, you will better understand the datapath's operation and be able to design your own datapath-based components.

Deconstructing the Component

This component's purpose is to show how to set up a simple datapath instruction and bring an output flag from the ALU. It uses the following pieces of a datapath.



The single operation to be performed is that on each clock cycle, A0 and D1 are fed to the ALU were they are added and fed back to A0. The carry (co) flag is brought out.

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

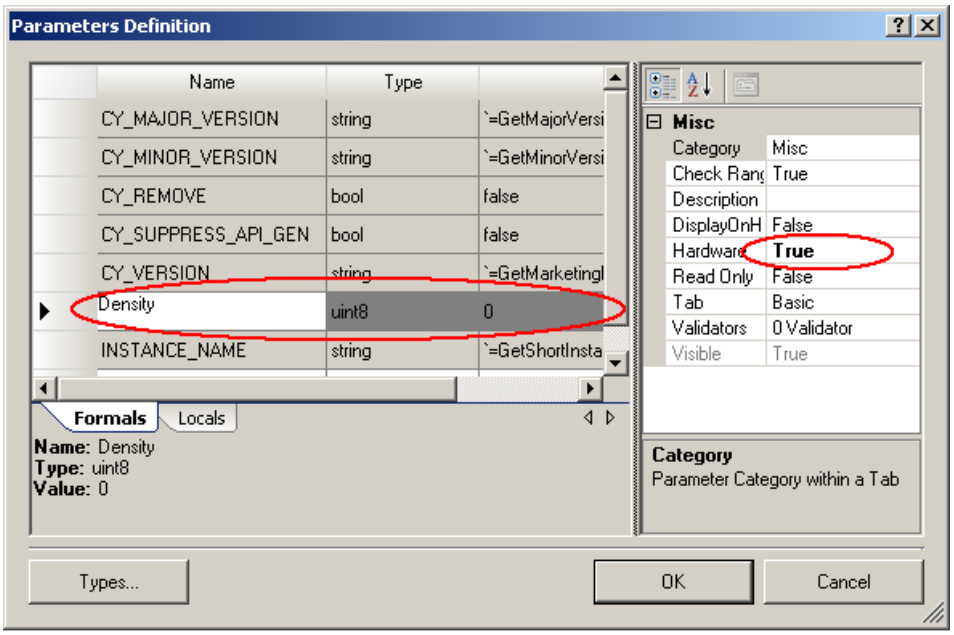
In the Workspace Explorer, click the **Components** tab. Then, right-click on the project and select **Import Component**. Navigate to where the CYCC_SimpleComps library project is, and select the CYCC_DSMod_v1_0 component. Click **OK** and the following files are shown for the component:

- 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.

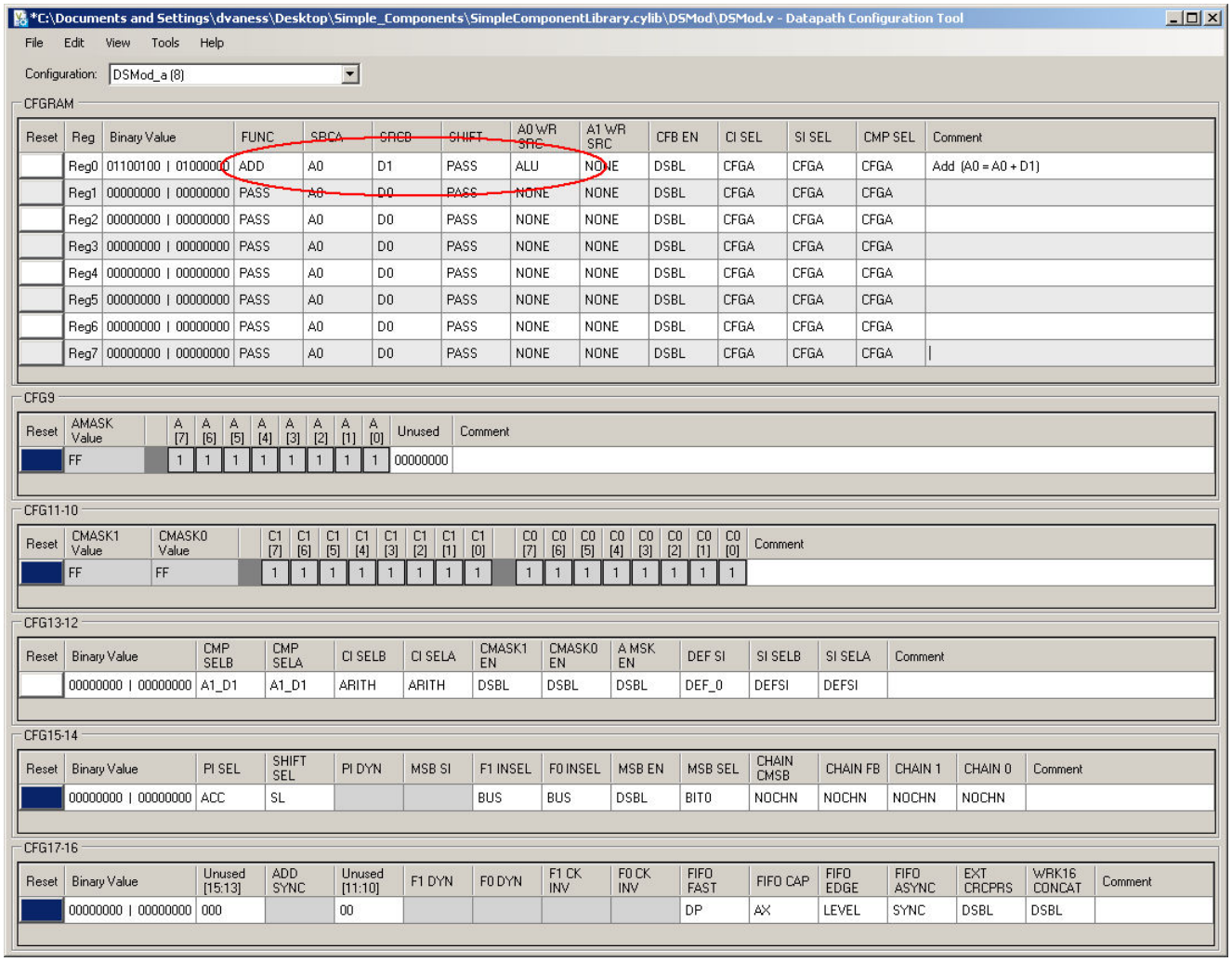
While viewing the symbol file, right-click on the canvas and select **Symbol Parameters**.



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 25-26, these definitions were passed from the symbol to this Verilog file when it was created. Line 28 is where the symbol Density parameter got passed. The first 19 lines of this header list register usage and the datapath instruction definitions.

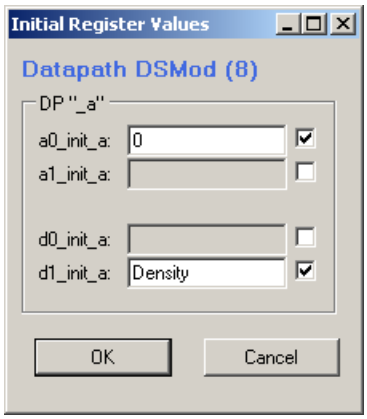
What follows is the datapath module definition. It was created and inserted by the Datapath Configuration Tool. This information is backward compatible so opening up the Datapath Configuration Tool for this Verilog file results in the following.



Note that only one operation is needed to be implemented.

For the add instruction A0 and D1 are added the result is fed back to A0.

These two registers have been initialized as shown.



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),           // Connect clock to datapath
/*  input [02:00]   */  .cs_addr(3'b0),      // Only first instruction (REG))
/*  output          */  .co_msb(DSout),      // Bring the co to the output
```

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 carry from the datapath

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