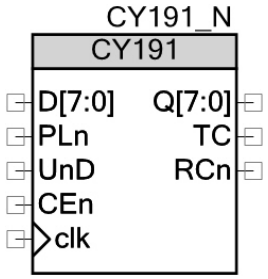
8-Bit Binary Up/Down Counter

CY191 v1.0



Features

- Synchronous reversible counting
- Synchronous parallel load
- Positive-edge triggered clock
- Simple up/down control input
- Simple to deconstruct

General Description

This counter is based on the 74HC191. It is a synchronous, pre-settable 8-bit binary counter with look-ahead carry. Synchronous operation is implemented with positive-edge triggered logic. The counter, Q[7:0] (datapath register A0), may be preset to the load value at inputs D[7:0], using the parallel load (PLn) signal. The count enable input (CEn) enables counting. The up/down input (UnD) determines the direction of the counting as indicated in the function table. It also controls the terminal count (TC) output so it is high when the counter is at the top while counting up or at the bottom when counting down.

The TC output will stay high until a state change occurs, either by counting, presetting, or until count direction (UnD) changes. The TC signal is used to generate the ripple clock (RCn) output. RCn is active for the last half the clock when both TC and CEn are active. The RCn can be used as a clock for the next higher stage of counter.

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.

Pin Description

Pin	Type	Function
D[7:0]	inputs	value to be loaded
PLn	input	parallel load enable input (active low)
UnD	input	up/down direction input
CEn	input	count enable input (Active low)

Pin	Type	Function
clk	input	clock input
Q[7:0]	outputs	counter value outputs
TC	output	terminal count output
RCn	output	ripple clock output (active low)

Function Table

Operating Modes	Inputs				Outputs		
	PLn	UnD	CEn	clk	Q	TC	RCn
parallel load	0	x	x	↑	D	UnD&0f00 | ~UnD&0xff	~(TC &~CEn &~clk)
count up	1	0	0	↑	Q+1	UnD&0f00 | ~UnD&0xff	~(TC &~CEn &~clk)
count down	1	1	0	↑	Q-1	UnD&0f00 | ~UnD&0xff	~(TC &~CEn &~clk)
hold	1	x	1	↑	Q	UnD&0f00 | ~UnD&0xff	~(TC &~CEn &~clk)

Deconstructing the Component

This component has connections to the Parallel Out (PO) and Parallel In (PI) interface. If you do not understand how they operate then please first review the CY161 component. This component has three synchronous inputs, so it easily fits into the eight available datapath configurations (instructions). With four different operations, it makes a good example on how to configure the datapath instructions.

For this counter there are four different operations.

- Load
- Count Up
- Count Down
- Hold (do nothing)

These four operations need to be implemented with datapath instructions.

Using PSoC Creator, open the CY191 example project and you will see the project schematic (*TopDesign.cysch*). It has a CY191 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

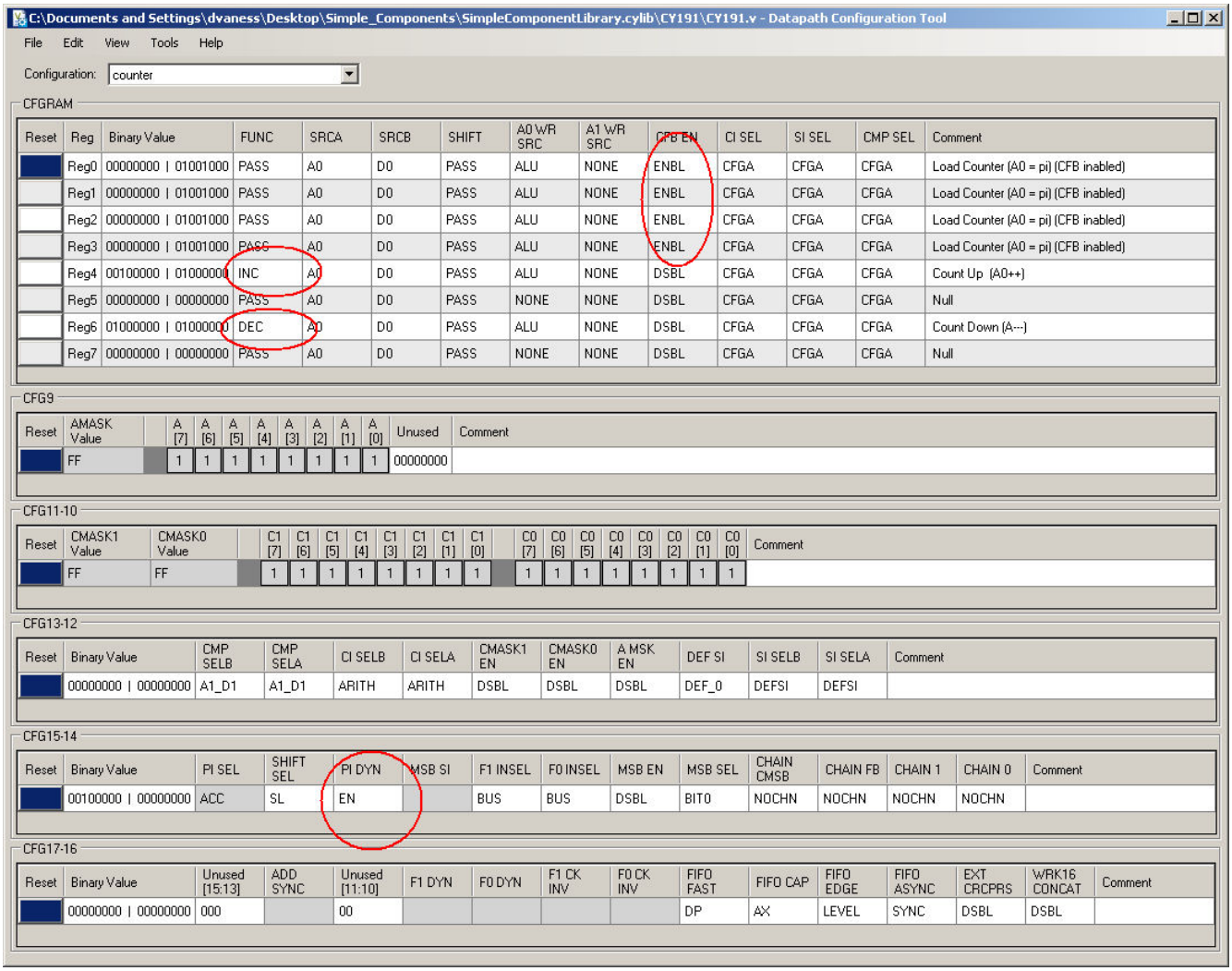
the CYCC_CY191_v1_0 component. Click **OK** and the following files are shown for the component:

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

Open the symbol file to find a symbol with five inputs and two outputs. It looks like the symbol shown on the first page. There are no additional symbol parameters.

Open Verilog file and notice that at lines 32 – 39, these definitions were passed to this Verilog file when it was created. The first 26 lines of this header list register usage and the datapath instruction definitions.

There is a need for intermediate signals and this is handled in lines 43 - 46. Line 45 shows that TC is a function of UnD and the AllOnes and AllZeros flags provided by the datapath. 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 although only four operations are needed to be implemented, all 8 of the datapath instructions are used. By judicious selection of their address position and duplication, the logic to control them has been greatly simplified. The PI_DYN bit has been enabled so the CFB_EN bit of each instruction controls the ASRC input to the ALU.

- For the load instructions, the parallel input is passed through the ALU and placed in A0.
- For the count up instruction, A0 is incremented by ALU and fed back to A0.
- For the count down instruction, A0 is decremented by ALU and fed back to A0
- For the hold instructions A0 is passed through the ALU and goes nowhere. It does nothing.

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 to connect
/*  input   [02:00]     */  .cs_addr({PLn, UnD, CEn}), // Connect to control logic
/*  output              */  .z0(AllZeros),             // Flag set when A0 == 0x00
/*  output              */  .ff0(AllOnes),             // Flag set when A0 == oxff
/*  input [07:00]       */  .pi(D[7:0]),               // Data connecte to PI
/*  output [07:00]      */  .po(Q[7:0])                // PO connected to output
```

These parameters do the following:

- Connect a clock signal to the datapath clock circuitry
- Provide three inputs to control the instruction processing
- Connect the “all 0s” flag to an output
- Connect the “all 1s” flag to an output
- Connect the parallel input to the datapath PI
- Connect the datapath PO to the parallel 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