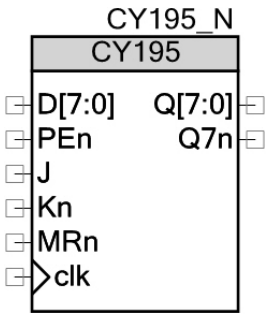
8-Bit Parallel Access Shift Register

CY195 v1.0



Features

- Serial data transfer
- Parallel-to-serial data transfer
- Serial-to parallel data transfer
- Parallel data transfer
- Simple to deconstruct

General Description

This shift register is based on the 74HC195 and performs serial, parallel, serial-to-parallel or parallel-to-serial data transfers. It operates on two primary modes: shift (Q0 → Q1) and parallel load. These are controlled by the state of the parallel load enable (PEn) active low input. Serial data enters the first flip-flop (Q0) via the J and K inputs when the PEn input is HIGH and shifted one bit in the direction Q0 → Q1 → Q2 → Q3. The J and Kn inputs provide the flexibility of a JK type input for special applications and by tying the pins together, a simple D-type input for general applications. The asynchronous master reset (MRn) active load input resets the Q[7:0] to 0x00. Its operation is independent of any other input.

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	parallel data input
PEn	input	parallel load enable (active low)
J	input	first stage J-input)
Kn	input	first stage K-input (active low)
MRn	input	asynchronous master reset (active low)
clk	input	clock input

Pin	Type	Function
Q7[7:0]	outputs	parallel outputs
Q7n	output	inverted output from the last stage

Function Table

Operating Modes	Inputs					Outputs			
	MRn	clk	PEn	J	Kn	Q0	Q1-Q6	Q7	Q7n
Reset	0	x	x	x	x	0	0-0	0	1
parallel load	1	↑	0	x	x	D0	D1-D6	D7	~D7
serial shift	1	↑	1	0	0	0	Q0-Q5	Q6	~Q6
	1	↑	1	0	1	Q0	Q0-Q5	Q6	~Q6
	1	↑	1	1	0	~Q0	Q0-Q5	Q6	~Q6
	1	↑	1	1	1	1	Q0-Q5	Q6	~Q6

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 first review the CY161 component. It also shows how to route a serial input into a register.

For this shifter there are two different operations.

- Reset
- Parallel load
- Serial load
- Hold (do nothing)

While the reset operation will be handled with datapath reset circuitry, the other three operations need to be implemented with datapath instructions.

Using PSoC Creator, open the CY195 example project to see the project schematic (*TopDesign.cysch*). It has a CY195 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

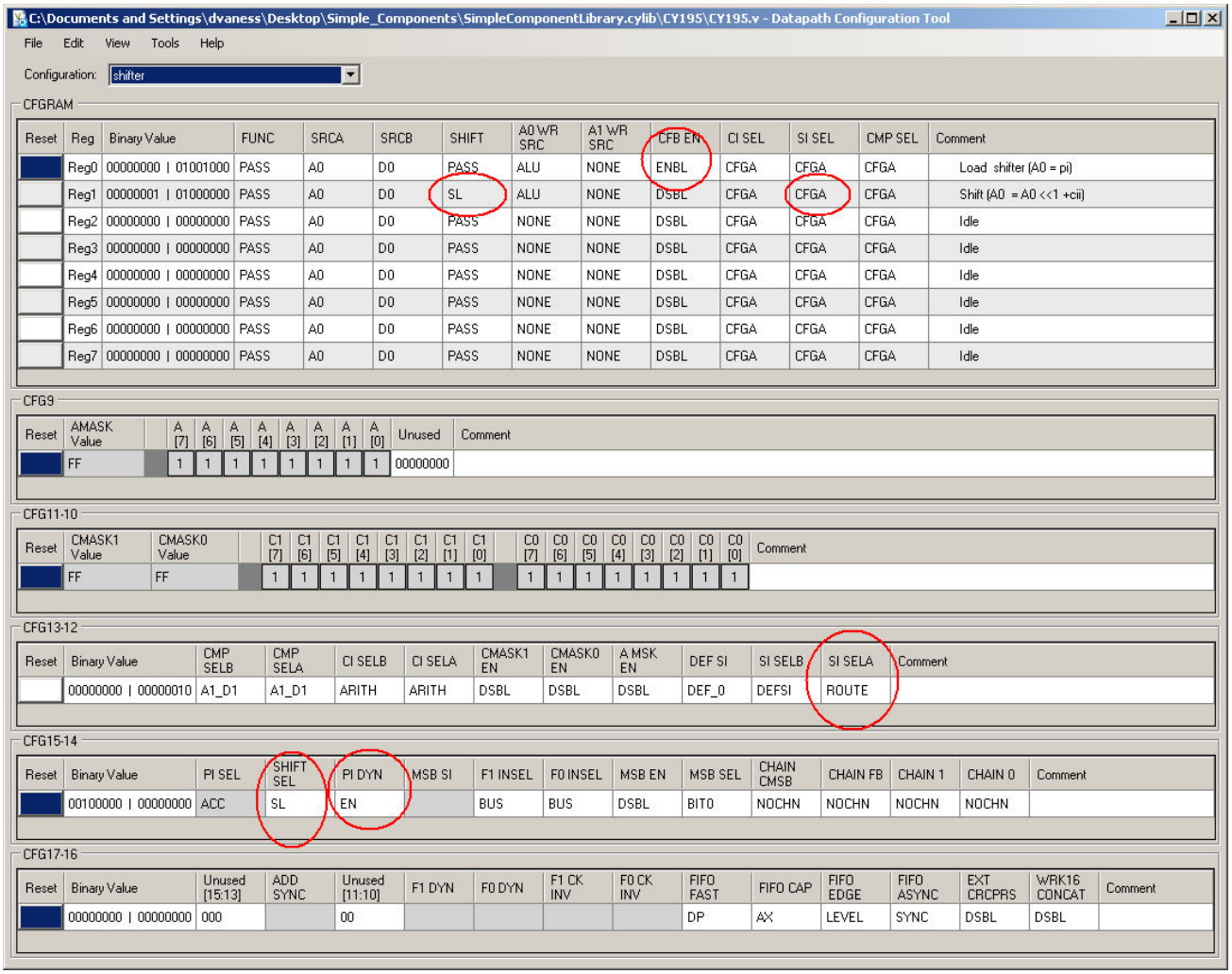
the CYCC_CY195_v1_0 component. Click **OK** and the following files are shown for the component:

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

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

Open Verilog file and notice that at lines 24 – 31, these definitions were passed to this Verilog file when it was created. The first 18 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 35 - 39. 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.



Only two operations are needed to be implemented. The PI_DYN bit has been enabled so the CFB_EN bit of each instruction controls the ASRC input to the ALU. The datapath is configured for left shit operations (SO on left SI on right). SI SELA is set for ROUTE to connect the serial input to the shift register.

- For the load instructions the parallel input is passed through the ALU and placed in A0.
- For the shift operation A0 is passed through the ALU and shifted with a serial input and passed back to A0.

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          */ .reset(reset),
    /* input          */ .clk(clk),              // Datapath clock
```

```
/* input  [02:00] */ .cs_addr({2'b0,PEn}),  // Control logic for instructiond
/* input          */ .route_si(ShiftIn),    // Serial input for shifter
/* input  [07:00] */ .pi(D[7:0]),           // Parallel input to shifter
/* output [07:00] */ .po(Q[7:0])            // Parallel  output from shifter
```

These parameters do the following:

- Connects a reset signal to the datapath reset circuitry
- Connects a clock signal to the datapath clock circuitry
- Controls datapath instruction processing
- Routes the serial input signal into the shifter
- Connects the parallel input into the datapath P1
- 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