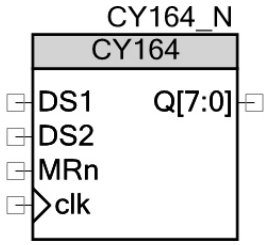
8-Bit Serial-In/Parallel-Out Shift Register

CY164 v1.0



Features

- Gated serial data inputs
- Positive-edge triggered clock
- Asynchronous master reset
- Simple to deconstruct

General Description

This shift register is based on the 74HC164. It is an 8-bit shift register with an asynchronous master reset. Synchronous operation is implemented with positive-edge triggered logic. The shifter, Q[7:0] (datapath register A0), is reset to zero with master reset input (MRn) providing an asynchronous clear. Data is entered serially through a combination two inputs (DS1 and DS2). Either input can be used as an enable for the 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
DS1	Inputs	input control
DS2	Input	input control
MRn	Input	master reset (active low)
clk	Input	clock input
Q[7:0]	Outputs	shifter outputs

Function Table

Operating Modes	Inputs				Outputs	
	MRn	clk	DS1	DS2	Q0	Q1 - Q7
reset	0	x	x	x	0	0 - 0
shift	1	↑	0	0	0	Q0 – Q6
	1	↑	0	1	0	Q0 – Q6
	1	↑	1	0	0	Q0 – Q6
	1	↑	1	1	1	Q0 – 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 please first review the CY161 component. It also shows how to route a serial input into a register. It requires only a single datapath configuration (instruction).

For this shifter there are two different operations.

- Reset
- Shift Let

While the reset operation will be handled with datapath reset circuitry, the shift left operation needs to be implemented with a datapath instruction.

Using PSoC Creator, open the CY164 example project to see the project schematic (*TopDesign.cysch*). It has a CY164 component connected to input switches, output LEDs, and a clock.

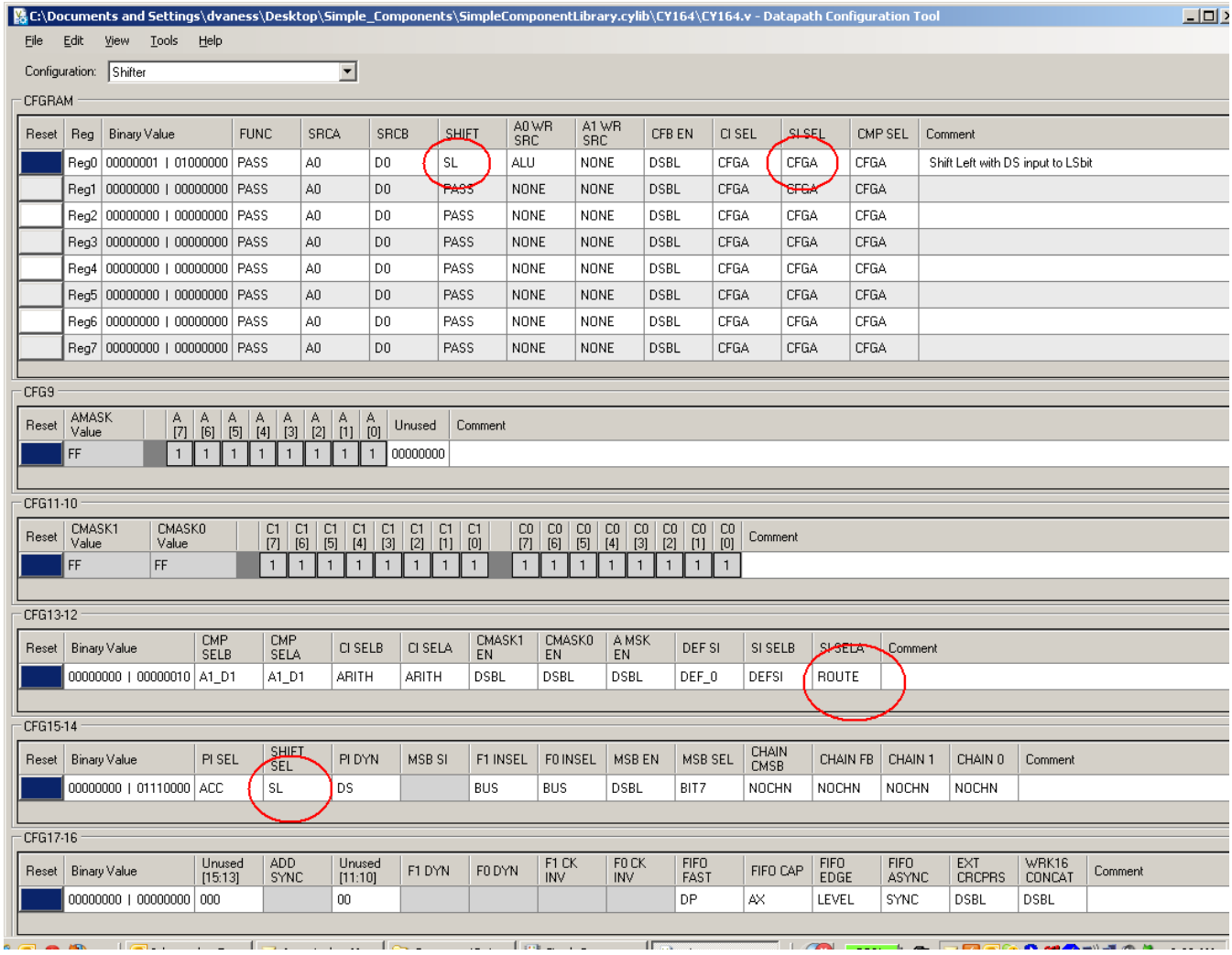
In the Workspace Explorer, click the **Components** tab. Then, right-click on the project and select **Import Component**. Navigate to where the CYCC_SimpleComponentLibrary project is, and select the CYCC_CY164_v1_0 component. Click **OK** and the following files are shown for the component:

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

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

Open the Verilog file and notice that at lines 24 – 28, these definitions were passed to this Verilog file when it was created. The first 22 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 32 - 35. 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:



This has only a single operation. Note that SHIFT_SEL was set to SL. This selects the serial in to come into the right side and the serial out to go out the left side. The instruction takes A0 and passes it through the ALU. Because the SHIFT bit in the instruction is set to SL, the output of the ALU is shifted left. To route the serial input, SI_SELA is set to ROUTE and the SI_SEL bit in the instruction is set to CFGA. If the bit had been set to CFGB then a 0 would have been shifted in. This passes back into 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),      // Reset input to the datapath
/*  input               */  .clk(clk),          // Clock input to the datapath
/*  input   [02:00]     */  .cs_addr(3'b0),     // Only use first instruction
/*  input               */  .route_si(Din),     // Input into shift register
/*  output [07:00]      */  .po(Q[7:0])         // Route PO to parallel output
```

These parameters do the following:

- Connects a reset signal to the datapath reset circuitry
- Connects a clock signal to the datapath clock circuitry
- Allows the first instruction to be continually processed
- Routes the serial input signal into the shifter
- Connects 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.