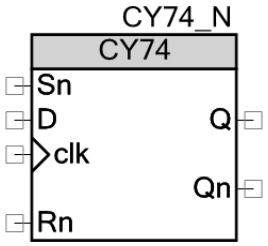
D-Type Flip-Flop with Set and Reset

CY74 v1.0



Features

- Synchronous positive edge triggered clock
- Asynchronous reset
- Synchronous set
- Complementary outputs
- Simple to deconstruct

General Description

This flip-flop is based on the 74HC74. It is a synchronous positive edge triggered D-type flip-flop. It differs from the original in that while the reset (Rn) input is asynchronous, the set (Sn) input had to be made synchronous because of the architecture of the CPLD used to implement it.

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 CY182 component before attempting this component.

Pin Description

Pin	Type	Function
Sn	input	synchronous set (active low)
D	input	D input to flip-flop
clk	input	clock input
Rn	input	asynchronous reset (active low)
Q	output	flip-flop output
Qn	output	complementary flip-flop output (active low)

Function Table

Operating Modes	Inputs			Outputs	
	clk	Rn	Sn	Q	Qn
reset	x	0	x	0	1
set	↑	1	0	1	0
load	↑	1	1	D	~D

Deconstructing the Component

This component is a good example of how to implement combinatorial and registered logic with Verilog. It is not necessary to know Verilog to be able to read Verilog. It looks like most any modern high-level programming language.

This datasheet will mark specific points in the code to help understand it. Look at enough examples and you will understand it and be able to design your own Verilog-based components.

Using PSoC Creator, open the CY74 example project to see the project schematic (*TopDesign.cysch*). It has a CY74 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_CY74_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 four inputs and two outputs. It looks like the symbol shown on the first page. There are no additional symbol parameters.

Open the Verilog file and you will see the following.

TopDesign.cysch | *CY74.v | CY74.cysym | Start Page | SimpleComponents.cydwr

```
1
2  //`#start header` -- edit after this line, do not edit this line
3  // ========================================================
4  // Simple CY74 D-type flip-flop to demonstrate how to iimplement
5  // logic in UDBs with Verilog
6  //
7  // ========================================================
8  `include "cypress.v"
9  //`#end` -- edit above this line, do not edit this line
10 // Generated on 02/25/2013 at 09:06
11 // Component: CY74
12 module CY74 (
13     output  reg  Q,       // Ouput
14     output  wire Qn,      // Complemaentary output
15     input   wire clk,     // Clock input
16     input   wire D,       // D flip-flop input
17     input   wire Rn,      // Reset (active low)
18     input   wire Sn       // Set (active low)
19 );
20
21 //`#start body` -- edit after this line, do not edit this line
22     assign Qn = ~Q;                     // Qn is complementary to Q;
23     always @(posedge clk or negedge Rn)
24     begin
25         if (~Rn)
26             Q <= 1'b0;                  // Rn has highest priority
27         else if (~Sn)
28             Q <= 1'b1;                  // Followed by Sn
29         else
30             Q <= D;                     // and, at last, D
31     end
32 //`#end` -- edit above this line, do not edit this line
33 endmodule
34 //`#start footer` -- edit after this line, do not edit this line
35 //`#end` -- edit above this line, do not edit this line
```

Note that at lines 13 – 18, these definitions were passed to this Verilog file when it was created. There are two types of assignments. The combinatorial logic type is shown in line 22 while registered assignments are shown in lines 26, 28 and 29. The “always” command is how Verilog defines the flip-flop’s logic. It is set up to change on the raising edge of the clock or the falling edge of Rn. If “`or negedge Rn`” is removed from line 23, then Rn becomes a synchronous control signal.

Lines 25 – 30 shows that Verilog has control statements. These lines of code could have been replaced with the following Boolean implementation.

```
Q = Rn & ( ~Sn | D);
```

Verilog’s control statements allow for more readable code. This frees the user from having to implement a complex design with only Boolean logic tools.

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