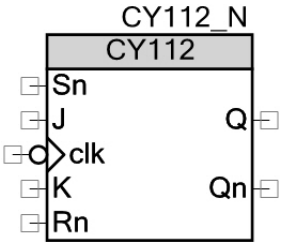
JK Flip-Flop with Set and Reset

CY112 v1.0



Features

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

General Description

This flip-flop is based on the 74HC112. It is a synchronous negative edge triggered JK 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 and CY74 components before attempting this component.

Pin Description

Pin	Type	Function
Sn	input	synchronous set (active low)
J	input	J input to flip-flop
clk	input	Clock input
K	input	K input to flip-flop
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	J	K	Q	Qn
Reset	x	0	x	x	x	0	1
Set	↓	1	0	x	x	1	0
Load	↓	1	1	0	0	Q	Qn
	↓	1	1	0	1	0	1
	↓	1	1	1	0	1	1
	↓	1	1	1	1	Qn	Q

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 come to understand it and be able to design your own Verilog-based components.

Using PSoC Creator, open the CY112 example project to see the project schematic (*TopDesign.cysch*). It has a CY112 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_CY112_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 see 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 the Verilog file and you will see the following.

Start Page | TopDesign.cysch | *CY112.v

```
1
2  //`#start header` -- edit after this line, do not edit this line
3  // ========================================================
4  // Simple CY112 JK-type flip-flop to demonstrate how to implement
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 13:00
11 // Component: CY112
12 module CY112 (
13     output  reg  Q,           // Output
14     output  wire Qn,          // Complementart Output
15     input   wire clk,         // clock (active on neg edge)
16     input   wire J,           // J input
17     input   wire K,           // K input
18     input   wire Rn,          // Reset (active low) async
19     input   wire Sn           // Set (active low )
20 );
21
22 //`#start body` -- edit after this line, do not edit this line
23     assign Qn = ~Q;                 // Complementary ouput;
24     always @(negedge clk or  negedge Rn)
25       begin
26         if (~Rn)                    // Rn has highest priority
27           Q <= 1'b0;
28         else if (~Sn)               // Followed by Sn
29           Q <= 1'b1;
30         else
31           begin
32             case ({J,K})            // Case statement allows look up table simplicity
33               2'b00 : Q <= Q;
34               2'b01 : Q <= 1'b0;
35               2'b10 : Q <= 1'b1;
36               2'b11 : Q <= ~Q;
37             endcase
38           end
39       end
40 //`#end` -- edit above this line, do not edit this line
41 endmodule
42 //`#start footer` -- edit after this line, do not edit this line
43 //`#end` -- edit above this line, do not edit this line
```

Note that at lines 13 – 19 the 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 23 while registered assignments are shown in lines 27, 29 and 33-36. The “always” command is how Verilog defines the flip-flop’s logic. It is set up to change on the falling edges of the clock or Rn. If “or negedge Rn” is removed from line 24, then Rn becomes a synchronous control signal.

Lines 25 – 39 shows that Verilog has control statements. For the J K signals, the case statement is used. Its function is apparent. These lines of code could have been replaced with the following Boolean implementation.

```
Q = Rn & (~Sn | ((J & ~Q)|(~K & Q)));
```

Verilog's control statements allow for more readable code. This frees the user from having to implement complex design with only Boolean logic tools. The case statement makes the code as easy to read as a look-up table.

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