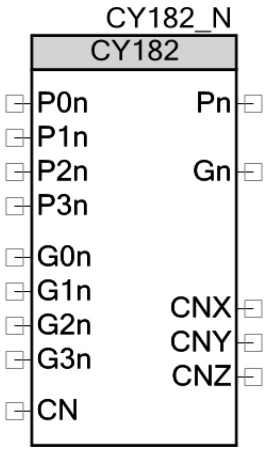
Look-Ahead Carry Generator

CY182 v1.0



Features

- Provides carry look-ahead across a group of four ALUs
- Multi-level look-ahead high speed processing
- Simple to Disassemble

General Description

This look-ahead carry generator is based on the 74HC182. It accepts up to four active low carry propagates (P0n – P3n) and four carry generates (G0n-G3n). It provides the anticipated carries. This component also has an active low carry propagate (Pn) output and an active low carry generate (Gn).

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 buit through its deconstruction.

Pin Description

Pin	Type	Function
G0n to G3n	inputs	carry generate inputs (active low)
P0n to P3n	inputs	carry propagate inputs (active low)
CN	input	carry input (active low)
Pn	output	carry propagate output (active low)
Gn	output	carry generate output (active low)
CNX, CNY,CNZ	outputs	function outputs

Function Table

Inputs									Outputs				
CN	**G0n**	**P0n**	**G1n**	**P1n**	**G2n**	**P2n**	**G3n**	**P3n**	**CNX**	**CNY**	**CNZ**	**Gn**	**Pn**
x	1	1							~1				
0	1	x							~1				
x	x	x	1	1						~1			
x	1	1	1	x						~1			
0	1	x	1	x						~1			
x	x	x	x	x	1	1					~1		
x	x	x	1	1	1	x					~1		
x	1	1	1	x	1	x					~1		
0	1	x	1	x	1	x					~1		
	x		x	x	x	x	1	1				1	
	x		x	x	1	1	1	x				1	
	x		1	1	1	x	1	x				1	
	1		1	x	1	x	1	x				1	
		1		x		x		x					1
		x		1		x		x					1
		x		x		1		x					1
		x		x		x		1					1

Deconstructing the Component

This component is a good example of how to implement combinatorial 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 CY182 example project to see the project schematic (*TopDesign.cysch*). It has a CY182 component connected to input control registers, 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_CY182_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 nine inputs and five 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 | CY182.cysym | **CY182.v** | SimpleComponents.cydwr

```
1
2  //`#start header` -- edit after this line, do not edit this line
3  // ================================================================
4  // Simple CY182 Look-ahead carry generator to demonstrate how
5  // to implement combinatorial 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 18:32
11 // Component: CY182
12 module CY182 (
13     output  wire CNX,  // Function output
14     output  wire CNY,  // Function output
15     output  wire CNZ,  // Function output
16     output  wire Gn,   // Carry generate output (active low)
17     output  wire Pn,   // Carry propagate output (active low)
18     input   wire CN,   // Carry Input
19     input   wire G0n,  // Carry generate input (active low)
20     input   wire G1n,  // Carry generate input (active low)
21     input   wire G2n,  // Carry generate input (active low)
22     input   wire G3n,  // Carry generate input (active low)
23     input   wire P0n,  // Carry propagate input (active low)
24     input   wire P1n,  // Carry propagate input (active low)
25     input   wire P2n,  // Carry propagate input (active low)
26     input   wire P3n   // Carry propagate input (active low)
27 );
28
29 //`#start body` -- edit after this line, do not edit this line
30     assign  CNX = ~((G0n & P0n)|(~CN & G0n));
31     assign  CNY = ~((G1n & P1n)|(G0n & P0n & G1n)|(~CN & G0n & G1n));
32     assign  CNZ = ~((G2n & P2n)|(G1n & P1n & G2n)
33                   |(G0n & P0n & G1n & G2n)|(~CN & G0n & G1n & G2n));
34     assign  Gn  = (G3n & P3n)
35                  |(G2n & P2n & G3n)
36                  |(G1n & P1n & G2n & G3n)
37                  |(G0n & G1n & G2n & G3n);
38     assign  Pn  = P0n | P1n | P2n | P3n;
39 //`#end` -- edit above this line, do not edit this line
40 endmodule
41 //`#start footer` -- edit after this line, do not edit this line
42 //`#end` -- edit above this line, do not edit this line
```

Note that at lines 13 – 26, the definitions were passed to this Verilog file when it was created. Lines 30 through 38 show that there are five assignments, one for each output. Each assign instruction uses a semicolon as a delimiter. An instruction can span multiple lines as shown in lines 32 and 33. Lines 34 through 37 shows how, with proper linefeeds and spaces, it can be made apparent that this instruction is just the ORing of a group AND terms. It is easy to compare it with its function table definition on the first page.

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