

## Carry-Lookahead Adders

Arithmetic logic blocks, such as adders and subtractors, are increasingly becoming performance bottlenecks in high-performance logic designs. Carry-lookahead adders are generally faster than cascaded adders because they reduce the time needed to generate carry propagation. The carry-lookahead can be achieved if the input carry bit for stage  $i$  is generated directly from the inputs to the preceding stages  $i - 1, i - 2, \dots, i - k$  rather than allowing the carry bit to be cascaded and rippled from stage to stage. An  $n$  - bit carry-lookahead adder can be constructed using  $k$  stages, each of which is a full adder stage modified by replacing its carry output line  $co$  by two signals called carry generate and propagate. These signals  $gi$  and  $pi$  are defined by the following logic:

$$gi = ai \cdot bi$$

$$pi = ai + bi$$

That is, a stage will generate a carry if both of its addend bits are '1', and it propagates carries if at least one of its addend bits is '1'.

Therefore, the carry signal that will be generated for the stage  $i + 1$  is defined as follows from the generate and propagate signals:

$$ci + 1 = gi + pi \cdot ci$$

If we recursively expand the  $ci$  term for each stage, and multiply out to obtain a two-level sum-of-products expression we can eliminate the carry ripple that is associated with cascaded adders. If this technique is followed, we can obtain equations for the carry out bits for each stage as shown below.

$$c1 = g0 + p0 \cdot c0$$

$$c2 = g1 + p1 \cdot c1$$

$$= g1 + p1 (g0 + p0 \cdot c0)$$

$$= g1 + p1 \cdot g0 + p1 \cdot p0 \cdot c0$$

$$c3 = g2 + p2 \cdot c2$$

$$= g2 + p2 (g1 + p1 \cdot g0 + p1 \cdot p0 \cdot c0)$$

$$= g2 + p2 \cdot g1 + p2 \cdot p1 \cdot g0 + p2 \cdot p1 \cdot p0 \cdot c0$$

$$c4 = g3 + p3 \cdot c3$$

$$= g3 + p3 (g2 + p2 \cdot g1 + p2 \cdot p1 \cdot g0 + p2 \cdot p1 \cdot p0 \cdot c0)$$

$$= g3 + p3 \cdot g2 + p3 \cdot p2 \cdot g1 + p3 \cdot p2 \cdot p1 \cdot g0 + p3 \cdot p2 \cdot p1 \cdot p0 \cdot c0$$

Each of the above equations corresponds to a circuit with only three levels of delay associated with it – one for the generate and propagate signals, and two for the sum-of-products shown. A carry-lookahead adder uses three-level equations such as these in each adder stage.

## Building Blocks of an $n$ -Bit Carry Lookahead Adder

**F3ADD (F3ADD\_1, F3ADD\_2):** A three-bit full adder with propagate and generate outputs

**PG1 .. PG4:** Carry/Borrow bit generator utilizing propagate and generate inputs

**F3ADD:** The F3ADD macro shown below performs the three-bit addition of  $a0 \dots a2 + b0 \dots b2$ , it also performs the propagate and generate functions. The propagate function determines if any of the addend bits are '1' by Oring each set of addend bits. If any of the addend bits are a one a propagate will be generated.

As shown below, we see that when either of a digit's addend bits are one, a propagate will be generated:

	a2	a1	a0	0	1	1
	b2	b1	b0	0	0	1
propagate $pi$	a2+b2	a1+b1	a0+b0	0	1	1

Note: only a single propagate will be produced although more than one may be generated as shown. The generate function determines if a carry to the next digit should be generated from the previous digit addition, as shown earlier in the binary addition basics section. Essentially, it performs the same function as the carry out of a regular adder, but does not incorporate the carry in signal in its logic. An example of how a generate bit is produced is shown below.

## Adder and Subtractor Macros in pDS and pDS+

Here we show a three-bit addition with a generate being produced:

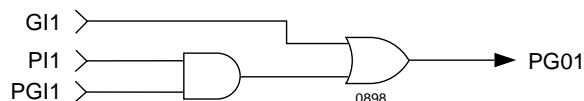
		1	1	1	0	- Carry bits from digit addition
a2	a1	a0	0	1	1	1 - 07
<i>gi</i>	b2	b1	b0	0	0	0 1 - 01
<hr/>			<hr/>			
			1	0	0	0 - Gives 08
Generate <i>gi</i>						

*gi* will be generated because  $= a_0 \cdot a_1 \cdot a_2 \cdot b_0$ ; here we can see that if  $a_0 \cdot b_0 = 1$  a carry will be generated for those two addend bits. Therefore, if that carry propagates to the next set of bits  $a_1 \cdot b_1$  and either of them are a one a carry will be generated to the next set. This function is recursive and if you look at the logic for the generate function below you will see that all combinations have been accounted for.

```
gi = (a0 · a1 · a2 · b0
# a1 · a2 · b1
# a0 · a2 · b0 · b1
# a2 · b2
# a0 · a1 · b0 · b2
# a1 · b1 · b2
# a0 · b0 · b1 · b2)
```

### Figure 1. Logic and Truth Table for the PG1 Macro

PGI1	PI1	GI1	PG01
x	x	1	1
1	1	x	1



**PG1 .. PG4:** The propagate-generate macros are carry bit generators utilizing propagate and generate inputs. The logic and truth table for the PG1 macro is shown in Figure 1.

The PG1 input of the macro is the carry in input from the initial stage of your adder, the PI1, and GI1 inputs are the propagate and generate inputs associated with the F3ADD macro outputs. Looking at the logic of the PG1 macro it can be seen that whenever a generate bit was produced from the F3ADD macro a carry out signal will be generated from the PG macro, assuming the PG macro is using the inputs from the F3ADD macro outputs. This also holds true if both of the propagate inputs into the PG1 are '1' since the initial carry in would be a '1' and one of the addend bits is '1', this would result in a carry out. Basically, the PG1 macro performs the function associated with the carry out of a regular adder.

As shown previously in the carry-lookahead adder section, the carry signal that will be generated for the stage  $i + 1$  is defined as follows from the generate and propagate signals:

$$ci + I = gi + pi \cdot ci \text{ (where } ci + I = PG0I, gi = GII, pi = PII, \text{ and } ci = PGII)$$

Therefore:  $PG01 = GI1 + PI1 \cdot PGII$

If we recursively expand the PG01 term for each stage, i.e. PG02 .. PG0n and multiply out to obtain a two-level sum-of-products expression, we can eliminate the carry ripple that is associated with cascaded adders. If this technique is followed, we can obtain equations for any PG0n bits for each stage as shown previously, but substituting the generate and propagate signals into the *ci* equations as we did above.

For stage two (macro PG02):

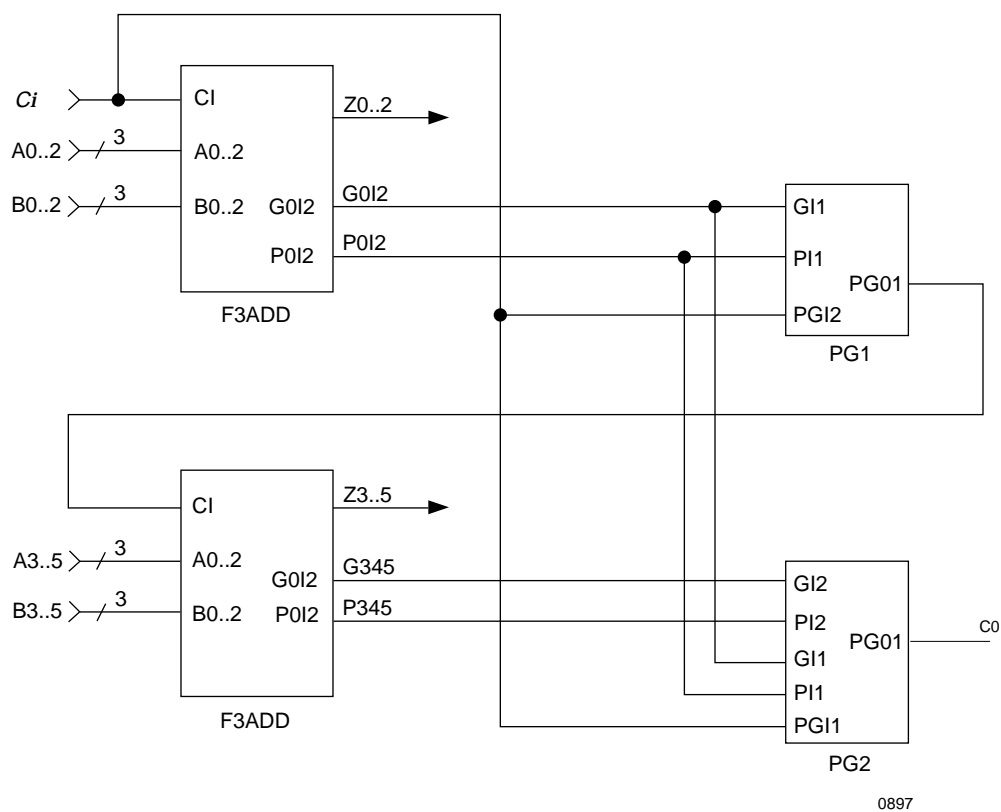
$$\begin{aligned} c2 &= gl + pl \cdot cl \\ &= gl + pl \ (g0 + p0 \cdot c0) \\ &= gl + pl \cdot g0 + pl \cdot p0 \cdot c0 \end{aligned}$$

$$PG02 = GI2 + PI2 \cdot GI1 + PI2 \cdot PI1 \cdot PG01$$

Here we show a six-bit adder utilizing two F3ADD macros, PG1 and PG2. Remember that the propagate and generate inputs to the PG*n* macro is associated with that stages adders outputs. That is, for PG02, GI1 and PI1 would come from the first adder, and GI2 and PI2 would come from the second adder.

## Adder and Subtractor Macros in pDS and pDS+

Figure 2. Six-Bit Adder

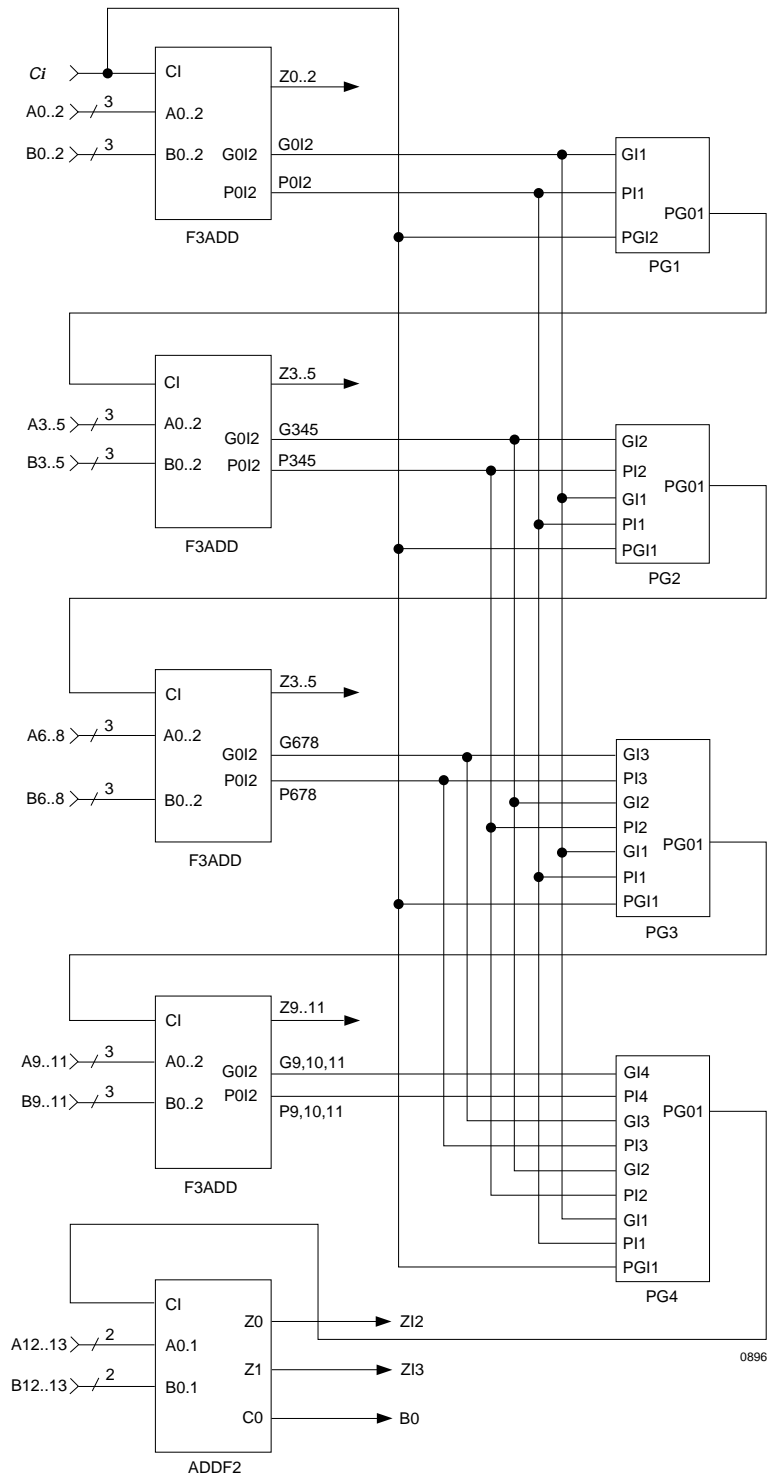


# Adder and Subtractor Macros in pDS and pDS+

In the case of the six-bit adder, the *co* will be the PG02 of the second propagate and generate macro PG2. But for adders that do not have a multiple of three, one of the other regular adders should be used to account for the

extra bits and the *ci* of the adder would be driven by the last PG0*n* in that network. This is shown in the figure below.

**Figure 3. 14-Bit Adder**







Copyright © 1996 Lattice Semiconductor Corporation.

E<sup>2</sup>CMOS, GAL, ispGAL, ispLSI, pLSI, pDS, Silicon Forest, UltraMOS, Lattice Logo, L with Lattice Semiconductor Corp. and L (Stylized) are registered trademarks of Lattice Semiconductor Corporation (LSC). The LSC Logo, Generic Array Logic, In-System Programmability, In-System Programmable, ISP, ispATE, ispCODE, ispDOWNLOAD, ispGDS, ispStarter, ispSTREAM, ispTEST, ispTURBO, Latch-Lock, pDS+, RFT, Total ISP and Twin GLB are trademarks of Lattice Semiconductor Corporation. ISP is a service mark of Lattice Semiconductor Corporation. All brand names or product names mentioned are trademarks or registered trademarks of their respective holders.

Lattice Semiconductor Corporation (LSC) products are made under one or more of the following U.S. and international patents: 4,761,768 US, 4,766,569 US, 4,833,646 US, 4,852,044 US, 4,855,954 US, 4,879,688 US, 4,887,239 US, 4,896,296 US, 5,130,574 US, 5,138,198 US, 5,162,679 US, 5,191,243 US, 5,204,556 US, 5,231,315 US, 5,231,316 US, 5,237,218 US, 5,245,226 US, 5,251,169 US, 5,272,666 US, 5,281,906 US, 5,295,095 US, 5,329,179 US, 5,331,590 US, 5,336,951 US, 5,353,246 US, 5,357,156 US, 5,359,573 US, 5,394,033 US, 5,394,037 US, 5,404,055 US, 5,418,390 US, 5,493,205 US, 0194091 EP, 0196771B1 EP, 0267271 EP, 0196771 UK, 0194091 GB, 0196771 WG, P3686070.0-08 WG. LSC does not represent that products described herein are free from patent infringement or from any third-party right.

The specifications and information herein are subject to change without notice. Lattice Semiconductor Corporation (LSC) reserves the right to discontinue any product or service without notice and assumes no obligation to correct any errors contained herein or to advise any user of this document of any correction if such be made. LSC recommends its customers obtain the latest version of the relevant information to establish, before ordering, that the information being relied upon is current.

LSC warrants performance of its products to current and applicable specifications in accordance with LSC's standard warranty. Testing and other quality control procedures are performed to the extent LSC deems necessary. Specific testing of all parameters of each product is not necessarily performed, unless mandated by government requirements.

LSC assumes no liability for applications assistance, customer's product design, software performance, or infringements of patents or services arising from the use of the products and services described herein.

LSC products are not authorized for use in life-support applications, devices or systems. Inclusion of LSC products in such applications is prohibited.

#### LATTICE SEMICONDUCTOR CORPORATION

5555 Northeast Moore Court  
Hillsboro, Oregon 97124 U.S.A.

Tel.: (503) 681-0118

FAX: (503) 681-3037

<http://www.latticesemi.com>

November 1996

---