# Carry-Save Addition Saves Logic, Time

Summing multiple operands is a common operation for signal processing applications. One such application requires summing eight, 16-bit operands to generate a 19-bit result. Pipelining is required to achieve the system's required set-up time ($t_S$) and clock-to-output time ($t_{CO}$). The straightforward means for this multi-operand addition, and the resulting logic complexity, are shown in the figure to the right. Two levels of pipeline registers are included, and 104 total registers are required. The external $f_{MAX}$ performance for the design is derived from the sum of $t_S$ (based on a 16-bit addition) and $t_{CO}$ (based on an 18-bit addition), both of which take four cascaded logic cells in a Cypress PASIC380 FPGA.

Carry-save addition (CSA) is a technique often used for summing the partial products associated with multiplication. CSA can be applied to this multi-operand adder design, and the tables here show the first level of the carry-save operation (see the tables).

Keep in mind that each outlined cell in the tables represents a three-input full adder; thus the basic CSA operation is to add the three inputs in a cell and save both the sum and the carry, then add these results to the remaining operands. For example, let's add these five 5-digit operands using a full adder:

| | | | | |
|---|---|---|---|---|
| a4 | a3 | a2 | a1 | a0 |
| b4 | b3 | b2 | b1 | b0 |
| c4 | c3 | c2 | c1 | c0 |
| d4 | d3 | d2 | d1 | d0 |
| e4 | e3 | e2 | e1 | e0 |

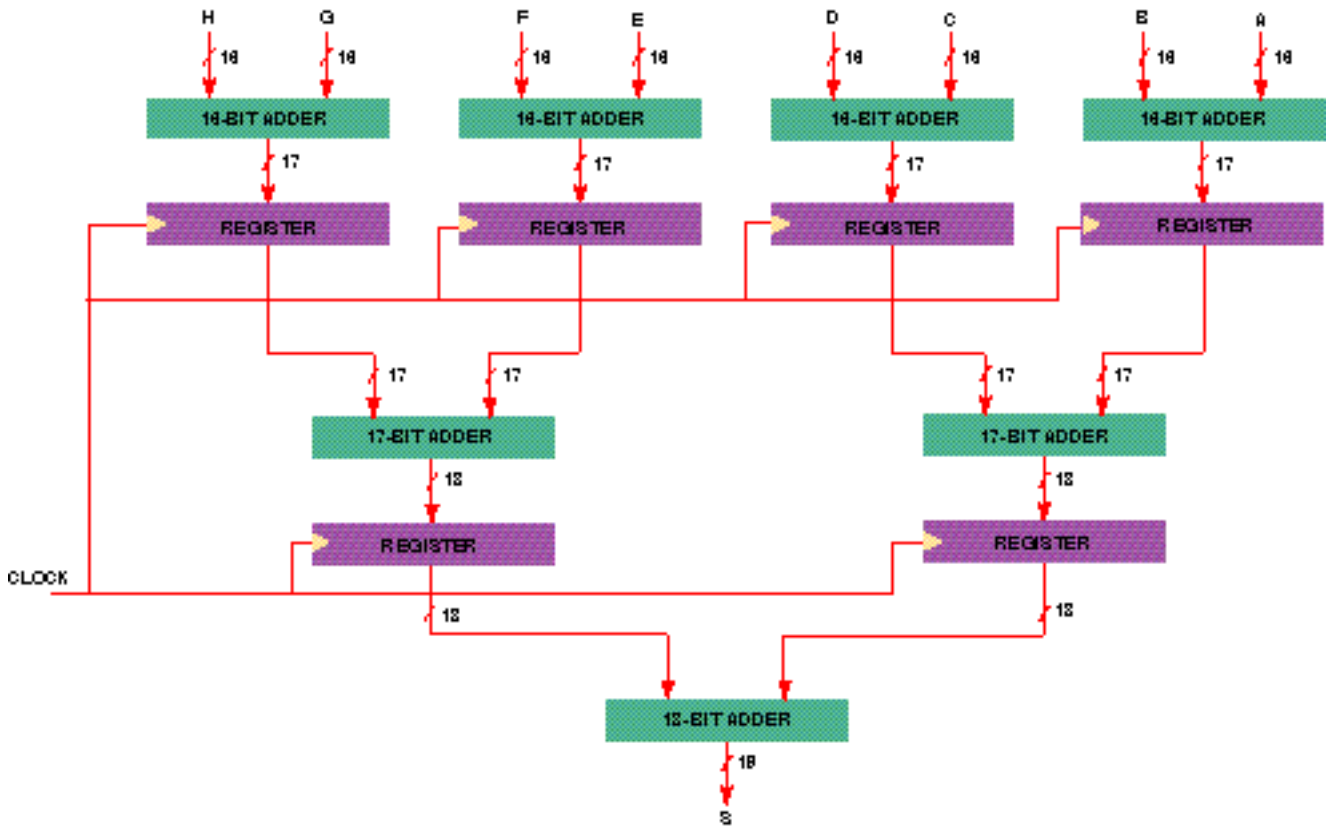
Using a full adder, *a0* + *b0* + *c0* produces a sum, *m0*, and a carry, *n1*,which is saved. Similarly, *a1* + *b1* + *c1* produce *m1* and *n1*—and so on through the digits of the operands. In this way, the original five operands reduce to:

| | | | | | | |
|---|---|---|---|---|---|---|
| | d4 | d3 | d2 | d1 | d0 | |
| | e4 | e3 | e2 | e1 | e0 | |
| | m4 | m3 | m2 | m1 | m0 | ← sums |
| n5 | n4 | n3 | n2 | n1 | | ← saved carrys |

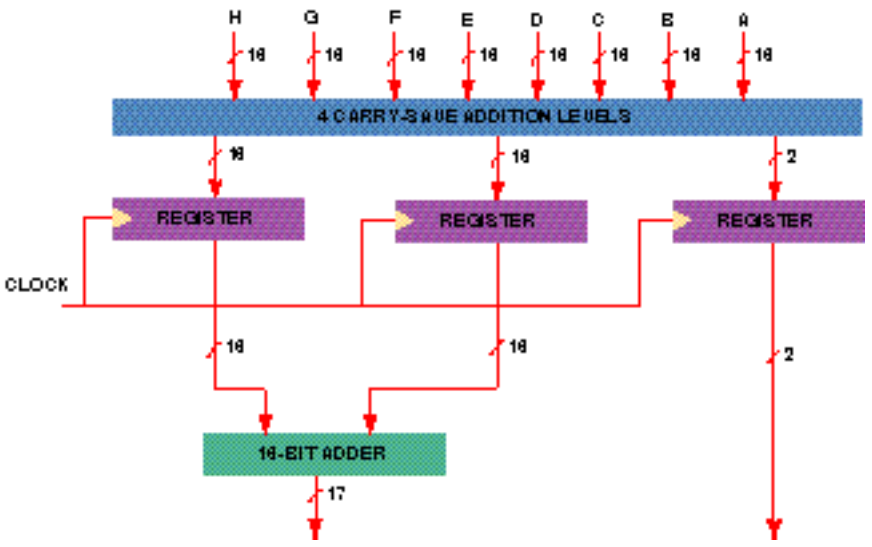
The next full-adder summings (*d0* + *e0* + *m0; d1* + *e1* + *m1*; etc.) produce another row of sums and another row of carrys, which leave three operand rows that can be summed in a full adder. The idea is to move "downward" from CSA level to CSA level in stepwise reductions leading to two operands, at most, which can be added in (say) a half adder.

The carry-save example circuit shown here requires four CSA levels to reduce the eight operands to two bits of the desired result plus two 16-bit operands that must still be added. A major advantage of the new implementation compared to the previous one is that only one pipeline register level is required instead of two. Also, only 34 registers are needed, and the total number of logic cells required for implementation is less than for the conventional design. The external performance of the new design is the same as before (in effect, there is just a single delay from CSA level to CSA level, since all the adders are in parallel), as $t_S$ is based on four CSA levels and $t_{CO}$ is based on a 16-bit addition. ❖

*For literature, visit the Cypress web site. See the appropriate site address (URL) for article 202 in the listing on the back cover.*




Conventional. Designed in the usual way as shown here, the logic for multiple-operand addition would require two levels of pipeline registers to assure that set-up and clock-to-output timing specifications are met. The design would require 104 registers and use four cascaded logic cells in a Cypress PASIC380 FPGA.




Save 70 registers. Using carry-save addition (CSA), the conventional design with its 104 registers reduces to the 34-register design shown here. The new design takes up less space in the FPGA, and its external performance is the same as the original.

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A15 | A15 | A13 | A12 | A11 | A10 | A9 | A8 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 |
| B15 | B14 | B13 | B12 | B11 | B10 | B9 | B8 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| C15 | C14 | C13 | C12 | C11 | C10 | C9 | C8 | C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| E15 | E14 | E13 | E12 | E11 | E10 | E9 | E8 | E7 | E6 | E5 | E4 | E3 | E2 | E1 | E0 |
| F15 | F14 | F13 | F12 | F11 | F10 | F9 | F8 | F7 | F6 | F5 | F4 | F3 | F2 | F1 | F0 |
| G15 | G14 | G13 | G12 | G11 | G10 | G9 | G8 | G7 | G6 | G5 | G4 | G3 | G2 | G1 | G0 |
| H15 | H14 | H13 | H12 | H11 | H10 | H9 | H8 | H7 | H6 | H5 | H4 | H3 | H2 | H1 | H0 |

Assignations. In the tables above, each box represents a full-adder cell. The tables show how the input operands to the first carry-save addition (CSA) level are assigned to 32 full-adder cells.

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | G15 | G14 | G13 | G12 | G11 | G10 | G9 | G8 | G7 | G6 | G5 | G4 | G3 | G2 | G1 | G0 |
| | H15 | H14 | H13 | H12 | H11 | H10 | H9 | H8 | H7 | H6 | H5 | H4 | H3 | H2 | H1 | H0 |
| | J15 | J14 | J13 | J12 | J11 | J10 | J9 | J8 | J7 | J6 | J5 | J4 | J3 | J2 | J1 | J0 |
| K16 | K15 | K14 | K13 | K12 | K11 | K10 | K9 | K8 | K7 | K6 | K5 | K4 | K3 | K2 | K1 | |
| | L15 | L14 | L13 | L12 | L11 | L10 | L9 | L8 | L7 | L6 | L5 | L4 | L3 | L2 | L1 | L0 |
| M16 | M15 | M14 | M13 | M12 | M11 | M10 | M9 | M8 | M7 | M6 | M5 | M4 | M3 | M2 | M1 | |

From level to level. The outputs of the first CSA level show that there are six operands remaining to be added; these outputs are the inputs to the second CSA level. Here, the table shows the outputs of the first CSA level. The input bits A0, B0, and C0 are supplied to a full adder having J0 and K1 as its sum and carry bit outputs. Similarly, L0 and M1 are the sum and carry bits for the addition of D0, E0, and F0. All the other boxes represent inputs being fed into full adders. The outputs of the first CSA level are input to a second CSA level, the outputs of the second CSA level are input to a third CSA level, and so on.