Axcelerator Carry-Connect Macros

Introduction

The Axcelerator dedicated carry-chain logic offers a very compact solution for implementing arithmetic functions without sacrificing performance. The AX architecture, on which Axcelerator devices are based, implements a new combinatorial cell (C-cell) that is an extension of the C-cell implemented in the SX family. C-cells are connected together with dedicated carry-connect resources. This document describes carry-chain logic in the Axcelerator devices. It goes through the details of the architecture, performance, and advantages of carry-chain logic.

The Combinatorial Cell

One of the main improvements of the new C-cell is the addition of carry-chain logic. The C-cell can be used in a carry-chain mode. However, if carry-chain logic is not required, it can be disabled to save power.

The C-cell features the following (Figure 1):

- 8-input MUX (data: D0-D3, select: A0, A1, B0, B1). User signals can be routed to any one of these inputs. Any of the C-cell inputs (D0-D3, A0, A1, B0, B1) can be tied to one of the four global clocks (CLKE/F/G/H).
- The inverting input (DB) can be used to drive the complement of any C-cell input.
- A carry input (FCI) and a carry output (FCO). The carry input signal of the C-cell is the carry output from the C-cell directly to the north.
- Carry-connect routing for the carry-chain logic.
- A hard-wired connection (direct connect) to the adjacent R cell (Register cell) for all C-cells on the east side of a SuperCluster.
- A control signal (CFN) that determines whether or not the C-cell is configured for carry-chain mode. It can also be set inactive to force FCO to zero to save power.

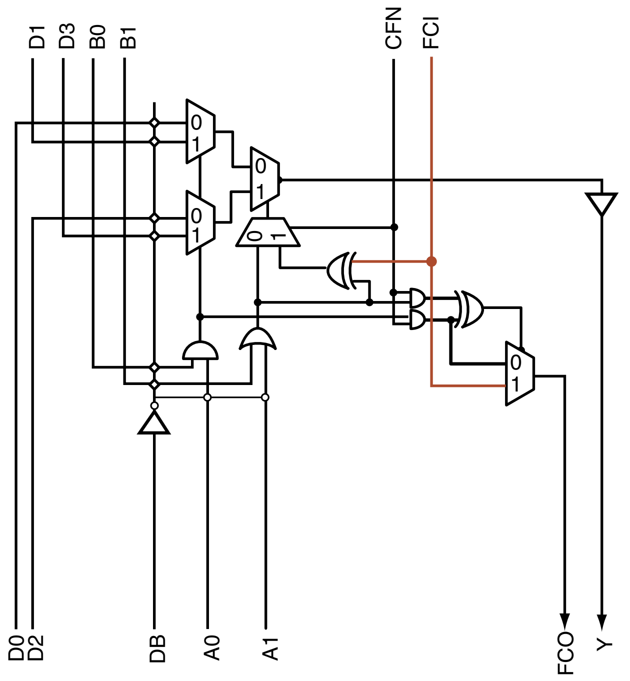


Figure 1 • C-Cell Block Diagram

This layout of the C-cell and the two-C-cell Cluster arrangement enable the implementation of over 4,000 functions of up to five inputs, as shown in Figure 2. For example, two C-cells can be used together to implement a 4-input XOR function with a single C-cell delay. This configuration is handled automatically for the user with Actel's extensive macro library (please see Actel's *Macro Library Guide* for a complete listing of available Axcelerator macros).

Carry Chain Model

The Axcelerator dedicated carry-chain logic offers a very compact solution for implementing arithmetic functions without sacrificing performance. To implement the carry-chain logic, the two C-cells in a Cluster are connected together so that the FCO (i.e., carry out) for the two bits is generated in a Carry-Look-Ahead scheme to achieve minimum propagation delay from the FCI (i.e., the carry in) to the two-bit Cluster. The two-bit carry logic is shown in Figure 3. The FCI of one Cluster is driven by the FCO of the Cluster immediately above it. Similarly, the FCO of one Cluster drives the FCI input of the Cluster immediately below it (Figure 4 on page 3). The carry-chain logic is selected via the CFN input. When carry logic is not needed, this signal is de-asserted to save power. The signal propagation delay between two C-cells in the carry-chain sequence is less than 0.1ns.

In the carry chain sequence, every two C-cells (in a Cluster) have a hard-wired connection to the adjacent R-cell (in the same Cluster) through the C-cell on the east side of the Cluster. This hard-wired connection results in a signal propagation time of less than 0.1ns.

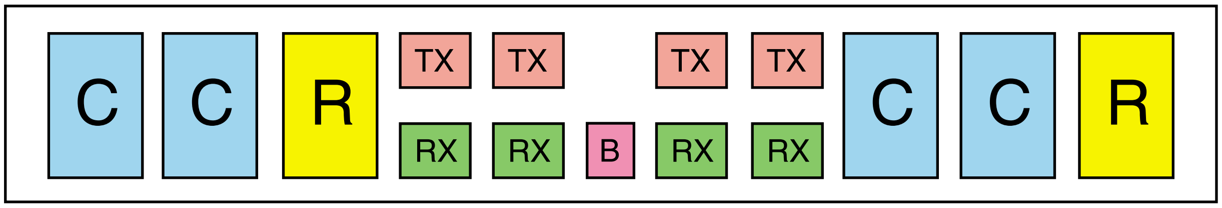


Figure 2* • *Axcelerator SuperCluster (Two Clusters)

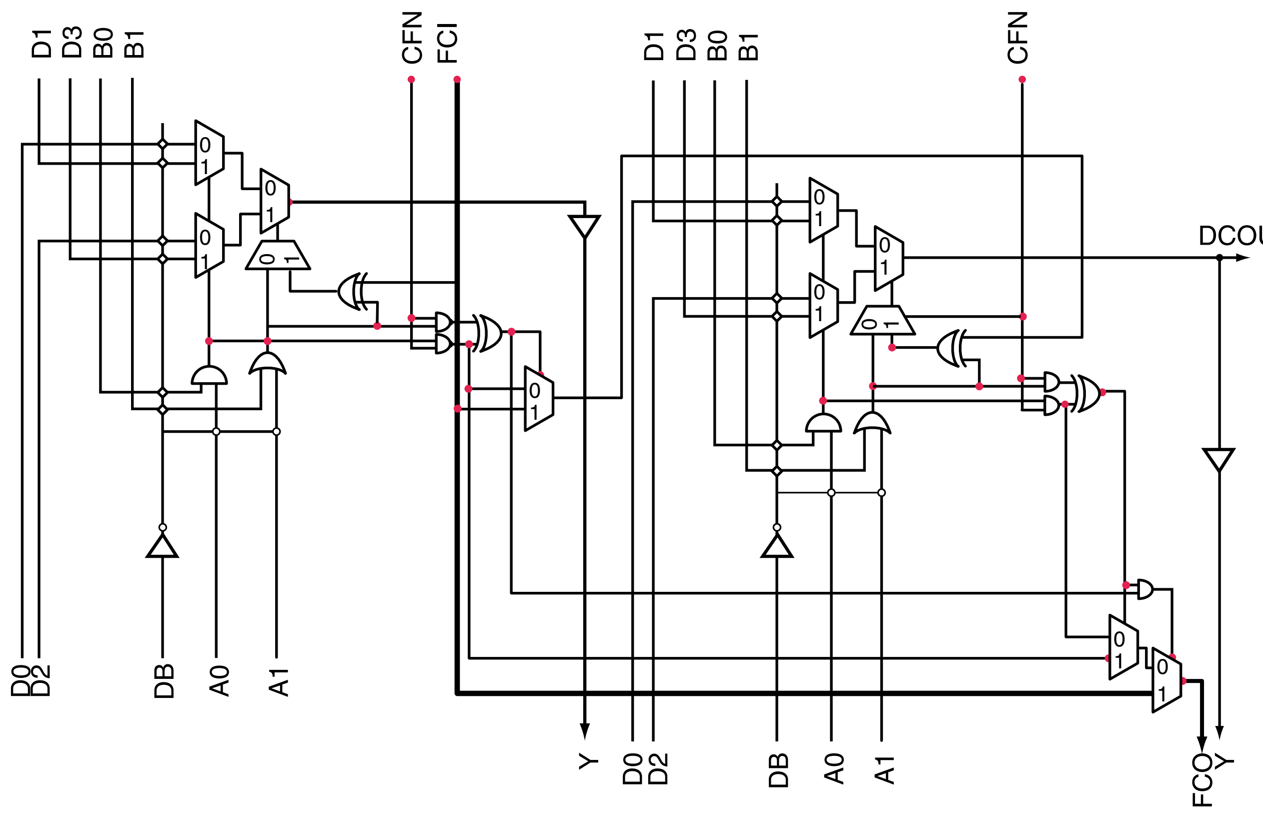


Figure 3* • *AX 2-bit Carry Logic

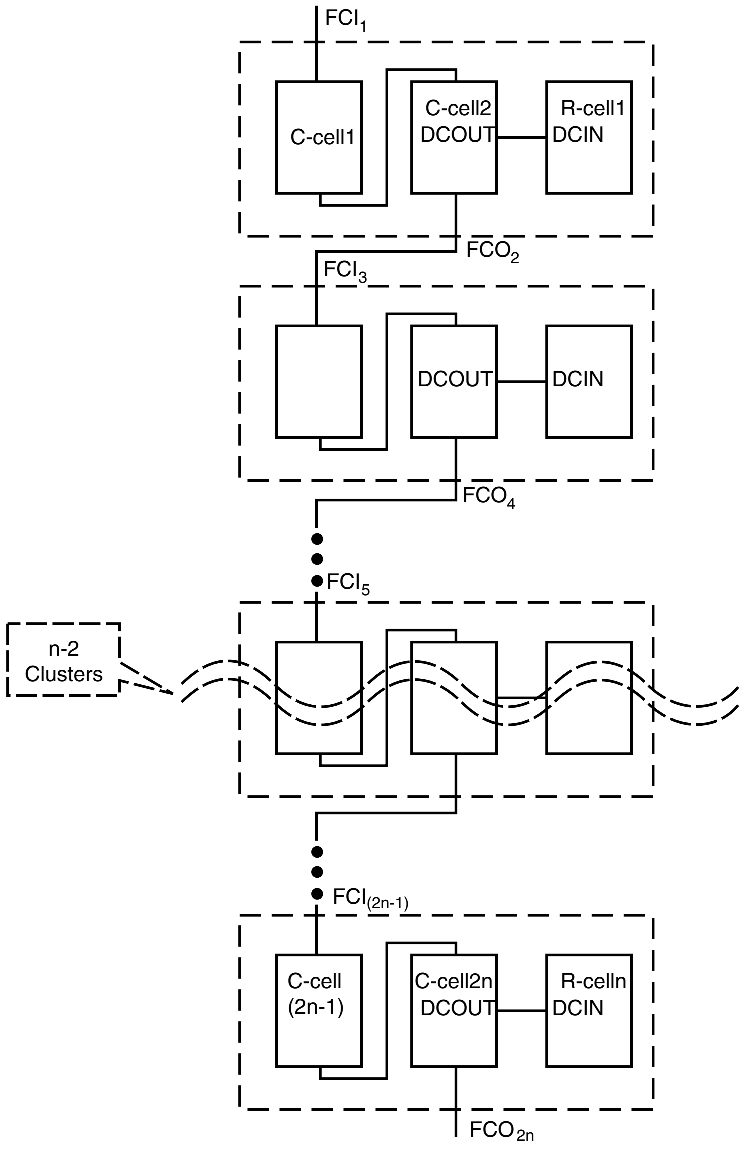


Figure 4 • Carry-Chain Sequencing of C-cells

Carry-Chain Macros

Carry-chain macros are designed to optimize the carry-chain structure offered in Axcelerator FPGAs for implementing various arithmetic functions. The following is a description of the different types of carry-chain macros:

0-Input Combinatorial	
FCO = 0	FCINIT_GND
FCO = 1	FCINIT_VCC
1-Input Combinatorial	
FCO = A	FCINIT_BUFF
CO = FCI	FCEND_BUFF
FCO = !A	FCINIT_INV
CO = !FCI	FCEND_INV

One-Bit Data Path Macros

The following hard macros make use of the dedicated fast carry chain – offering increased performance for widely used arithmetic functions.

Table 1 • Hard Macros

S = A ^ B ^ FCI
FCO = A B + A FCI + B FCI
	ADD1

S = !(A ^ B ^ AS) ^ FCI
FCO = (A !(B ^ AS) + A FCI + FCI !(B ^ AS)
	ADDSUB1

S = (A B) ^ PI ^ FCI
FCO = A B PI + A B FCI + PI FCI
	MULT1

S = A ^ !B ^ FCI
FCO = A !B + A FCI + !B FCI
	SUB1

Higher-level incrementer, decrementer, and comparator macros are constructed as soft macros by using hard macros as building blocks and connecting one of the inputs to a constant. Since the nets connected to the FCI and FCO pins are hard-wired, the initial FCI inputs of a group of one-bit data path macros can only be connected to one of the FCINIT macros. Likewise, the terminal FCO pin can only be connected to one of the FCEND macros. Each FCINIT and FCEND macro consumes one C-cell. Figure 4 presents examples of soft macros.

Performance

The dedicated carry-chain logic offers a very compact implementation for high-speed adders, subtractors, comparators, multipliers, and counters, achieving a significant improvement in speed and density. Table 2 compares the logic consumption in Axcelerator vs. SX-A for various arithmetic functions.

Please refer to the "Appendix" section on page 6 for details on the resulting area and speed for different types of multipliers. All multiplier options preceded by FC use the fast-carry logic.

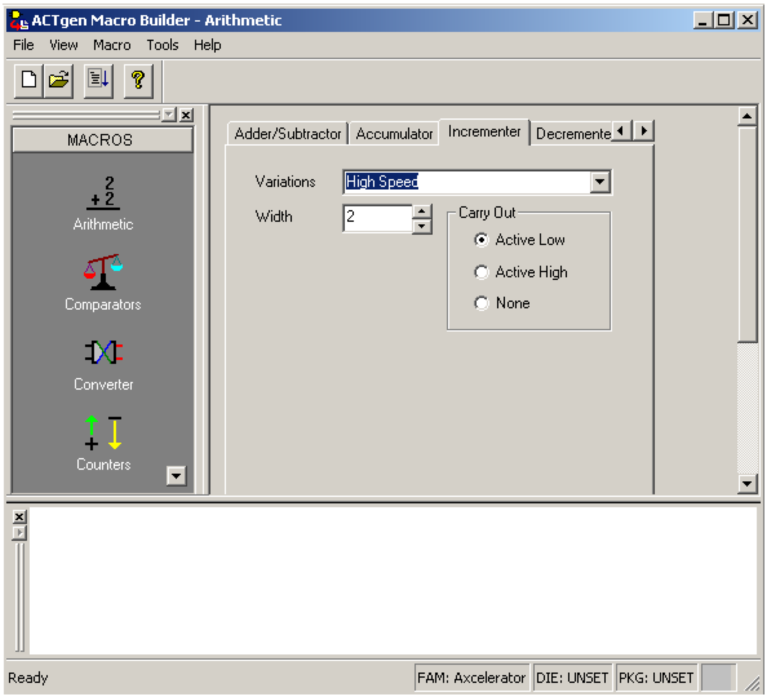
Table 2 • Device Utilization for Arithmetic Functions Implemented and Axcelerator and SX-A

A54SX32A		**AX500**	
Function	**Area**	**Function**	**Area**
16-bit adder	73 C-cells	16-bit adder	17 C-cells
32-bit adder	155 C-cells	32-bit adder	68 C-cells
12x12-bit multiplier	562 C-cells	16X16-bit multiplier	168 C-cells
16x16-bit multiplier	995 C-cells	32X32-bit multiplier	288 C-cells

User Flow

There are two ways to use carry-chain logic to implement functions:

1. Using Actel's ACTgen macro generator: In this case, the user can choose whether or not to use carry-chain logic. To use carry-chain logic, the user has to select the "high speed" option in the variations menu as illustrated in Figure 5 on page 5.
2. Using carry-chain macros manually: This can be done by instantiating the carry-chain macros in the design. A list of these macros is included in this document. Detailed descriptions can be found in Actel's *Macro Library Guide*.



Variations: High Speed

Width: 2

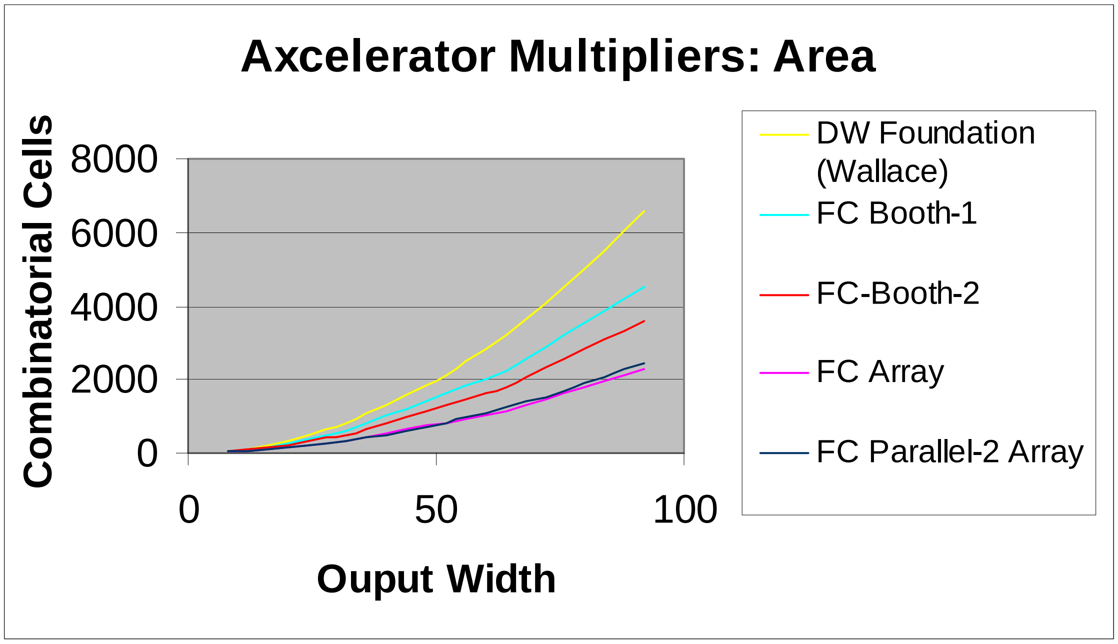
Carry Out: [✓ Active Low] Active High None

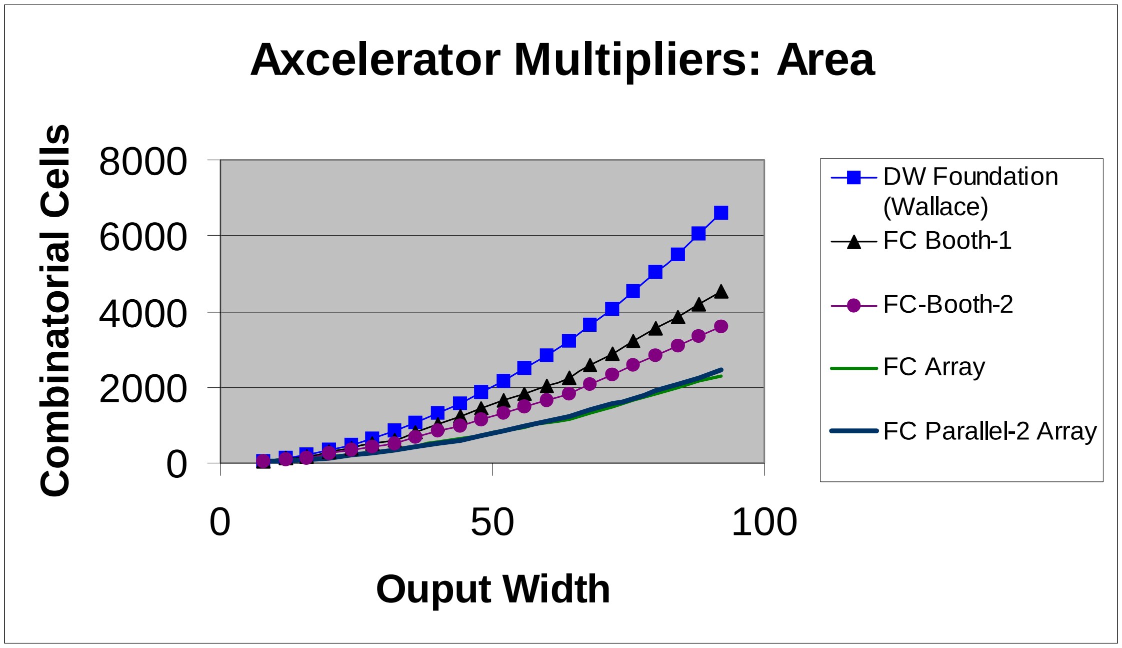
Figure 5 • *Selecting Carry-Chain Logic in ACTgen*

Conclusion

Dedicated carry-chain logic utilizes the AX architecture to implement different arithmetic functions with minimum delay. It can improve the logic density by over 75%, and the speed by over 40%. These improvements come with minimal effort on the part of the user.

Appendix


Axcelerator Multipliers: Area
Combinatorial Cells
8000
6000
4000
2000
0
0
50
100
Ouput Width
DW Foundation (Wallace)
FC Booth-1
FC-Booth-2
FC Array
FC Parallel-2 Array


Axcelerator Multipliers: Area
Combinatorial Cells
8000
6000
4000
2000
0
0
50
100
Ouput Width
DW Foundation (Wallace)
FC Booth-1
FC-Booth-2
FC Array
FC Parallel-2 Array

http://www.actel.com

Actel Europe Ltd.
Maxfli Court, Riverside Way
Camberley, Surrey GU15 3YL
United Kingdom
Tel: +44 (0)1276 401450
Fax: +44 (0)1276 401490

Actel Corporation
955 East Arques Avenue
Sunnyvale, California 94086
USA
Tel: (408) 739-1010
Fax: (408) 739-1540

Actel Asia-Pacific
EXOS Ebisu Bldg. 4F
1-24-14 Ebisu Shibuya-ku
Tokyo 150 Japan
Tel: +81-(0)3-3445-7671
Fax: +81-(0)3-3445-7668

5192716-0/8.02