Chapter 4
Concurrent Checking for the Adders

In this chapter it will be described how the general principles of concurrent checking can be applied for the design of self-checking adders.

Adders are examples of regular circuits built up by 1-bit adder cells. The internal structure of adders is well known and functionally they can be described by simple equations for the sum and carry bits. Therefore, the general methods for concurrent checking as presented in Chapter 3 of this book can be adapted to the internal structure of the different adder types. Also the adder cells and the internal structure of the adders can be modified to ease the design of error detection circuits.

Different possibilities of adaptation of the general error detection methods to self-checking adder designs, the modification of the internal structure for the computation of the sum and carry bits and the modification of the adder cells will be demonstrated in this chapter for the different adder types.

A high error detection probability for all errors caused by single stuck-at faults and for soft errors directly induced in the registers combined with a small area overhead and a short delay for the computation of the sum bits are the challenging design goals.

The most interesting concrete results described in this chapter are as follows:

- It will be demonstrated how the general method of code-disjoint partial duplication with parity checking for the non-duplicated part can be efficiently used for the design of self-checking adders. This will be shown for carry look-ahead adders, carry skip adders and carry select adders.
- A new type of adder, the sum bit-duplicated adder, will be introduced and it will be shown how this sum bit-duplicated adder can be used for the design of self-checking carry look-ahead and carry select adders with soft error detection in the output registers.
- It will be explained that by use of this sum bit-duplicated adder, almost the same error detection probability as duplication and comparison, but with a lower area overhead can be achieved for partially duplicated adders.
- It will be shown how the already existing functionally redundant parts of carry select adders and the carry look-ahead adders which were implemented to improve
the speed of the adders without error detection can be efficiently exploited to achieve a small additional area overhead for self-checking designs.

- It will be described how the area of a self-checking carry select adder can be reduced by replacing the duplicated adder blocks for the carry-in signals 1 by simple Add1-circuits.

The best possible state-of-the-art error detection circuits for the different adder types will be presented in this chapter.

4.1 Basic Types of Adders

In this section the different types of adders without concurrent checking will be described in brief.

Ripple adders, carry look-ahead adders, carry skip adders and carry select adders will be considered.

In a carry ripple adder the delay for the computation of the most significant sum bit is high. The most significant sum bit can only be computed if the carry signals of all the preceding adder cells have already been determined. It will be shown how this delay can be significantly reduced by use of a “fast ripple adder” for which the carry signals of the adder cells are split into two different carry signals and for which the delay is the delay of a single NAND-gate per adder cell.

In a carry look-ahead adder the carry-in signals of the adder cells are computed in a special look-ahead unit. The hierarchical structure will be explained for this look-ahead unit.

In a carry skip adder the carry-in signal of an adder block may skip the block if all the propagate signals of the adder cells of this block are equal to 1. Carry skip adders with constant and variable block sizes will be presented.

It will be explained how carry select adders with duplicated adder blocks for both the carry-in signals 0 and 1 are designed and that the adder blocks for the carry-in signal 1 can be replaced by much simpler Add1-circuits.

Addition is the most frequent and most important operation in digital computers. With an $n$-bit adder two binary-encoded $n$-bit operands $a = (a_0, \ldots, a_{n-1})$ and $b = (b_0, \ldots, b_{n-1})$ are added to form the $n$-bit sum $s = (s_0, \ldots, s_{n-1})$.

The least significant bits (LSB) of the operands and the sum are $a_0, b_0$ and $s_0$ respectively. The corresponding most significant bits (MSB) are $a_{n-1}, b_{n-1}$ and $s_{n-1}$.

For an $n$-bit adder the $n + 1$-bit carry vector $c$ is denoted by $c = (c_{-1}, c_0, \ldots, c_{n-2}, c_{n-1})$ with the carry-in signal $c_{-1} = c_{in}$ and with the carry-out signal $c_{out} = c_{n-1}$.

For $i = 0, \ldots, n-1$ the $i$-th sum bit $s_i$ and the $i$-th carry bit $c_i$ are functionally determined as:

$$s_i = a_i \oplus b_i \oplus c_{i-1}$$

$$c_i = a_i b_i \lor (a_i \lor b_i)c_{i-1} = a_i b_i \lor (a_i \oplus b_i)c_{i-1}.$$