17 Multiplier

Although multipliers are usually purely combinational circuits, they are known to represent a class of hard to verify circuits. This is due to the high degree of dependability of the Boolean variables from each other (i.e. implementations have a high degree of connectivity between the gates). For this reason representations for Boolean functions like ROBDDs [Brya86] fail to provide efficient representations for multipliers, leading to problems when verifying multipliers of large bit width.

Moreover, the verification must perform a data abstraction, since the specification is given in terms of natural numbers and the implementation is based on bit vectors.

17.1 Specification

Given two natural numbers A and B (of fixed bit width). Compute the natural number P with the following property

$$P = A \cdot B \tag{17-1}$$

17.2 Implementation

17.2.1 Algorithm

The multiplication of two unsigned integers, represented as bit vectors $A = a_0 2^0 + a_1 2^1 + \ldots + a_{n-1} 2^{n-1}$ and $B = b_0 2^0 + b_1 2^1 + \ldots + b_{n-1} 2^{n-1}$ may be described by equation 17-2 and equation 17-3 in case of $n = 4$.

$$P = A \cdot B = \sum_{i=0}^{n-1} a_i 2^i B = \sum_{i=0}^{n-1} 2^i \left(\sum_{j=0}^{n-1} a_i b_j 2^j \right) \tag{17-2}$$

$$\begin{aligned} P = A \cdot B = {} & (a_0b_0)+2(a_1b_0+a_0b_1)+4(a_2b_0+a_1b_1+a_0b_2)+8(a_3b_0+a_2b_1+a_1b_2+a_0b_3) \\ & +16(a_3b_1+a_2b_2+a_1b_3)+32(a_3b_2+a_2b_3)+64(a_3b_3) \end{aligned} \tag{17-3}$$

17.2.2 Global Realization Architecture

Looking at equation 17-2, it is apparent, that the n^2 product terms (16 in equation 17-3) are all used only once. This leads to a hierarchical implementation, where first the product terms are computed, which are then fed into n^2 full adders. Finally, the resulting carry values are taken care of by a serial addition (Fig. 17-1).

In Fig. 17-1, 16 (n^2 with $n = 4$) multiplier base modules are used, which compute one product term and then perform an addition with carry. The module is specified in table 17-1 and realized as given in Fig. 17-1.

S =	$(ab) \oplus (c \oplus CIN)$
COUT =	$abc \vee abCIN \vee cCIN$
=	$((ab \wedge c) \wedge (ab \wedge CIN) \wedge \overline{(c \wedge CIN)})$

Table 17-1: Specification of the multiplier base module

17.2.3 Detailed Description

In this section, the implementation is described at a more detailed level.

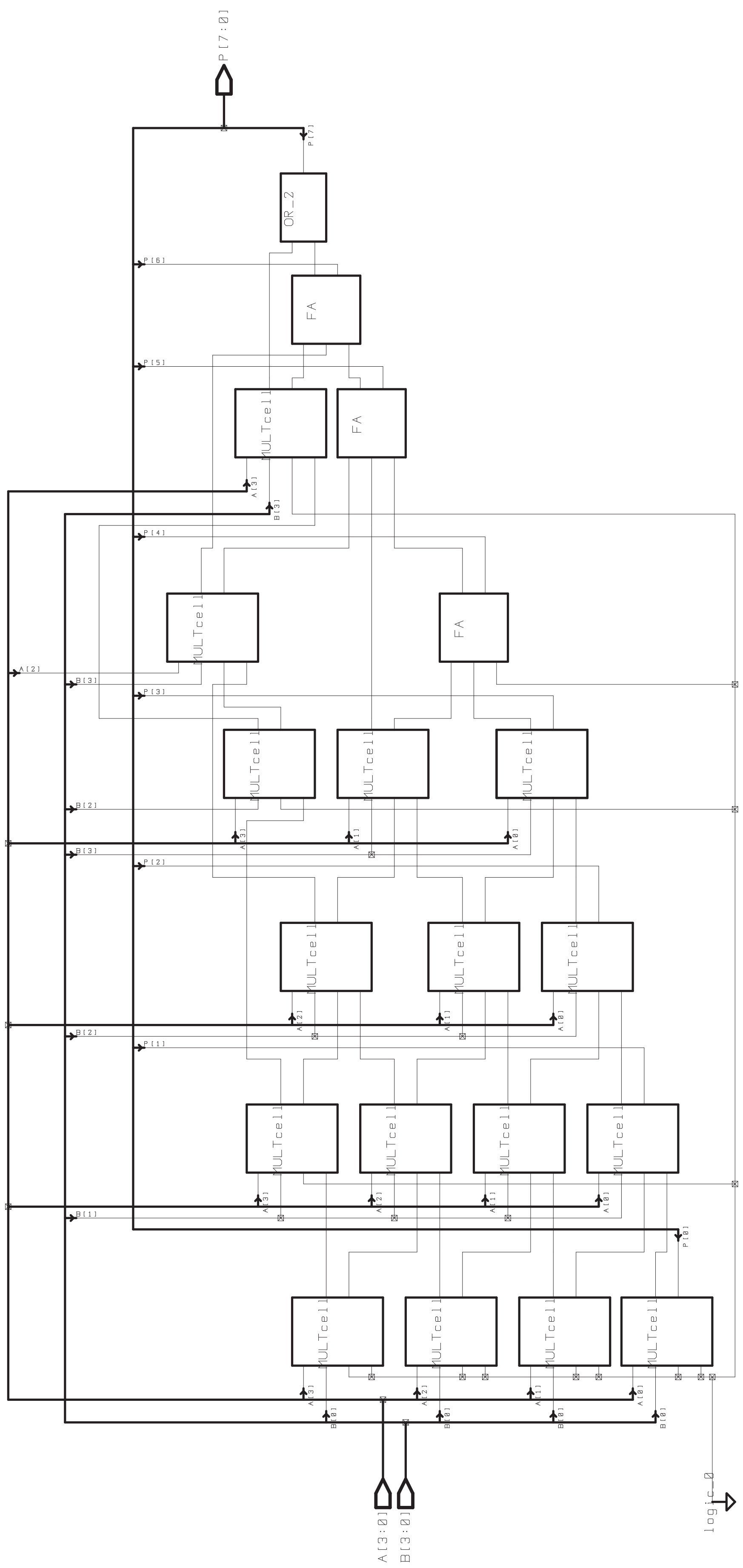

Figure 17-1: 4 Bit Multiplier

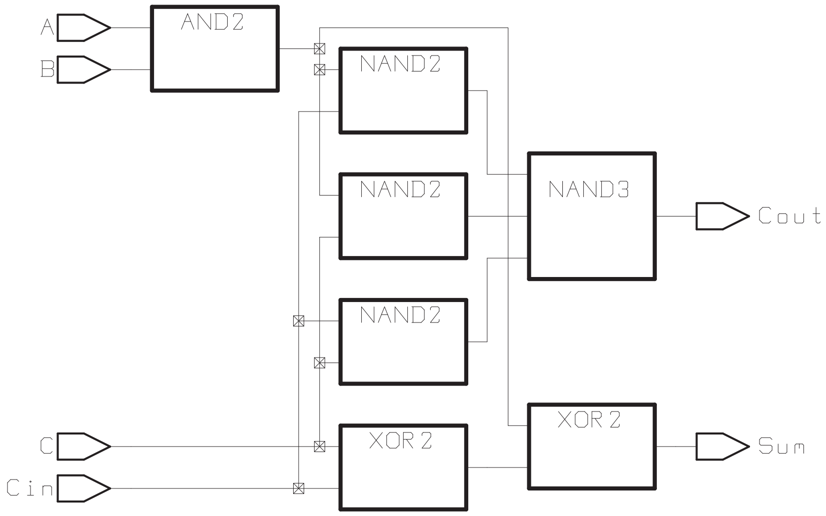

Figure 17-2: Multiplier base module (MULTcell in Fig. 17-1)

First the product terms a_0b_0 to $a_{n-1}b_0$ are computed, using the modules of figure Fig. 17-1. The inputs **c** and **CIN** are set to 0. The result of multiplying the least significant bits a_0b_0 is available at the global output of the circuit after computation.

In a second step, the product terms a_0b_1 to $a_{n-1}b_1$ are computed. The input **c** of a base cell which computes a_ib_1 $(0 \le i \le n-2)$ is set to a_ib_0 (the result of the first computation).The input **c** of the cell, which computes $a_{n-1}b_1$ is set to 0. Input **CIN** of all modules is set to the carry of the computation of $a_{i-1}b_0$. Now the computation of the second least significant bit of *P* is finished and the results, available at the outputs **S**, are needed to proceed further.

In the *n*-th step, the circuit computes the product terms a_0b_{n-1} to $a_{n-1}b_{n-1}$ using *n* multiplier base modules. The input **c** of the module which gets the most significant bits for computation (a_{n-1} and b_{n-1}) is set to 0 again. All other modules get the result of the previous step via **c**. Input **CIN** of the cell computing a_ib_{n-1} gets the carry of the cell computing $a_{i-1}b_{n-2}$. The result of the cell computing the least significant bit (a_0b_{n-1}) is available at the primary outputs of the whole circuit. The outputs **S** of all other cells is connected to inputs **b** of a full adder. Hence *n*-l full adders are needed. The second input **a** of each adder is connected to the carry out of the multipliers base module one position left (with regard to the module providing the signal for the adder input **b**). **CIN** of the rightmost adder is set to 0. All outputs **COUT** are connected to the inputs **CIN** of those full adders computing the next higher significant bit of the product *P*. The lowest row of the combinational circuit consists of a ripple-carry adder. The most significant bit of P results Or-ing **COUT** of the rightmost full adder and **COUT** of the rightmost multiplier cell placed above the adder.

A simulation can be found in Fig. 17-3.

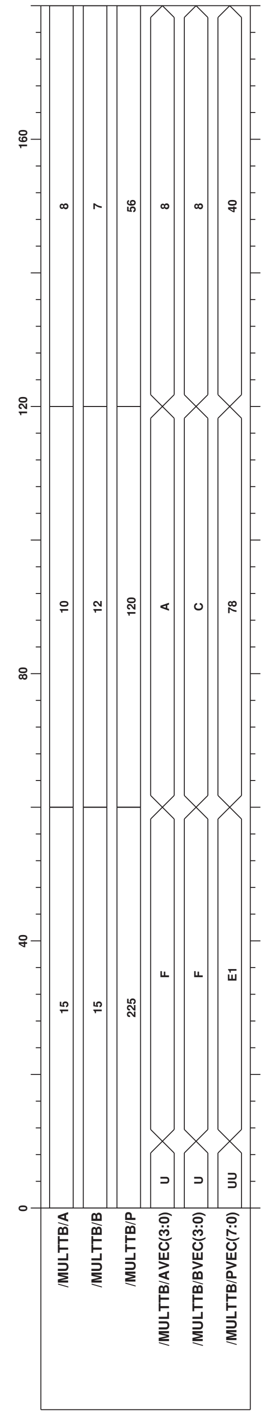

Figure 17-3: Simulation of the Multiplier

17.3 Status and Acknowledgments

Thanks to H.-P. Eich and C.-J. Thomas for designing the circuit in a commercial design system.

17.4 Literature

[Brya86] R. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.