18 Divider

A divider is mainly introduced for having an additional purely combinational example besides a multiplier and because it is used in other verification examples like the GCD.

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.

18.1 Specification

Given two natural numbers, the dividend N and the divisor D, calculate the quotient Q and the rest R such that Q is the largest natural number and R is a number with $R < D$ such that

$$N = D \cdot Q + R \tag{18-1}$$

18.2 Implementation

18.2.1 Algorithm

The implementation closely follows the “restoring cellular array divider”, presented in [Hwan79] (pp. 264ff., see also [Haye88]). As the multiplier, presented in section 17, it is based on a regular, cell based design.

In contrast to other implementations, this design performs two operations at each computation cycle. The first subtracts the divisor for the dividend, staring with the most significant bit. If the subtraction results in a negative difference, a bit 0 will be produced for the quotient. The second operation is a restoring addition, if a negative difference occurred. The latter operation produces parts of the rest R. Each computation cycle is finished by a left-shift. If a positive result was achieved by the substraction, then a 1 bit is generated for the quotient and no addition is performed.

18.2.2 Global Realization Architecture

The circuit is built by a iterative cellular array as depicted in Fig. 18-1. Each cell DIVcell uses a controlled subtraction with 4 inputs and outputs.

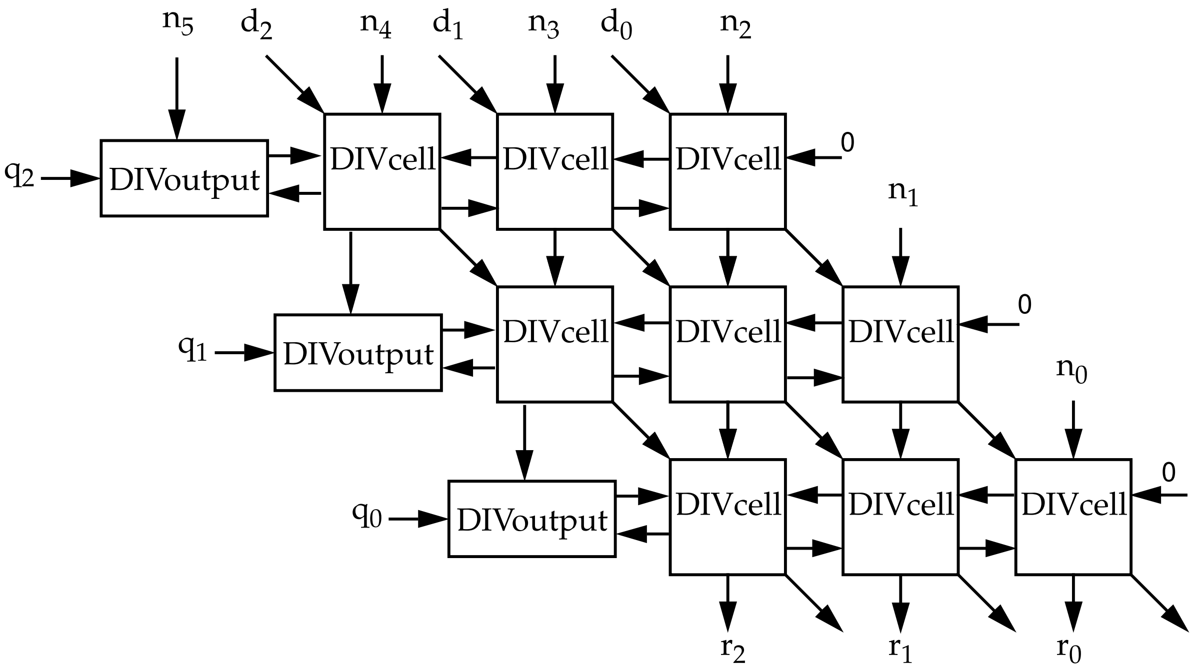

Figure 18-1: Divider

The signals of one DIVcell module are given in Fig. 18-2 and are specified in table 18-1. The signals **DivOut** and **SubOut** are omitted in the table, since they are identical to **DivIn** and **SubIn**, respectively.

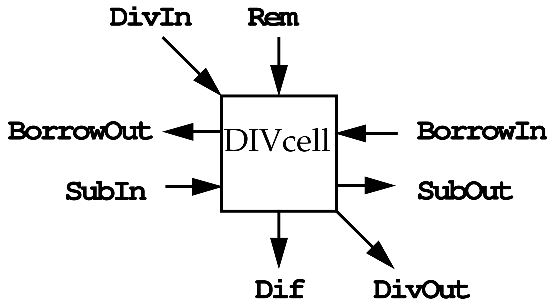

Figure 18-2: Signals of the divider base module DIVcell

BorrowOut =	(**Rem** **DivIn**) ∨ (**Rem** **BorrowIn**) ∨ (**DivIn** **BorrowIn**)
Dif =	(**Rem** **SubIn**) ∨ (**Rem** **DivIn** **BorrowIn**) ∨ (**Rem** **DivIn** **BorrowIn**)
	∨
	(**Rem** **DivIn** **BorrowIn** **SubIn**) ∨ (**Rem** **DivIn** **BorrowIn** **SubIn**)

Table 18-1: Specification of the divider base module

The signale of a DIVoutput cell are given in Fig. 18-3 and are specified in table 18-2.

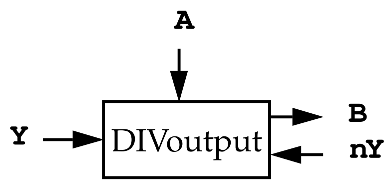

Figure 18-3: Signals of the divider base module DIVcell

Y =	**A** ∨ ¬ **B**
nY =	¬ **Y**

Table 18-2: Specification of the divider interface module DIVoutput

The inputs **Rem**, **DivIn** and **BorrowIn** are used for rest, divisor and carry input. **BorrowOut** denotes the carry output and **SubIn** is a control signal for all cells of the respective row. The signal **Dif** for the computed rest output realizes a function as shown in table 18-1.

Dif =	**Rem** ⊕ **DivIn** ⊕ **BorrowIn**, if **SubIn** = 0
Dif =	**Rem**, if **SubIn** = 1

Table 18-3: Specification of the rest computation

The realization of a cell DIVcell is given in Fig. 18-4.

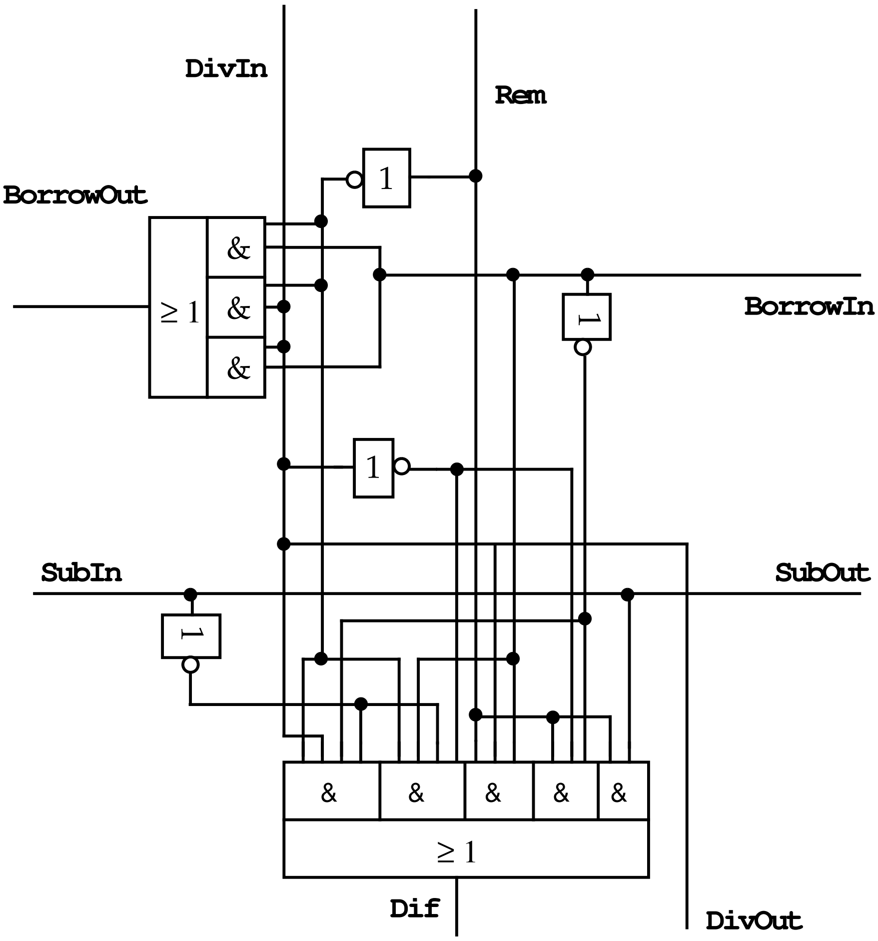

Figure 18-4: Divider base module

18.3 Detailed Implementation Description

For building a divider with an m-bit dividend and a n-bit we need $(m-n)n$ cells. The $(m-n)$ rows of the array are shifted one position right to each other and consist of n cells each. In Fig. 18-1 there are shown 4 unsigned bit vectors: the dividend $N = \{n_5, \ldots, n_0\}$, the divisor $D = \{d_2, \ldots, d_0\}$, the quotient $Q = \{q_2, \ldots, q_0\}$ and the rest $R = \{r_2, \ldots, r_0\}$.

The input **DivIn** (divisor bit in) of the first row the appropriate bit of the divisor is applied, where the leftmost cell gets the most significant bit. Input **BorrowIn** (borrow in) of the i-th cell is connected to the output **BorrowOut** (borrow out) of the (i-1)-th cell $(0 < i \le n-1)$. Input **BorrowIn** of cell 0 is set to 0. Output **BorrowOut** of cell n-1 is connected to an inverter. The inverted signal is “ORed” with most significant bit of the dividend. It is available as a global output and defines the MSB of the quotient. This signal is also inverted again and fed to input **SubIn** (subtract control in) of the (n-1)-th cell. Output **SubOut** (subtract control out) of the i-th cell is connected to **SubIn** of the (i-1)-th cell. Output SubOut of the 0-th cell remains unconnected. All outputs **DivOut** (divisor bit out) of the cells at a particular row are connected to the inputs DivIn (divisor bit in) of the next lower row, according to their indices.

Output **Dif** (difference) of the (n-1)-th row is OR-connected with the inverted **BorrowOut** output of the (n-1)-th cell of the next lower row. All outputs **Dif** of the cells n-2 to 0 are connected to the inputs **Rem (**remainder bit) of the cells n-1 to 1of the next lower rows. The inputs of **Rem** of row 0 get the dividend (MSB left side).

Besides input **DivIn** and **Rem**, all rows are connected as described above for the first row. The j-th row gets a signal via **DivIn** from output **DivOut** of the (j-1)-th row and via **Rem** a signal from the output **Dif** $(0 < j \leq m - n)$. Input **Rem** of the 0-th cell of row j gets the j+1-highest bit of the dividend. Via the outputs **Dif** of the lowest row the rest of the division is available.

Moreover, every row provides a bit of the quotient, according to its row index j and together with the OR-connected and inverted signal of **BorrowOut** of the (n-1)-th cell.

The implementation of a base module DIVcell, the DIVoutput cell and a design using a 6 bit dividend and 3 bit divisor is shown in Fig. 18-5, Fig. 18-6 and Fig. 18-7, respectively.

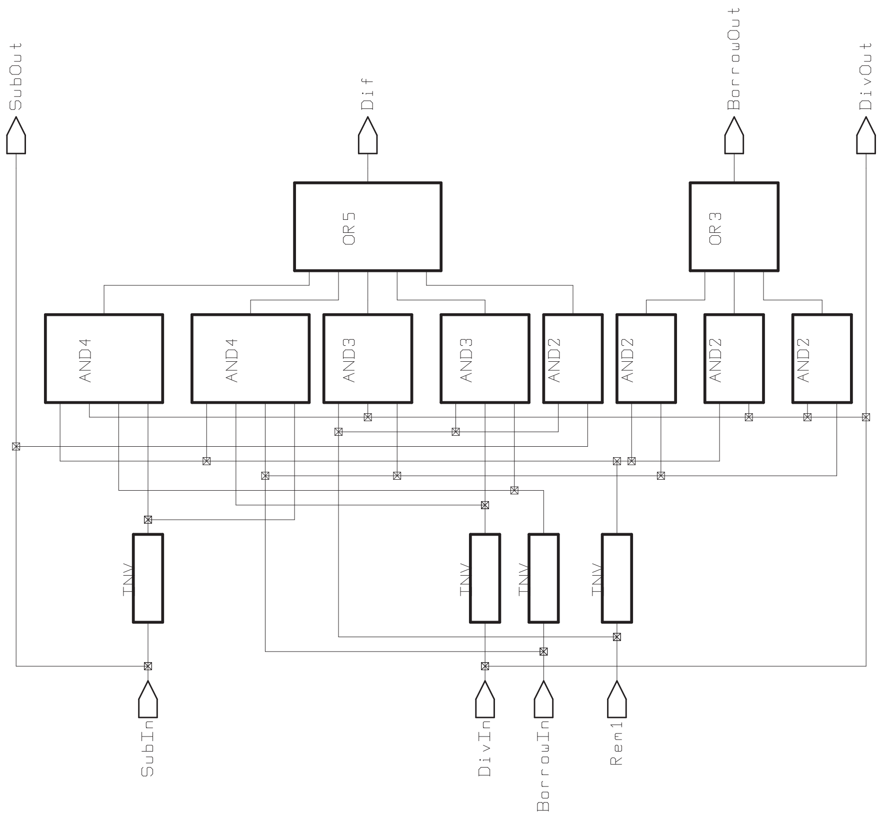

Figure 18-5: Divider base module designed in SYNOPSYS

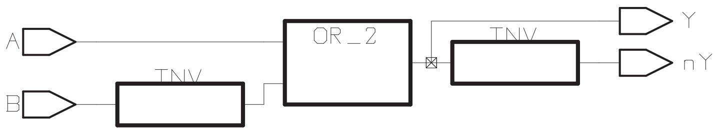

Figure 18-6: Output Cell DIV output

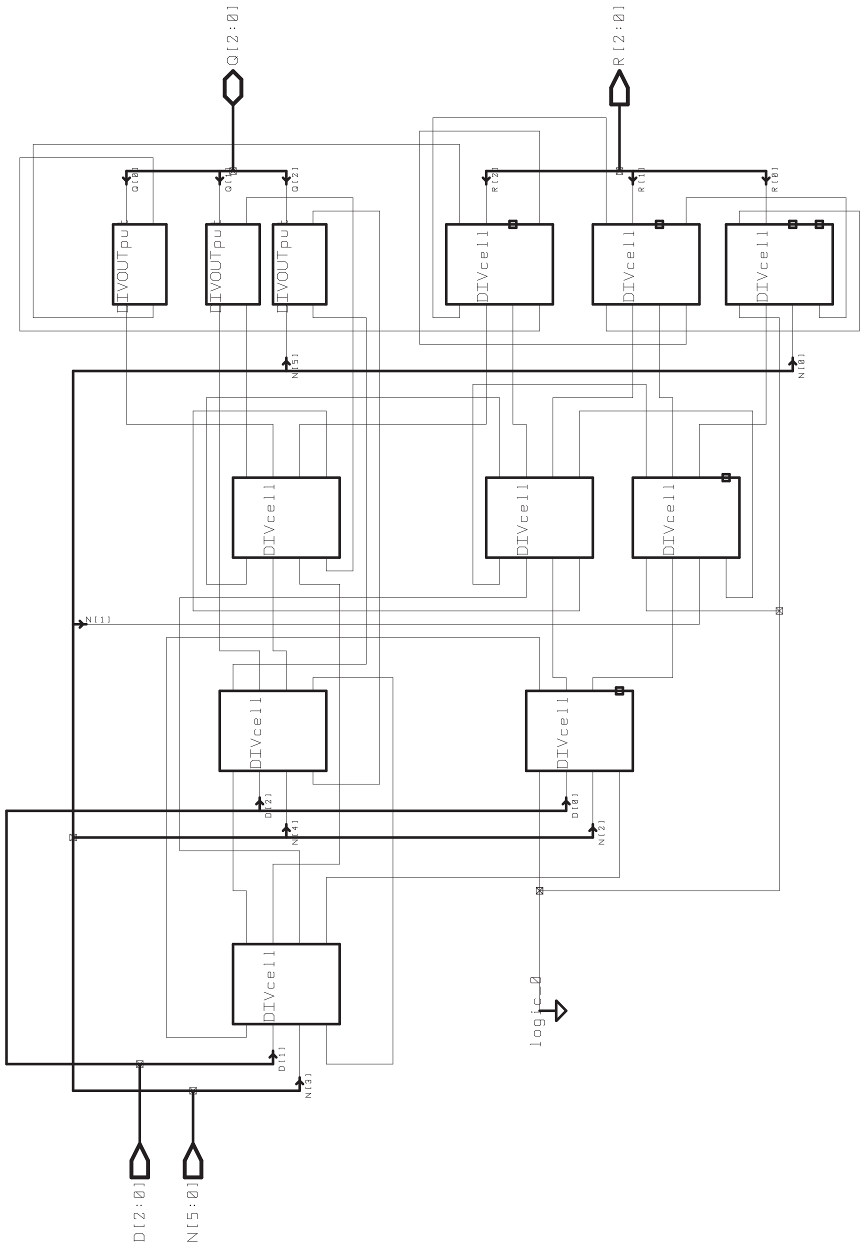

Figure 18-7: Divider designed in SYNOPSYS

Fig. 18-8 shows the simulation waveform..

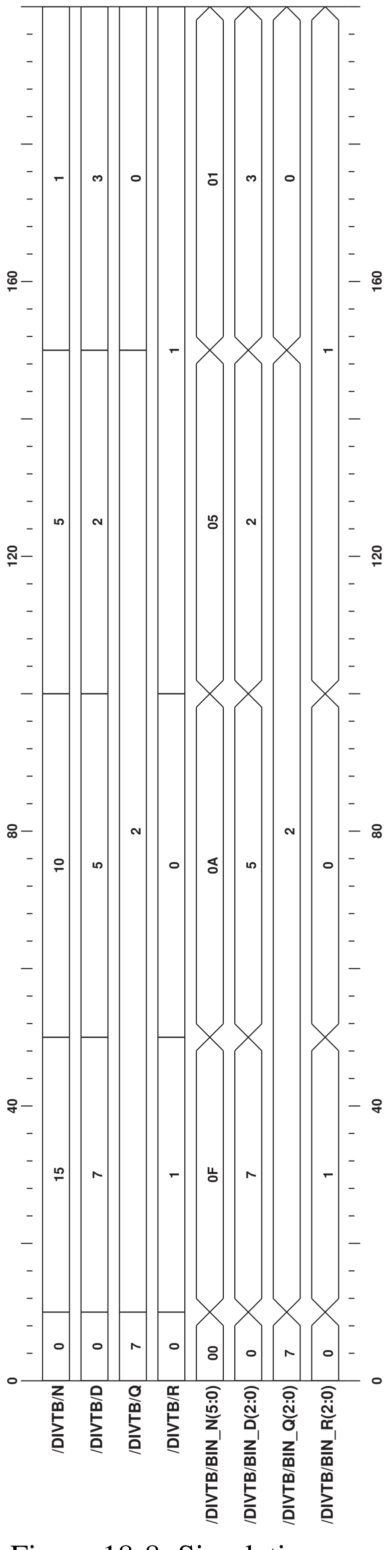

Figure 18-8: Simulation waveform

18.4 Status and Acknowledgments

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

Thanks to G. Janssen from Eindhoven University for pointing out numerous flaws in the first version of this example.

18.5 Referencee

[Fram92] CADENCE Design Framework II version 4.2a. Reference Manual, February 1992.