21 1Syst (Filter)

21.1 Introduction

The filter circuit of this section and the matrix multiplier of section 22 have been chosen as witnesses of one-dimensional and two-dimensional systolic architectures. The notion of “systolic arrays” has first been introduced by Kung and Leierson [KuLe78]. Basics on these specialized regular architectures can be found in [Kung82]. The circuit to be presented here is a filter element also taken from [Kung82].

Systolic arrays require additional efforts for specifying the intended behavior, since here data have to be applied at different times in a special order to achieve a correct functioning of the circuit. Moreover, more-dimensional systolic arrays are a good illustration of more-dimensional generic circuits.

21.2 Specification

The circuit processes a stream of input values $\{x_1, x_2, \ldots, x_n\}$, $x_i \in N$, $1 \le i \le n$. These input values have to be multiplied with a list of weights $\{w_1, w_2, \ldots, w_k\}$, $w_j \in N$, $1 \le j \le k$. Each y_i of the resulting output stream $\{y_1, y_2, \ldots, y_{n+1-k}\}$ is computed as in equation 21-1.

$$y_i = w_1 x_i + w_2 x_{i+1} + \ldots + w_k x_{i+k-1} \tag{21-1}$$

An example computation for $k = 3$ and $n = 5$ is given in equation 21-2.

$$\begin{aligned} y_1 &= w_1 x_1 + w_2 x_2 + w_3 x_3 \\ y_2 &= w_1 x_2 + w_2 x_3 + w_3 x_4 \\ y_3 &= w_1 x_3 + w_2 x_4 + w_3 x_5 \end{aligned} \tag{21-2}$$

In the implementation given below, first the weights w_i are fed serially into the input **`StreamIn`**. After k weights have reached their respective position in the k cells, they are stored by setting **`StoreWeight`** to true. Afterwards the input values are fed serially into the **`StreamIn`**. The values y_i are computed and serially shifted out on output **`ResultOut`**.

This specification can be verified for concrete values of k weights or as a generic circuit.

21.3 Implementation

21.3.1 General architecture

The implementation is realized using overlapping additions and multiplications. In the first cycle t_1 the input value is multiplied with the weight stored in the respective stage. In the second cycle t_2 this product is to be added to the now available intermediate result y_i. In order to achieve this, the input values x_i must be separated by two clock ticks (Fig. 21-1).

During an initialization phase the weights w_j are loaded into the stages by using the first k input values of the input stream as weights. In the following, inputs values are treated as *n* bit integers and output values as *m* bit integers (Fig. 21-2).

Let w a weight and z an internal variable. If there is a rising edge at $t1$, then $x^{t+1} = x^t$ and $z = x^t \cdot w$ is computed. If there is a rising edge at $t2$, then $y^{t+1} = z + y^t$ is computed.

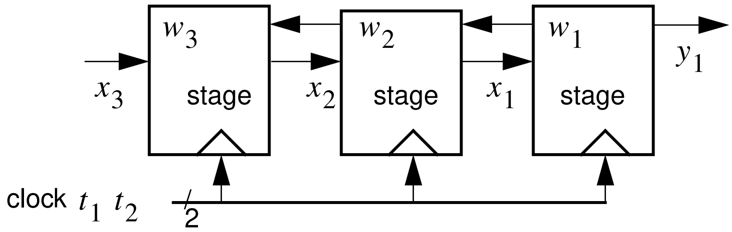

Figure 21-1: Basic array structure computing the first result

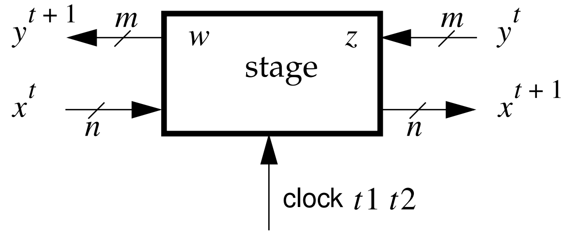

Figure 21-2: Black box view of one stage

21.3.2 Implementation of one stage

The stage has two global inputs **`StreamIn<n-1:0>`** and **`ResultIn<m-1:0>`.** Input **`StreamIn<n-1:0>`** is fed via a 1:2 n bit demultiplexer into two n bit registers. Using the control signal **`SelectWgtStr`** of the demultiplexer either the register for storing the weight (**`SelectWgtStr`** = 0) or the register for the input value (**`SelectWgtStr`** = 1) is selected. Both register store their respective inputs, if **`StoreWgt`** = 1 (**`StoreStr`** = 1) and there is a rising edge at t1.

The content of the input register is available at the output **`EStreamOut<n-1:0>`** and the input of an n bit multiplier. The multiplier also gets the weight.[1] The $2n$ bit output of the multiplexer is stored in a register if **`StoreRes`** = 1 and there is a rising edge at clock t2 (Fig. 21-3).

Intermediate results, available at input **`ResultIn<m-1:0>`** are stored in a m bit register if **`StoreRes`** = 1 and a rising edge at t2. The content of this register as well as the multiplication results are added via an adder[2] to get the result at **`ResultOut<x:1>`** with $x \geq max(n, m) + 1$.

The general clocking scheme must be such that the rising edge of t1 occurs before the rising edge of t2.

The realization of a stage in a commercial design system is given in Fig. 21-3 and Fig. 21-4.

Fig. 21-5 shows an example computation with $w1 = 4$, $w2 = 2$, $w3 = 1$ and $x1 = 9$, $x2 = 8$, $x3 = 7$, $x4 = 6$ and $x5 = 5$, respectively. The output of a behavioral VHDL description (the "specification") produces the output stream $59, 52, 45$, the structural VHDL description (the implementation) produces $3B, 34, 2D$ (hexadecimal).

21.4 Status and Acknowledgments

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

21.5 Literature

[Kung82] H.T. Kung. Why systolic architectures. *IEEE Computer*, pages 37–46, January

1. The multiplier is not further specified here since its realization is arbitrary. Is is possible to use e.g. the multiplier of section 17.
2. For the adder the same holds as for the multiplier.

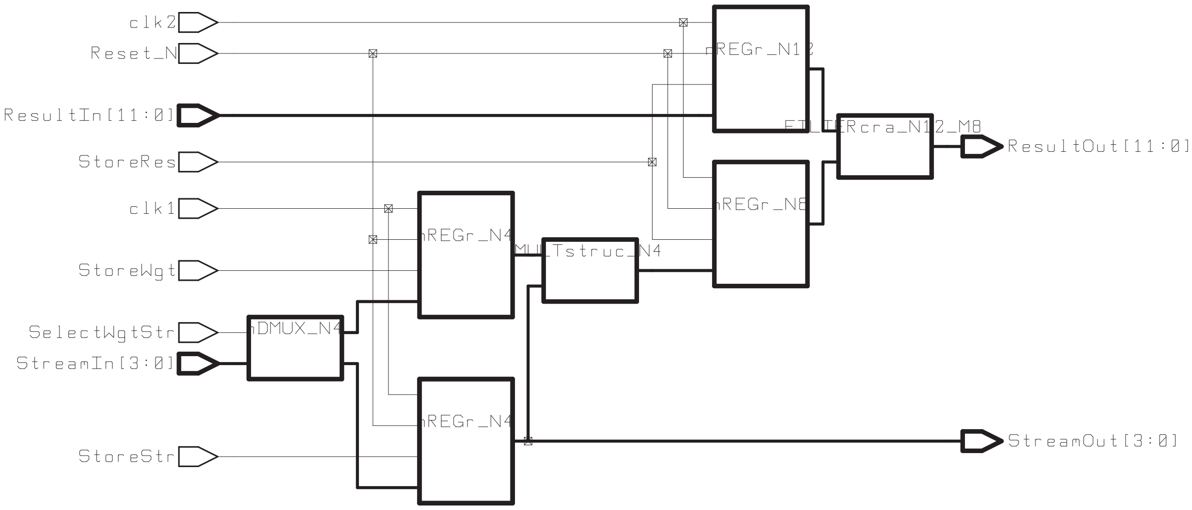

Figure 21-3: SYNOPSIS realization of one stage

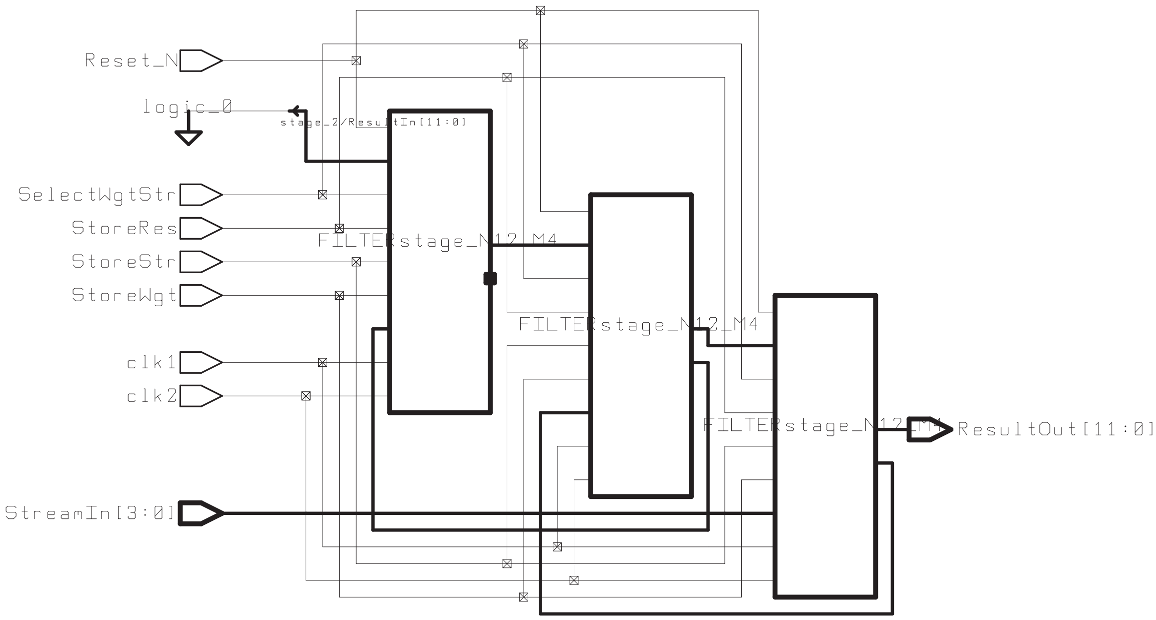

Figure 21-4: SYNOPSIS realization of a complete structure

1982.

[KuLe78] H.T. Kung and C.E. Leierson. Systolic arrays (for VLSI). In *Sparse Matrics Proceedings*, pages 256–282. Society for Industrial and Applied Mathematics 1979, 1978.

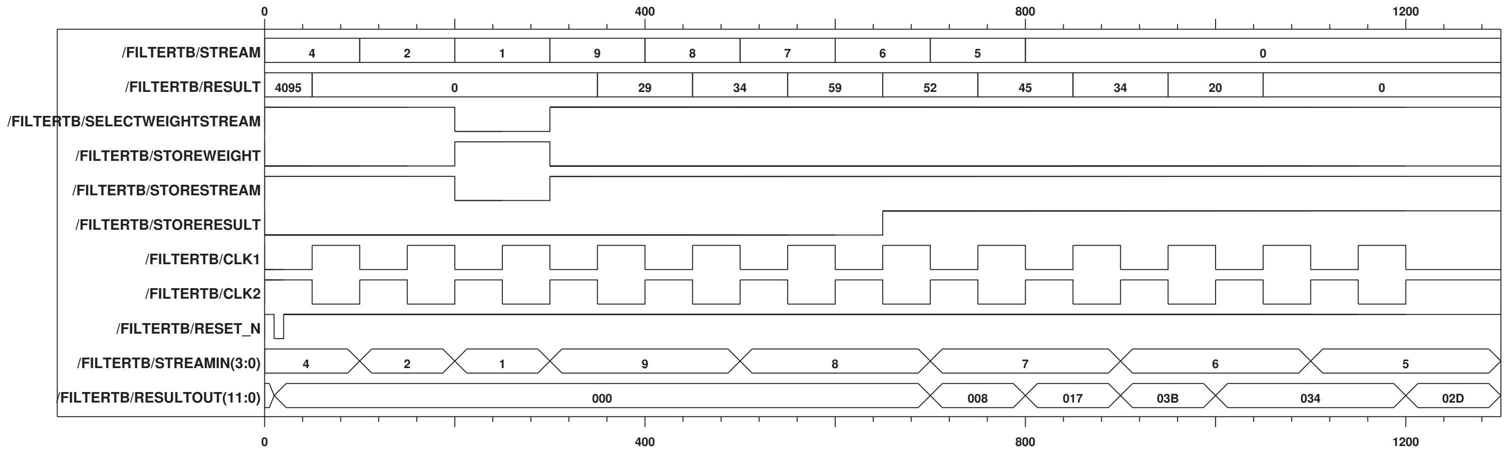

Figure 21-5: Detailed timing diagram