

## FAST PIPELINED COMPLEX CONVOLVER

L. Breveglieri and V. Piuri

*Indexing terms:* Systolic arrays, Convolution, Digital signal processing

A fast architecture for fully-pipelined convolution of complex numbers is presented. High regularity and modularity allow for a compact VLSI implementation. High throughput efficiently supports massive computation.

*Introduction:* Massive computation of the complex convolution  $Y_k = \sum_{i=1}^N W_i X_{k+i}$  (in correlation form) is required in many applications for digital signal and image processing. Efficient and compact architectures must be designed to support real-time applications by means of dedicated VLSI/WSI implementations.

We present a new pipelined structure in which input data are serially supplied to a semisystolic array of bit slices. Multiplications are performed in a serial-parallel scheme, while addition of the convolution terms  $W_i X_{k+i}$  is embedded in the addition of the partial products. Outputs are serially generated. Our approach is based on the multiplier structure proposed in Reference 1. Even though serial operations are considered to be slower than parallel operations, they are characterised by a reduced number of input and output pins, a simplified internal interconnection structure, hence a high clock rate, a high throughput, a reduced silicon area and ease of testability. All these features are very advantageous for VLSI integration. Overlapping and compression of multiplications and additions allow optimisation of time and silicon area.

The real and the imaginary parts of the operands are in full fractional two's complement LSB-first representation;  $n$  bits are used for each part (real and imaginary) of the input operands and of the convolution weights, and  $2n$  bits are used for the output  $Y_k = \sum_{i=1}^N W_i X_{k+i}$  of the convolver. We assume that the number  $N \geq 1$  of convolution terms never induces overflow. Weights are fixed during convolution, but they may be loaded off-line before starting the nominal computation.

*Multiplier structure:* The multiplier is the basic building block of the convolver. The product of two complex numbers ( $w = a + ib$ ,  $x = c + id$ ) can be reduced to four real products and two sums/subtractions:  $Z = w * x = (a + ib)(c + id) = (ac - bd) + i(ad + bc) = Z^{re} + iZ^{im}$ . The real and imaginary parts of each weight  $W$  are serially stored off-line in the weight registers. Two separate input lines are used to load these shift registers. The real and the imaginary part of each sample  $X$  are supplied to the multiplier simultaneously on separate input lines. The real and imaginary parts of the result  $Y$  are output serially from two separate output lines.

The multiplication is performed by using the serial/parallel to serial algorithm, because it is the simplest and most efficient. Serial to parallel conversion of the weights is performed by the weight registers, without additional costs. The real part  $Z^r$  is computed by subtracting the two real products  $ac$  and  $bd$ ;  $ac$  is obtained by shifting and adding the rows  $a_{n-i}c2^{-(1+i)}$  ( $0 \leq i \leq n-1$ ) of the partial product real matrix  $ac$  (similarly for  $bd$ ). Instead of computing the partial-product matrices  $ac$  and  $bd$  separately, by first adding their rows and then subtracting the two products  $ac$  and  $bd$ , we merge the summation of the rows with the subtraction of the products. The operation performed is:  $(a_{n-i}c - b_{n-i}d)2^{-(1+i)}$  ( $0 \leq i \leq n-1$ ). The same can be carried out for the imaginary part  $Z^i$  (in this case, all operations are additions). By compressing subtractions and sums into a single, though more complex, operation, high performance can be achieved.

The multiplier structure is shown in Fig. 1a, and Fig. 1b gives the details of the bit slices. When the operand  $c + id$  is

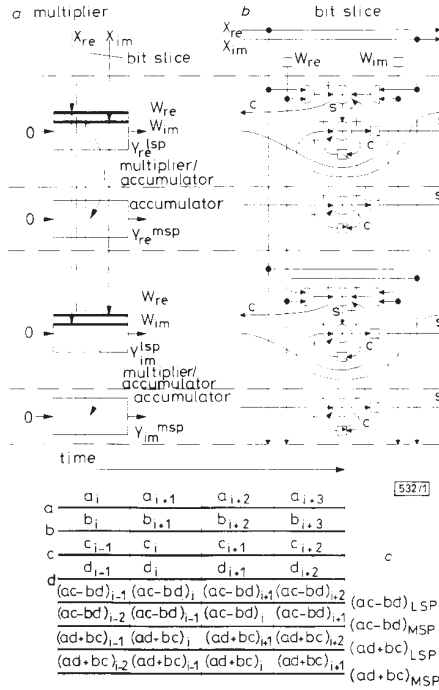


Fig. 1 Complex multiplier, bit slice and time diagram

- a Multiplier
- b Bit slice
- c Time diagram

serially input, the product sections generate the partial product matrices  $ac$ ,  $bd$ ,  $ad$  and  $bc$  (one row for each bit of the operands).  $Z^r$  and  $Z^i$  are computed by the two pipelined multiplication/accumulation sections; their least significant parts (LSPs) are computed and output while the operand  $c + id$  is input. The content of each multiplier/accumulator (representing the most significant part of the result, MSP) is then transferred in parallel into the corresponding shift register of the accumulation sections; there, MSPs are shifted and added by serial adders placed at the end of the pipeline, during the subsequent  $n$  clock periods; the product and the multiplication/accumulation sections are available to perform the next multiplication. Fig. 1c gives the complete time diagram of data flow; output delivery and input acquisition are overlapped without idle clock cycles between the operands, i.e. full external pipelining is obtained.

**Convolver structure:** Convolution involves multiplications and cascaded summations. These summations are similar to the summations of partial products, which are performed inside

multiplications. We can view each summation of the convolution as a further summation in the multiplication and, thus, we can compact these two operations into a single operation.

The convolver is obtained by cascading  $N$  stages of complex multipliers, as shown in Fig. 2a. If all registers

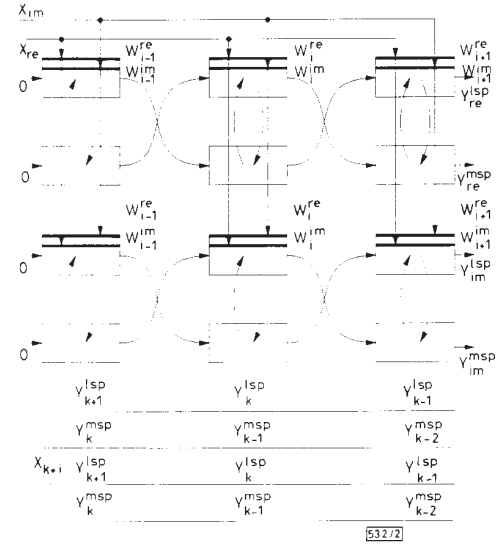


Fig. 2 Complex convolver and time diagram

- a Convolver
- b Time diagram

holding the real parts of the weights are cascaded, only one input line is required to load the weights in the whole convolver (similarly for the imaginary parts). The outputs of the multiplication/accumulation sections of stage  $i$  are connected to the inputs of the corresponding accumulation sections of stage  $i+1$ . The outputs of the accumulation sections of stage  $i$  are connected to the inputs of the corresponding multiplication/accumulation sections of stage  $i+1$ . During the first  $n$  clock cycles, complex products are computed as discussed for the multiplier. LSPs, generated in stage  $i$  for the  $k$ th convolution, are serially stored in the shift registers of the accumulation sections of stage  $i+1$ . Data contained in the multiplication/accumulation sections and in the accumulation sections of stage  $i$  are then exchanged in parallel. In stage  $i$ , this operation transfers the sums and the carries, representing the MSPs, contained in the multiplication/accumulation section, into the accumulation section of the same stage. At the same time, LSPs, generated in stage  $i-1$  and stored in the accumulation sections of stage  $i$ , are used to initialise in parallel all carries in the multiplication/accumulation sections of stage  $i$  for the subsequent multiplication, concerning the  $(k+1)$ -th convolution; at each product cycle, such initialisation implies an automatic addition of the LSP of the product  $W_i X_{k+i-1}$  to the LSP of the product  $W_i X_{k+i}$ , in the first  $n$  clock cycles considered above. During the subsequent  $n$  clock cycles, MSPs are serially generated in the accumulation section of stage  $i$  and are delivered to the multiplication/accumulation section of stage  $i+1$ . In stage  $i$ , the MSPs coming from stage  $i-1$  provide the serial input carries during the generation of the LSPs, i.e. guarantee an automatic addition of the MSP of product  $W_{i-1} X_{k+i-1}$  to the MSP of product  $W_i X_{k+i}$ . In other words, the complex addition contained in the convolution is implicitly performed by properly initialising the shift registers of the multiplication/accumulation sections. The complete time diagram of data flow is given in Fig. 2b.

The condition on  $N$  avoiding overflow can be removed by modifying the convolver architecture. If  $N$  allows values of  $Y$  to be generated which can be represented in real and imaginary parts of  $(H+1)n$  bits each, we use  $H \geq 1$  accumulation sections in each stage instead of only one for the real

(imaginary) parts. The multiplication/accumulation section of stage  $i$  is connected to the first accumulation section of stage  $i + 1$ ; in turn, the  $h$ th accumulation section of stage  $i$  ( $h \in [1, H]$ ) is connected to the accumulation section  $h + 1$  in stage  $i + 1$ . The  $H$ th accumulation section in stage  $i$  is connected to the multiplication/accumulation section of stage  $i + 1$ . Data exchange after the first  $n$  clock cycles consists in a parallel data rotation in each stage.

*Architectural evaluations:* The output latency of each multiplication stage is zero, because the first bit of the result is delivered immediately after presentation of the first bit of the input. The output latency of the whole convolver is  $(N - 1)n$  clock cycles. The clock period is upper bounded only by the propagation delay of a single full adder. Full pipelining is achieved, so that no wait cycles are required between consecutive input samples.

The structure of the multiplier is easily testable [2]: thus, also for the complete convolver it is possible to achieve a complete fault coverage, under the single stuck-at fault model, in linear time with respect to the number of bit slices. No additional gate and/or signal is required for testing.

30th September 1992

L. Breveglieri and V. Piuri (*Politecnico di Milano, piazza L. da Vinci 32, Milano, I20133 Milano, Italy*)

## References

- 1 BREVEGLIERI, L., PIURI, V., and SCIUTO, D.: 'Fast pipelined multipliers for bit-serial complex number'. IEEE Proc. COMPEURO '91, 1991
- 2 BREVEGLIERI, L., PIURI, V., and SCIUTO, D.: 'A fast pipelined complex multiplier: the fault-tolerance issues'. IEEE Proc. Workshop on Defect and Fault-Tolerance '92, 1992
- 3 BREVEGLIERI, L., and DADDA, L.: 'A serial-input bit-sliced convolver'. Proc. ICCD '88, 1988