DESIGN AND FPGA IMPLEMENTATION OF DIGIT-SERIAL FIR FILTERS

D.S. Dawoud* and S. Masupa**
**Research Centre for Radio Access Technologies, School of Electrical, Electronic and Computer engineering, University of KwaZulu Natal, King George V avenue, Durban, 4041, South Africa*
***University of Botswana, Faculty of Engineering, P.Bag 0022 , Gaborone, Botswana*

Abstract: In this paper the design of a digital-serial N-tap FIR filter with programmable coefficients is presented. The design considers the general case of W-bit sample word and m-bit coefficient word. The processing of the data within the filter takes place with full precision. The output data is truncated to W bits. The design introduced the new digit-serial multiplier that guarantees minimum processing time and reduces the hardware requirements. Sign-amplitude representation for the coefficients and two's complement for the input samples simplified the circuit configuration and allows the use of a single common two's complement circuit for all the filter sections. A 100-tap, 8-bit word length version filter is implemented using an ALTERA FPGA device. The filter can be used in real-time processing with sample rates ranging from 7.5 to 22 MHz

Key words: FIR filter, canonical and transposed forms, digit-serial multiplier, FPGA, Multiply-accumulate unit.

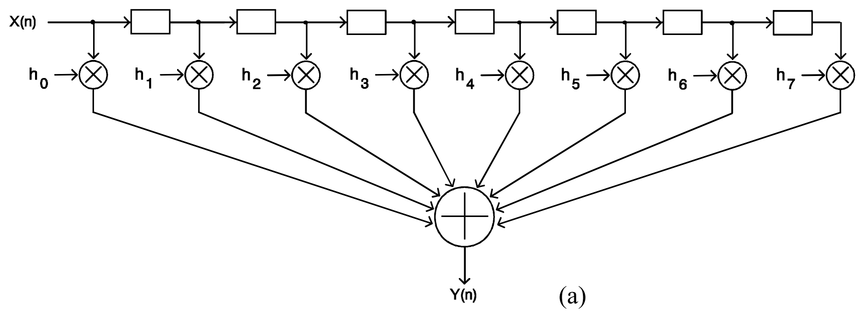
1. INTRODUCTION

Bit parallel designs process all of the bits of an input simultaneously at a significant hardware cost. In contrast, a bit serial structure processes the input one bit at a time, generally using the results of the operations on the first bits to influence the processing of subsequent bits. The advantage enjoyed by the bit serial design is that all of the bits pass through the same logic, resulting in a huge reduction in the required hardware. Typically, the bit serial approach requires 1/mth of the hardware required for the equivalent m-bit parallel design.

The price of this logic reduction is that the serial hardware takes m clock cycles to execute, while the equivalent parallel structure executes in one. The time-hardware product, however, for the serial structure is often smaller than for equivalent parallel designs because the logic delays between registers are generally significantly smaller. This means that the serial machine can operate at a higher clock frequency. In the case of FPGAs, signal routing contributes significant propagation delays and often uses up logic cells. The serial structures tend to have very localized routing, often to only one destination. In contrast, the parallel machines usually need signals extended across the width of the processing element. The limited and slow routing resources in FPGAs make the serial processing elements even more attractive. In some cases, the overall throughput for a serial design implemented in an FPGA can actually exceed that of an equivalent parallel design in the same device.

In the case of DSP, the output sample is the sum of a number of terms while the term itself represents a multiplication of W-bit sample word and m-bit coefficient word. Accordingly, the processing of the data in case of DSP has two levels; the first is on the level of the calculation of the term (multiplication process) and the second level is on the system level to get the output by accumulating the outputs of the first level. The fully serial implementation uses one serial-bit multiplier to calculate the terms in serial form and one accumulator to accumulate the results. Such an implementation guarantees the minimum amount of hardware required but at the same time produces a system that cannot be used with any real-time application. On the other hand, using a parallel multiplier for each term in the expression of the filter together with a parallel adding network to get the output, results in a tremendous amount of hardware and very fast. Such high speeds are often not required in many applications. Many of the researches in the field of DSP concentrate on finding algorithms and implementation techniques that result in real-time systems with reasonable hardware complexity that can be implemented using FPGA devices.

In [2] and [3] the main architectural ideas are reviewed. The powers-of-two coefficients are exploited by Evans in [4]. The bit-serial approach is studied in [5], [6]. In [7] and [8], the canonic signed digit arithmetic is applied with the bit-serial style of implementation. [9] constructs several circuits based on the digit-serial systolic multiplier. In [10], [11], [13], [14], [15], [16], the distributed arithmetic is introduced. Finally, a comparison between



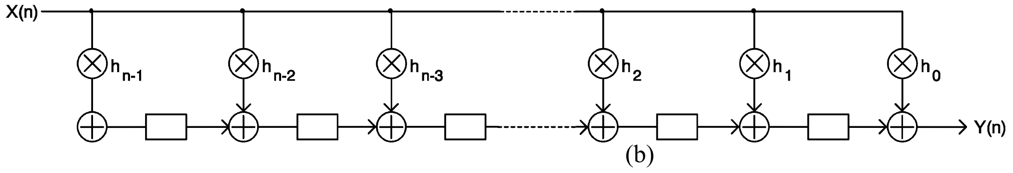


Figure 1 FIR filter Structures: (a) canonical form; (b) inverted form

serial and parallel approaches is presented in [1]. From the implementation side, references [2], [3], [4], [7], [8], [9], [10], [11], [12], [13], [14] use Xilinx chips; [15] and [16] Altera devices; [1] both Xilinx and Altera; [4] both Orca and Xilinx; and [5] uses CLI.

The proposed structure calculates each term in serial form using multiply-accumulate (MAC) block, and uses N MAC blocks (N is the number of taps in the filter) working in parallel to get the final output sample. In serial processing, to keep full precision along the whole datapath during the multiplication $H_i.X_i$ and the accumulation of these values, the system needs registers of word length ($W_{\text{in-data}} + m_{\text{coefficient}} + \log_2 N$) and the same number of cycles is required to complete the multiply/accumulate operation. The proposed MAC hardware reduces the number of cycles to half this value.

The organization of this paper is as follows. In the next section the FIR filter architecture, the different canonical and inverted form topologies to design FIR filters are summarized. The expression and the topology that we are going to consider in implementation are explained. Section 3 covers all the design and implementation aspects. The digit-serial architectures are briefly exposed, the basic digit-serial adder cell is explained, the proposed digit-serial multiplier is also given and finally the input/output/control block is given. Section 4 deals with the FPGA implementation while in section 5 some conclusions and results are given.

2. FIR STRUCTURE

The N-tap FIR digital filter is normally described by the equation:

$$y(n) = \sum_{i=0}^{N-1} H_i X_{n-i} \tag{1}$$

In our application, the coefficient H_i has m bits width and will be represented in sign-amplitude form. The input sample has a word length W-bit and is represented in 2's complement form:

$$X_n = -2^{W-1} x_{W-1} + \sum_{i=0}^{W-2} x_i 2^{-i} \tag{2}$$

The filter equation, accordingly, will take the form:

$$y(n) = \sum_{i=0}^{N-1} (-H_i x_{n-i,W-1} 2^{W-1}) + \sum_{i=0}^{N-1} \sum_{j=0}^{W-2} H_i x_{n-i,j} 2^{-j} \tag{3}$$

The transposed forms (symmetrical and nonsymmetrical) are considered in this paper.

Using sign-amplitude representation for the coefficients and two's complement form for the input samples results in the following simple multiplication algorithm:

i. If the coefficient H_i is positive: multiply H_i with the extended sign X_i,
ii. If the coefficient H_i is negative: multiply the amplitude of H_i with the extended sign two's complement of the input sample X_i.

The result of the multiplication will be in the two's complement form. Figure 1a shows the direct (transversal or canonical) form flow diagrams of equation (1) while Figure 1b shows the transposed structure form of N-tap FIR filter.

The main advantage of FIR filters is its linear-phase response. This property is achieved when the impulse response satisfies the symmetry or non-symmetry conditions. Symmetric FIR filter topologies for both canonical and transposed forms are shown in Figure 2. The non-symmetrical structure needs N multipliers. This number is significantly reduced in the symmetrical case: (N-1)/2 if N is odd and N/2 if N is even.

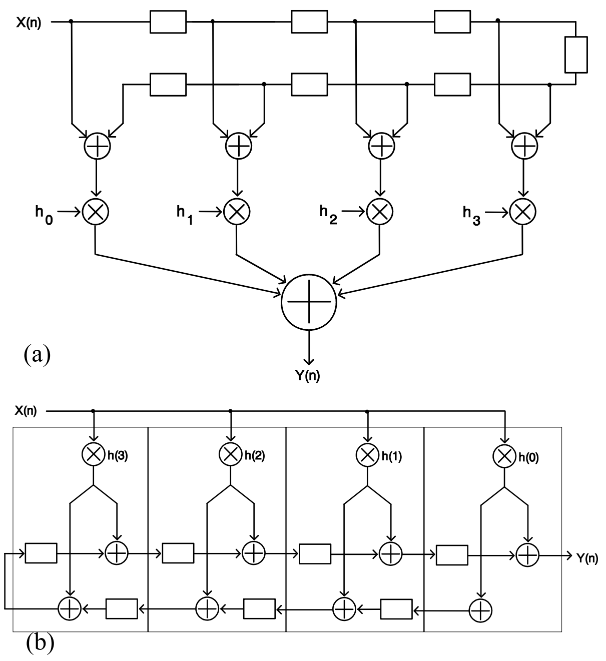


Figure 2 Symmetric FIR filter structures: a) canonical form; b) transposed form

3. THE PROPOSED FPGA-BASED FIR FILTER STRUCTURE

The N-tap FIR filter structure is shown in Figure 3. It consists of one input/output unit and an array of N multiply-accumulate (MAC) blocks with coefficient storage. In the case of a nonsymmetrical filter, each MAC cell receives two sets of signals, one from the input/output unit and one from the previous cell. The cell processes the received data and propagates the result in serial form to the next cell. In case of symmetrical filter, each cell receives three sets of data; one set from the input/output unit, the second an accumulated value from the previous cell and the third an accumulated value from the next cell. The cell processes the received data and generates two accumulated values; one of these propagates to the next cell and the other to the previous one. In the following, the basic blocks are considered.

3.1 The Multiplier-Accumulator (MAC) Unit

The basic unit in our design is the MAC unit. For N-tap filter, at sampling instant n, the i^{th} MAC block receives the input sample $X(n-i)$ and calculates the partial value y_{pi}, where:

$$y_{pi}(n) = H_i X(n-i) + [\sum_{J=i+1}^{N-1} H_j X(n-j)] \tag{4}$$

with,

$$y(n) = y_{p0}(n) = H_0 X(n) + \sum_{j=1}^{N-1} H_j X(n-j) = \sum_{j=0}^{N-1} H_j X(n-j)$$

The first term in equation (4) represents the multiplication operation and the second term represents the accumulated sum of all the terms of the filter equation starting from the last term ($H_{N-1}X(n-N+1)$) up to $H_{i+1}X(n-i-1)$. This means that the MAC unit implements the multiplication and accumulation operations. To achieve its function, the MAC is composed, in general, of three blocks: two's complement circuit controlled by the sign of the coefficient; next is a serial-multiplier to implement the serial-bit multiplication; the third block is a storage section. The storage section consists of one parallel-in (or serial-in)-parallel-out register to store one of the coefficients and one serial-in-serial-out register to store the accumulated value. As mentioned in section-1, to keep full precision along the whole datapath the length of the register storing the accumulated value is ($W_{\text{in-data}} + m_{\text{coefficient}} + \log_2 N$) and the serial multiplier needs the same number of cycles to complete the multiply/accumulate operation.

To reduce the hardware requirements and to reduce the processing time, we are proposing the use of one common two's complement circuit as part from the input/output unit and to use a new serial multiplier (serial-digit multiplier slice) that needs half the number of cycles to get the result. The proposed serial-digit multiplier generates two bits simultaneously from the partial value y_{pi} at each cycle.

In the following sections we are going to start by introducing the serial-bit multiplier and then the other blocks that form the MAC cell.

Digit-Serial Architectures

In digit-serial computation, data words of size L bits are partitioned into digits of size K bits (the digit-size, K, is divisor of the word-size, L) and are processed serially one digit at a time with the least significant digit first. A complete word is processed in P = L/K clock cycles and consecutive words follow each other continuously. The time of P cycles is named a sample period. In every digit-serial operator, it is necessary to add some control signal to indicate a new word entry. The digit-serial processors include parallel-serial and serial-parallel converters to process in digit format and to present the result in parallel format. A set of digit-serial architectures can be designed by using different digit-sizes.

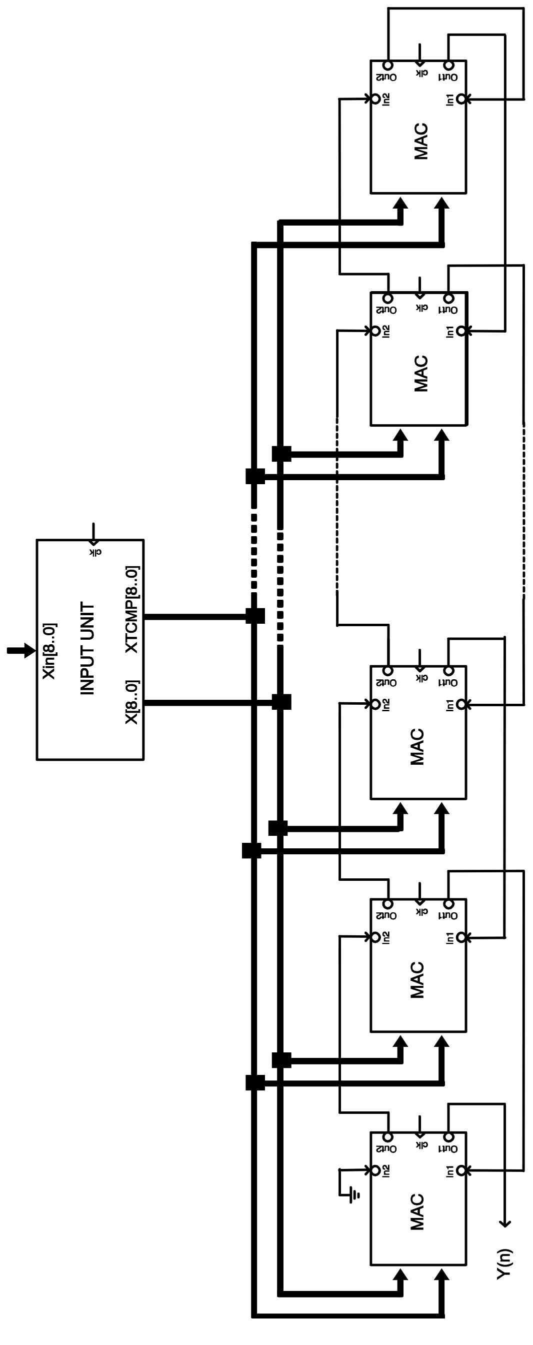


Figure 3 Proposed FPGA Implementation

To implement the multiplication operation $H.X$, we normally generate a partial product matrix with m (m is the width of H) rows and $W+m$ vertical lines (W is the word length of X). The bits forming vertical line j have the same weight 2^j. In our proposal we are going to deal with the bits of the vertical lines and split them into digits of two bits each.

2-bit Full Adder (2 bits Digit-Serial Adder)

Figure 4a shows the conventional single-bit full adder with three inputs and two outputs. The single-bit full adder adds bit a_i of word A, bit b_i of word B and an input carry bit c_{in} and generates the sum bit s_i and the output carry bit c_{out}. The basic element used to build the MAC of the proposed system is a two-bit full adder. In literature it is known as digit-serial adder with digit size K=2. This adder is shown in Figure 4b. The digit-serial full adder has, in its general form, five inputs (a_i, a_{i+1}, x_i, x_{i+1}, and c_{in}) and three outputs s_i, s_{i+1} and c_{out}, where s_i and s_{i+1} represent the sum of the input signals with the carry c_{out} propagates back to the input after one delay period. In case of FPGA devices, the digit-serial adder element can be implemented using a look-up-table (LUT) or, as in our implementation, by using two single-bit full adders together with some flip-flops to store the outputs. The digit-serial adder symbol is given in Figure 4c.

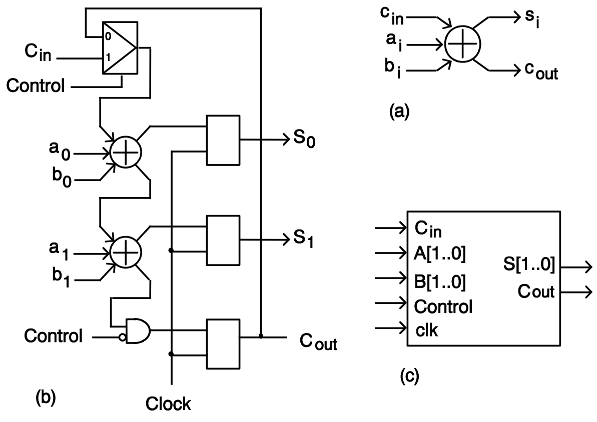


Figure 4 a) Full adder; b) digit-serial adder with K=2; c) symbol of the digit-serial adder

Digit –Serial Multiplier

The proposed digital-serial multiplier is shown in Figure 5 for the case of a filter coefficient word length of 8 bits. The word length of the input sample has no effect on the hardware. In case of using 16 bits coefficients, two circuits can be cascaded.

It is possible to look to the proposed digit-serial multiplier as a modified form of one of the slices used to implement the Wallace-tree parallel multiplier. Here, the slice processes at the same time two columns of the partial product matrix without any horizontal propagation for the carry. The depth of the slice equals to the word length of the coefficient A_i. The proposed digital-serial multiplier splits the partial product matrix of the product $A_i.X_i$ into vertical slices each of two columns of weight 2^j and 2^{J+1} and processes the slices in serial form to reduce the hardware. At cycle j, the MAC hardware generates the elements of columns number ($2j-1$) and ($2j$). The elements of the two columns represent the inputs of the digit-serial multiplier circuit. The outputs of the multiplier are the two bits s_{2j} and $s_{(2j-1)}$ of the final product. The circuit delays the carry bits generated within the circuit during cycle j to be used in the next cycle. To achieve the accumulation, equation (4), we use a one digit-serial adder to add the output bits s_{2j} and s_{2j-1}

of the multiplier together with the two bits that have the same weights of the previous accumulated value to get the bits of the new accumulated value. The process is repeated $(m+W)/2$ times to get the output. A control signal is used to clear all the carry bits before starting the multiplication of new words.

Multiplexer Block

To reduce the hardware requirements, we used one common two's complement circuit as part of the Input/Output block. The Input/Output block forms directly the two's complement of the received input signal and feeds the MAC with the direct and the two's complement forms of the input sample. The input stage of the MAC circuit is a set of *m* 2x1 multiplexers. The bits of the direct and the two's complement forms of the input signal are connected to the inputs of the M multiplexers. The sign bit of the corresponding coefficient controls the multiplexers.

Register Block

Each MAC has two registers:

1. $(m+1)$-bit register to store the coefficient a_i . The coefficient is stored in sign-amplitude form with *m* bits to store the amplitude. This register can be either serial-in parallel-out (SIPO) or parallel-in-parallel-out (PIPO) register.

The coefficients are stored as constant cells in the FPGA architecture. Any one or more of the coefficients can be modified by sending the appropriate bit stream(s) to the FPGA.

2. Serial-in-serial-out (SISO) register: This register is used to store the intermediate values y_{pi} . The MAC can be designed to processes the data with full precision. The length of this register will vary accordingly from $W+m+1$ for $i = N-1$ to $W+m+\log_2 N$ for the output MAC (i.e. for $i = 0$). In our case, we select double precision, i.e., the register length equals to $(m+W)$.

3.2 The Input /Output and Control Unit

Both the input and the output of the filter are serial data streams with each word presented least significant bit first. The input words are of the same length as the output. For precision processing, the intermediate word length is equal to the sum of the word sizes of the input (number of bits excluding sign extension required for processor), the coefficients (length of multipliers) and the number of levels in the digit-serial multiplier slice. The word size of the input need not be the same as that of the coefficients. The input is sign extended to bring it to the same length as the output. This requirement is due to the nature of the serial processing; the inputs need to be as long as the outputs. The extra few bits due to the digit-serial multiplier allow the column sum to grow without overflow. The multiplier array must be reset before each new word begins to shift in. The delays, however, cannot be reset since they hold the old words. In the FPGA implementation, a local reset was wired to the array columns corresponding to the multipliers

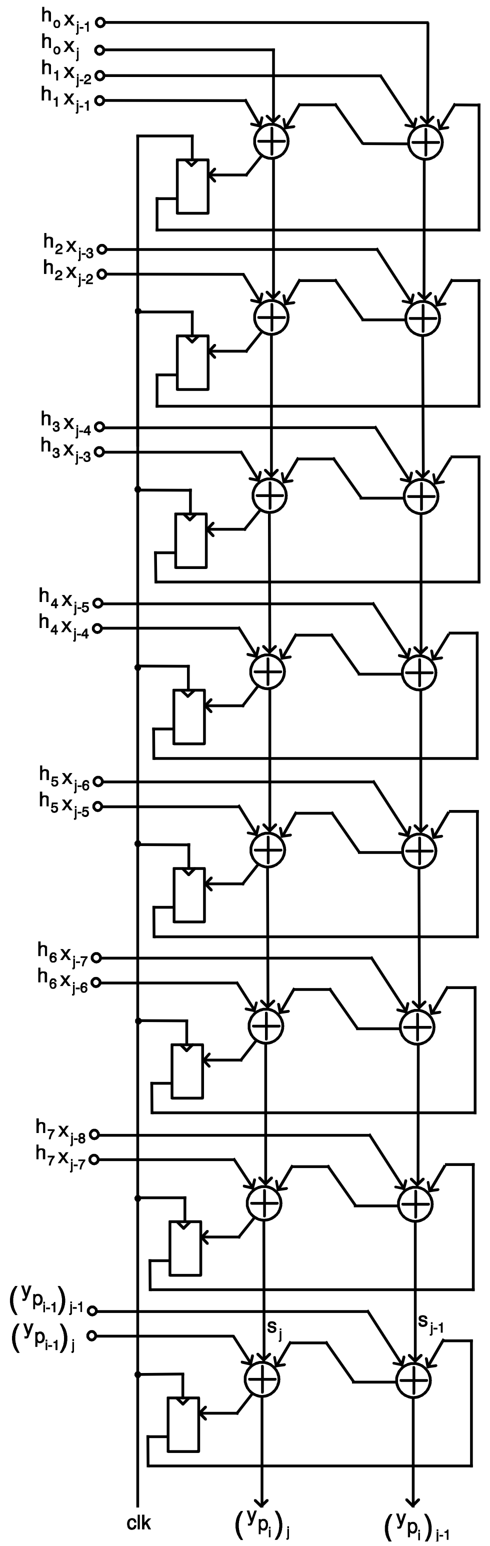


Figure 5 Digit-serial array multiplier with digit-size K=2

instead of the global reset. This local reset was brought out as a control line in addition to the global reset that clears the filter.

Cascading two filters may obtain more taps. This is done by adding an extra word delay to the end of the delay chain to feed the serial input of the second filter. The serial outputs of the two filters are summed using a serial adder to obtain the final output. This expansion scheme can be extended to create any number of taps. The block diagram of the Input unit is shown in Figure 6. It consists of three main blocks: input registers; bit-serial two's complement block; and control unit.

Bit-Serial Two's Complement Unit

In the proposed FIR filter, the coefficients and the sample inputs are signed numbers. As mentioned before, the system uses two's complement representation for the input samples, and sign-amplitude representation for the coefficients. To allow all the MAC blocks used to build the filter to work in parallel, each block must contain a separate bit serial two's complementor. This solution presents a large waste in the hardware. To reduce the hardware, the circuit of the proposed MAC has two sets of inputs each of *m* bits. The first set carries the bits of the input sample $X(n-i)$ directly, and the other input receives the bits of the two's complement of the input sample. The two sets of input signals are connected to an *m* 2x1 multiplexers controlled by the sign bit of the corresponding coefficient. By this way, the system uses only one common two's complementor which is a part of the input unit of the filter. The circuit of the two's complementor consists of two flip-flops, an XOR gate and an OR gate. One of the two flip-flops acts as detection flip-flop and it must be reset before starting processing each new input sample. This is because the detection flip-flop causes the XOR to invert the input continuously after the first "1" is detected.

Multiple Precision to Single Precision Block

The full precision data has to be truncated to single precision before going into the final serial-parallel converter. In the case of *W*-bit word, the only useful bits for us are the *W* most significant bits of the result. Therefore, to feed these *W* bits into the serial-parallel converter these have to arrive with the right format.

Control of the filter is achieved by generating an initialization signal at each new sample time. This single clock-cycle wide pulse is delivered to the filter as the LSB of each sample is presented to the multipliers. This signal ensures that the carry signals are reset at the beginning of each process cycle. (Delayed versions of this signal are input to the serial column adder, initialising each carry-save adder in the adder tree).

4. FPGA IMPLEMENTATION

The filter described above was implemented in an ALTERA EPF10K200SRC240-1 FPGA. Default placement and routing compilation options were selected.

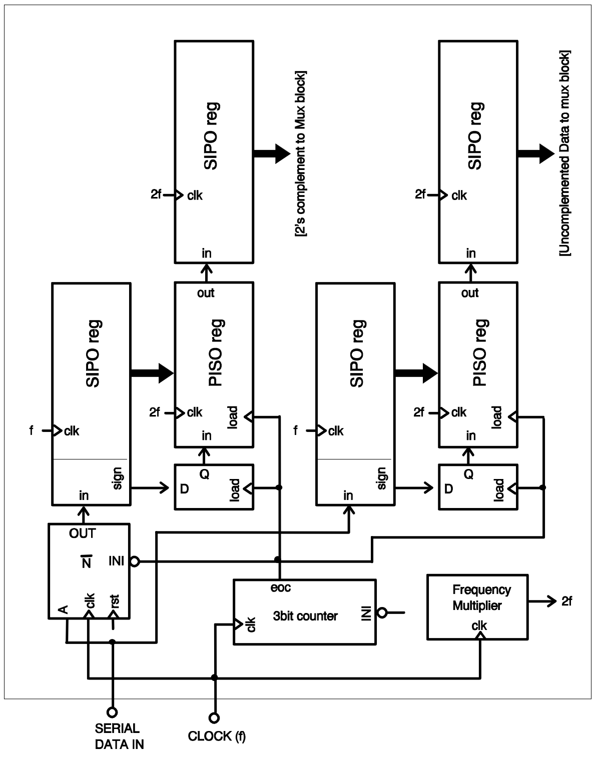


Figure 6 Input/Output Unit

The FPGA chip accommodated the Input/Output/Control unit and 100 MAC units allowing 200-tap symmetrical FIR filter. The bit-serial implementation reaches a real –time operation nearly to 7.5 MHz. As a matter of fact, another version is tested using LUT to implement the digit-serial adder.

5. CONCLUSIONS

A study of full precision digital-serial FIR filter with programmable coefficients has been presented. Symmetrical and nonsymmetrical inverted form FIR filters have been considered. The design methodology using each of these structures has been detailed and implemented in an ALTERA FPGA device. The area-time calculation for the proposed system and some of the existing system showed a great improvement using our system. The main improvement came from the proposed digit-serial multiplier. As a matter of fact, the hardware requirement to build one of the digit-serial multiplier slice is *W*/2 times less than the hardware required to build one parallel multiplier.

The filter has been automatically implemented using the default parameters of the partitioning, placement, and routing tools. Thus, in critical applications, an important area reduction and speedup can be expected if some of these tasks are manually performed.

6. REFERENCES

[1] R. Petersen and B. Hutchings, "An Asssesment of the Suitability of FPGA-based Systems for Use in DSPs", in Lecture Notes in Computer Science, nº975, pp.293-302, Springer-Verlag, Berlin: 1995.

[2] Chi-Jui Chou, Satish Mohanakrishnan, and Joseph B. Evans, "FPGA Implementation of Digital Filters", Proc. Int. Conf. Signal Proc. Appl. & Tech. (ICSPAT'93), 1993.

[3] J. Isoaho, J. Nousiainen and O. Vainio, "FPGA-implementable Digital Filters", More FPGAs, Eds. W.R. Moore and W. Luk, Abindon EE&CS books, 1994.

[4] Joseph B. Evans, "Efficient FIR Filter Architectures Suitable for FPGA Implementation", IEEE Trans. Circuits & Systems, July 1994.

[5] R.J.Andraka, "FIR filters fits in an FPGAs using a Bit-Serial approach", 3rd PLD Conference, Mar 1993.

[6] J. Hancq, H. Leich, "Implementation of digital filters on reconfigurable Field-Programmable Gate Arrays", Signal Processing VII: Theories and Applications, Eds. M.Holt, C. Cowan, P. Grant, W. Sandham, 1994.

[7] L.E. Turner, P.J.W. Graumann, and S.G. Gibb, "Bit-serial FIR Filters with CSD Coefficients for FPGA", in Lecture Notes in Computer Science, nº975, pp.310-320, Springer-Verlag, Berlin: 1995.

[8] Shousheng He, Mats Torkelson, "FPGA Implementation of FIR Filters Using Pipelined Bit-Serial Canonical Signed Digit Multipliers", IEEE 1994 Custom Integrated Circuits Conference.

[9] H. Lee and G. Sobelman, "FPGA-Based FIR Filters Using Digit-Serial Arithmetic,", Proc. of the Tenth Annual IEEE International ASIC Conference and Exhibit, pp. 225-228, 1997.

[10] Les Mintzer, "FIR Filters with Field-Programmable Gate Arrays", Journal of VLSI Signal Processing, vol. 6, pp. 119-127, 1993.

[11] Xilinx, "The fastest filter in the west", Xilinx Inc., http:\www.xilinx.com.

[12] "The role of the distributed aritmetic in FPGA-based signal processing", Xilinx Inc., http://www.xilinx.com.

[13] G.Goslin. A Guide to Using FPGAs for Application-Specific Digital Signal Processing Peformance. Xilinx Inc.

[14] G.Goslin, and and Bruce Newgard, "16-Tap, 8-bit FIR Filter Application Guide", Xilinx Inc., http://www.xilinx.com, 1994.

[15] Altera, "Implementing FIR Filters in FLEX Devices", A-AN-073-01, http;//www.altera.com.

[16] Altera, "FIR Filters",A-FS-01-01, http:// www.altera.com, 1996.

[17] ALTERA, "Data Book", 2000.