4.4. Linear Filtering

4.4.1. Parallel synthesis of FIR digital filters

A coefficient dependent multiplication scheme for Field Programmable Gate Array (FPGA) implementation of direct form Finite Impulse Response (FIR) digital filters. Using conventional design methods FPGA devices are not very efficient in digital FIR filter implementation due to the complexity of the multipliers. Analysis of filter coefficients shows that a large number of redundant leading sign bits can be find in their two's complement binary representation. Coefficient dependent multiplier logic synthesis decreases the needed amount of Configurable Logic Blocks (CLBs) by 30-40%, determined by the filter order and the selected coefficient accuracy. Memory based bit-serial distributed multiplier and accumulator units (DMACs) can be used effectively to realize multipliers with different coefficient size on the XILINX XC4000 third generation FPGA family.

The basic transversal FIR filter structure

FIR filters are one of the most important class of the digital filters. Although FIR filters can have recursive computation methods, practically they are always implemented as transversal filters. The signal flow graph (SFG) of the transversal filters shown in Fig. 4.4.1 is a simple tapped delay line, without any

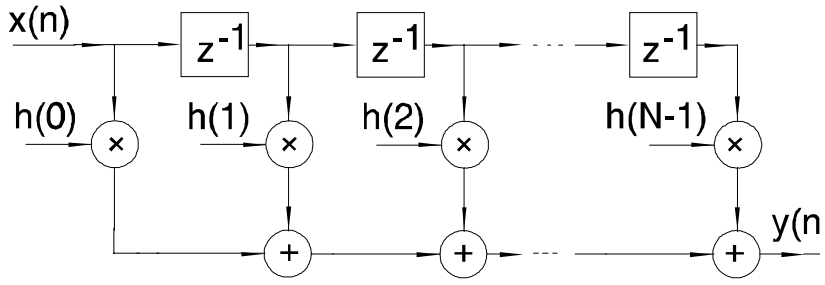

Figure 4.4.1: Transversal FIR filter

feedback, in which the outputs of each stage, multiplied by a fixed constant, are added together. The most important advantage of these transversal filters is their inherent stability property.

The **y(n)** output sequence of a FIR filter is a linear convolution of the input data sequence **x(n)**, and the sequence of the filter coefficients **h(k)**, k=0, ... N-1.

$$y(i) = \sum_{k=0}^{N-1} h(k)\,x(i-k) \qquad i=0,\ldots\ M+N-2; \tag{4.4.1}$$

Linear convolution is a common computational problem in digital signal processing. There are many well known fast algorithms, which reduce significantly the number of multiplications, but they are usable only in block processing mode [1].

In the obvious implementation of the convolution the computation of a given output sample needs N multiplications and N-1 additions. The multiplications are usually performed according to the data representation, but the accumulation operates on extended double precision word size to preserve numerical accuracy and avoid overflow during computation.

General purpose digital signal processors contain internal parallel hardware multiplier and accumulator (MAC) architecture optimized for this type of computation. Because of the pipe-line operation, one stage of the transversal FIR filter is executed during an instruction cycle. The same functional units are used for every computation, there are no possibility to implement application specific properties. The basic data and filter coefficient accuracy are typically equal and are between 16 and 32 bits. These processors deal with word size operands.

The internal architecture of some special signal processor, such as INMOS IMS A100 is the direct implementation of the transversal FIR filter signal flow graph. The major processing part of this chip incorporates 32 stages of the filter. The MAC units are handle 16 bits data and 4, 8, 12 or 16 bits coefficients. The multipliers internally are not fully parallel, they work with 4 bit groups of the coefficients, like the digit serial arithmetic units.

The above mentioned signal processors ignore the actual numerical values of the filter coefficients, they have general purpose hardware architecture. In the case of dedicated implementations with fixed coefficient, they numerical values can be taken into account. Analysis of the filter coefficients shows, that their dynamic range is about 2 or 3 orders of magnitude. In two's complement binary representation of small values there are a large number of redundant leading sign bits, 0s or 1s. For example, if the absolute value is smaller than 10^{-2}, the first 6 sign bits has no extra information, and a 16 x 16 bit size operation can be carried out by 10 x 16 size hardware unit, using sign extension at the output. The next section will discuss this in depth and points out how it can be exploited to reduce the complexity of the selected memory based bit serial distributed vector multiplier accumulator (DMAC)

logic [7]. The third section introduces the target FPGA device from the point of view of digital signal processing and gives a short summary of the bit-serial operational primitives, defined for the FIR filter. Detailed description is provided about the DMAC units.

To support the design procedure a direct translator program is developed. This package defines the actual block diagram of the filter according to the input parameters, determines the coefficient dependent multipliers, and synthesizes the control and I/O interface logic.

The coefficient dependent multiplication scheme

The same result of a multiplication can be obtained with different multiplication schemes [2], [3]. Assume the product of two 2's complement binary coded words is desired, where the data and coefficient size is B_x and B_c bits long, respectively.

$$y = h\,x$$

$$\textit{where} \quad h = -h_0 2^0 + \sum_{i=1}^{B_c - 1} h_i 2^{-i} \quad \textit{and} \quad x = -x_0 2^0 + \sum_{j=1}^{B_x - 1} x_j 2^{-j} \tag{4.4.2}$$

This gives

$$y = \{-h_0 2^0 + \sum_{i=1} h_i 2^{-i}\} \cdot \{-x_0 2^0\} + \{-h_0 2^0 + \sum_{i=1} h_i 2^{-i}\} \cdot \sum_{j=1} x_j 2^{-j}$$

$$\textit{and} \quad y_k \in \{\, 0 \;..\; B_x + B_c + 1\} \tag{4.4.3}$$

where **y** is a B_x+B_c-1 bit long product. (Two's complement hardware multipliers in case of fractional input operands usually generate B_x+B_c bit long product with one bit sign extension.) In the common written form, the product is given as the sum of shifted partial products. Because of the sign of $\mathbf{x_0}$, the last partial product is subtracted. For fractional input operands, the product is also true fractional and smaller than any one of the inputs. For single precision output the rounded or truncated result bits are from $\mathbf{y_o}$ to $\mathbf{y_{-Bx+1}}$. The $\mathbf{y_1}$ product bit is already sign extension, and has the same logical value as $\mathbf{y_0}$.

In case of multiplications, where the two's complement binary code of the multiplicand is known a priori, the redundant leading sign bits (0s or 1s), except the last one, can be neglected. For positive values this statement is self evident, and for

negative values, it can be seen from

$$-h_0 2^0 + h_1 2^{-1} + h_2 2^{-2} \ldots + h_L 2^{-L} = -h_L 2^{-L} \qquad \textbf{(4.4.4)}$$
$$where \quad h_i = 1, \quad i = 0, \ldots L;$$

Assume, that $|\mathbf{h}| < \mathbf{2^{-L}}$, than the following two representation of **h** are equal.

$$-h_0 2^0 + \sum_{i=1}^{B_c - 1} h_i 2^{-i} = -h_L 2^L + \sum_{i=L+1}^{B_c - 1} h_i 2^{-i} \qquad \textbf{(4.4.5)}$$

This gives smaller partial product length and of course reduced logic complexity. Using this identity, the B_c x B_x size multiplication is feasible by (B_c-L) x B_x size hardware units. If L is comparable with B_c, then the degree of simplification is higher.

The numerical values of the elements of the impulse response sequence of the FIR filters are usually smaller than 1 and have a wide dynamic range. An example low pass filter **h(i)** coefficients are given in Table 4.4.1, the frequency response function is shown in Fig. 4.4.2, and the magnitude distribution of **h(i)**-s as a function of L is presented in Fig. 4.4.3.

Table 4.4.1. Example filter coefficients
Linear phase FIR filter, N=49, Fc=.255, Hamming window

h(0) = h(48) = 7.2633626e-004	*h(12) = h(36) = 5.2730628e-003*
h(1) = h(47) = -8.7136148e-004	*h(13) = h(35) = -1.6337261e-002*
h(2) = h(46) = -8.8238038e-004	*h(14) = h(34) = -6.4827732e-003*
h(3) = h(45) = 1.3775415e-003	*h(15) = h(33) = 2.4319301e-002*
h(4) = h(44) = 1.3249345e-003	*h(16) = h(32) = 7.6192901e-003*
h(5) = h(43) = -2.4256632e-003	*h(17) = h(31) = -3.6391525e-002*
h(6) = h(42) = -2.0347079e-003	*h(18) = h(30) = -8.6016522e-003*
h(7) = h(41) = 4.1898892e-003	*h(19) = h(29) = 5.6901874e-002*
h(8) = h(40) = 2.9711350e-003	*h(20) = h(28) = 9.3591583e-003*
h(9) = h(39) = -6.8818568e-003	*h(21) = h(27) = -1.0193494e-001*
h(10) = h(38) = -4.0750757e-003	*h(22) = h(26) = -9.8369084e-003*
h(11) = h(37) = 1.0785544e-002	*h(23) = h(25) = 3.1690480e-001*
	h(24) = 5.1000648e-001

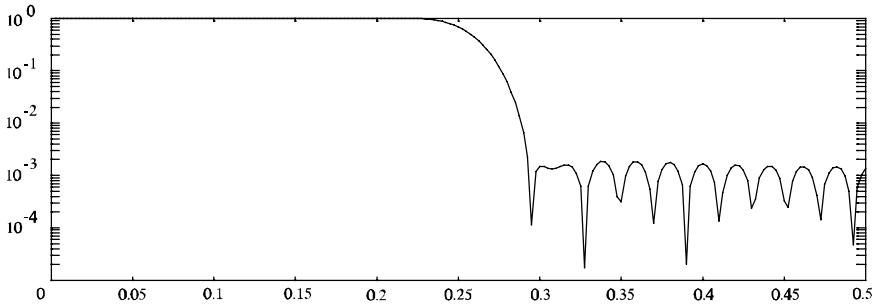

Figure 4.4.2: Frequency response of the example filter

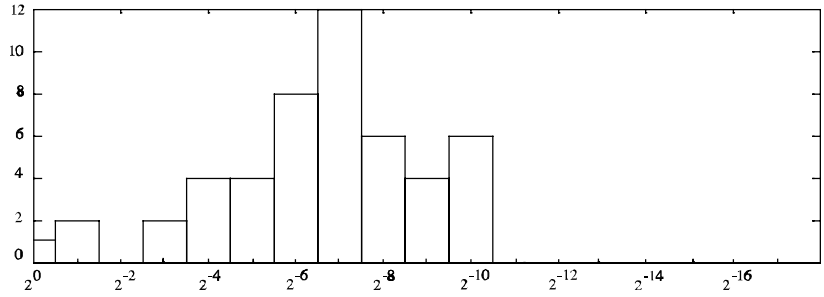

Figure 4.4.3: The magnitude distribution of $h(i)$s

Different FIR filters in type and order have been investigated and the observed distribution is similar. This directly follows from the shape of the impulse responses of the filters. It should be noted however, that as the filter order is increased or the passband width is decreased the maximum of the distribution is shifted right, raising the available reduction. According to Fig. 4.4.3, one can read out the ratio of the possible hardware reduction for the example filter. For the typical 16 bit or smaller coefficient accuracy, the savings is significant, but in case of 20 or more bits is still more than 20%.

Fig. 4.4.3 shows that instead of 49 (or 25, if the symmetry of the linear phase filter coefficients in the given implementation can be exploited) full size multiplication, in more than half of the cases the effective bit length of the

coefficients is reduced by 6 or 8 bits for this filter.

Gate complexity of state of the art FPGA logic devices [4] already approaches or exceeds the 10,000 gate level, but this is still well under the requirements of digital signal processing. Special design methods and arithmetic structures should be applied for high utilization of the available logic resources. Application of CSD codes and bit serial arithmetic also leads an area effective implementation [5], and with careful logic design high bit clock rate can be obtained.

The bit serial operational primitives

Basically an FPGA consists of three configurable elements: configurable logic blocks (CLBs), input/output blocks (IOBs), and programmable interconnect. The CLBs are the main resources of the logic and their complexity and flexibility has a great effect on the device performance. In the XILINX 4000 family the CLBs have two 4-input independent function generators. These function generators are implemented as memory look-up tables, allowing for the generation of any arbitrary logic function up to 4 variables. These 16x1 size memory modules in a special configuration mode can be used as RAMs also, and the read/write cycle time in that case is the same as the logic delays. In the CLBs there are two D type flip-flops with some programmable options. Both the function generators and the registers have their own inputs and outputs.

One of the most advanced feature of the 4000 family CLBs is the built-in hidden carry logic. This carry logic significantly increases the speed and efficiency of parallel adder/subtracter units. The carry logic has separate, dedicated routing resources connecting adjacent CLBs above and below each other. Using this carry logic one CLB can realize 2-bit arithmetic element.

Although the routing resources are not part of the logic, their hierarchy fits well to this filter implementation. Global networks are used for time critical control signals, long lines and short lines for normal routing.

The SFGs of the transversal FIR filters can be constructed from a limited number of operational primitives (OPs). This type of functional and hierarchical design methodology for bit-serial signal processing was discussed in [6]. The basic primitives are the bit-serial adder, subtracter, multiplier, bit and word size delays. All these elements are treated as primitives although some of them may be very complicated logic modules. The SFG OPs library is completed with the control and I/O interface primitives. The former contains counters and special control signal generators, the later includes parallel, serial LSB first and serial MSB first I/O building blocks.

In most cases the OPs are directly mapped to the CLBs of the FPGA. The bit-serial adder and subtracter primitives need 1 CLB. The bit delays are simple D flip-flops connected as shift registers. The word size z^{-1} delays shown on Fig. 4.4.1. are

realized using the CLB function generator special RAM configuration option. The hardware cost of this type of word delay is 1/2 CLB/word and 1 CLB/word up to 16 bit or 32 bit wordlength, respectively. Without this RAM option the proposed filter implementation would be practically unrealizable. The memory address generation is managed by a programmable hard macro address counter.

The most complicated OP is the multiplier. Analysis of the SFG in Fig. 4.4.1. shows that multiplications form an inner product-like structure, where the products are summed at a common junction point. This type of inner product, where multipliers are known constants, can be effectively realized by a memory look-up table based distributed vector multiplier [7]. Block diagram of the multiplier is shown in Fig. 4.4.4.

The ROM contains precomputed linear combinations of the coefficient values. The depth of the memory is determined by the number of serial input multiplicands. The width of the memory and that of the parallel adder/subtracter are equal to the multiplier coefficient effective data size. In every bit cycle a new partial product sum is added to the earlier, right-shifted value of the accumulator. If the input and coefficient data length are B_x and B_c, respectively, then the B_x+B_c-1 bit long product will be shifted out serially on SPR. For 2's complement input data representation the ADD/SUB units perform subtraction only in the B_x-th MSB bit cycle. After that the ROM output register is cleared and the most significant part of the "high precision" product scheme is shifted out with sign extension on the most significant bit position.

In the XILINX XC4000 FPGA family, because of the given 16x1 size ROM function generator the most cost effective, optimal number of the input signals for the multiplier modules are 4. Because of this (and the symmetry of the impulse response of the linear phase FIR filters) the number of the multiplier modules is $\lceil \mathbf{N/8} \rceil$. The serial product streams are summed by a binary addition tree. The hardware cost of one multiplier module is $\mathbf{B_c-L+1}$ CLBs, so it is near $\mathbf{(B_c-L)/4}$ CLBs/multiplication. Using the magnitude distribution of the impulse response to create the 4 size groups of the $\lceil \mathbf{N/8} \rceil$ multipliers, significant savings can be achieved. Different sizes of multiplier modules from 2 bit to 24 bit have been developed as library hard macro modules, because automatic logic synthesis using the dedicated carry logic is not available, only for some fixed configurations.

The filter data flow is completed with synchronizing delays and appropriate control signals. The OPs have no built in control path. An independent control network is realised using control primitives. The timing of the control signals is determined by the B_x data size and N, the filter order. The most important, time critical control signals are distributed on the whole chip using the special global networks of the FPGA.

The natural I/O format for the presented arithmetic is the bit-serial LSB first format. However, if required parallel, or bit-serial MSB first, or any combinations of these ones can be selected. These I/O options have no effect on the computational speed, only on the latency in the case of MSB first format selection.

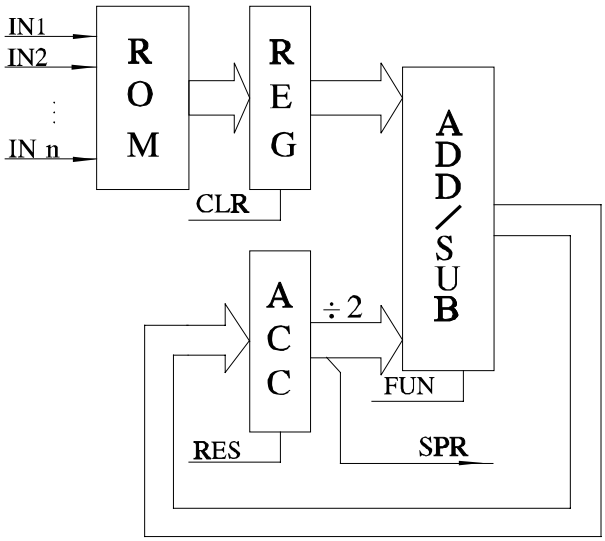

Figure 4.4.4: The multiplier module

Peformance analysis of synthesised FIR filter

Table 4.4.2. gives a short summary about the CLBs requirements of the example filter with and without coefficient dependent logic synthesis. In this case $\mathbf{B_x=B_c=16}$ bits were selected and the symmetry of the linear phase filter impulse response is exploited by pre addition of the equal components. Post additions generate the output signal from the serial product streams.

Table 4.4.2. CLB requirements of the example N=49 FIR filter

Hardware element	Direct method	Coefficient dep. method
Delay line	24	24
Pre additions	24	24
Multiplications	111	75
Post additions	8	8
Control	18	18
	185	149

The reduction in the multiplier logic is more than 33 %. In the second case there have left enough resources to realize any I/O option for example on the 196 CLB

complexity XC4005 device. The full synchronous pipe-line architecture of the filter logic ensures high bit clock rate, according to the development system timing analyzer up to 30 MHz. The computation of a given **y(i)** output sample needs $\mathbf{B_x+B_c+\log_2(N/2)}$ bit clock period. This means about 500 kHz sampling rate for the example filter.

This subchapter has proposed a new method for digital FIR filter implementation on XILINX 4000 FPGA devices. Exploiting the redundancy in the binary representation of the filter coefficients essential complexity reduction is achieved. The flexible bit-serial DMAC arithmetic units and the programmability of the logic elements allows coefficient dependent, application specific logic synthesis. Using the suggested method a relatively high order transversal FIR filter can be realized by the XILINX XC4005 10,000 gate level device. The example filter needs about 170 CLBs. FPGA FIR filter implementations are comparable in performance by the usual DSP processor based realizations. An automatic filter design tool is developed under the MATLAB environment, which directly generates the netlist file from the filter coefficients using the bit serial operational primitive library.

4.4.2. Parallel synthesis of IIR digital filters

A resonator based digital filter (RBDF) implementation is presented using field programmable gate array (FPGA) elements. The realization is based on the common structure developed for recursive discrete transforms. The globally parallel property of the recursive structure is maintained, while the arithmetic operations are realized by bit-serial logic. LSB first arithmetic combined with memory table based distributed vector multipliers gives an area effective realization. Third generation FPGA devices provide efficient tools to build up DSP signal flow graphs (SFGs). An automatic filter design tool is under development, which will directly compile the netlist file from the filter coefficients.

The resonator based IIR digital filters

RBDFs are descended from a common structure designed to realize recursive discrete transforms [Pec86]. The structure decomposes the input signal to orthogonal components as defined by the selected recursive transformation. The parameters of the structure are the values of the basis and reciprocal basis vectors of the transformation [Pec86]. This general arrangement leads to time variant computational parameters. A typical example is the recursive Walsh-Hadamard transformation (RWHT). In the RWHT, the time variant parameters are the +1 and -1 values, so they have little effect on practical realization. The application of the RWHT to digital filtering and an implementation on FPGA elements are presented

in [Pec90] and [Feh89], respectively.

The parameters of the common structure will be time invariant in the case of the recursive Fourier transform, or the RBDFs based on it. The corresponding modified structure is shown in Fig. 4.4.5. This special arrangement consists of a regular parallel connection of digital resonators supplemented by a common feedback loop. All of the resonators get the same update signal and operate independently of each other.

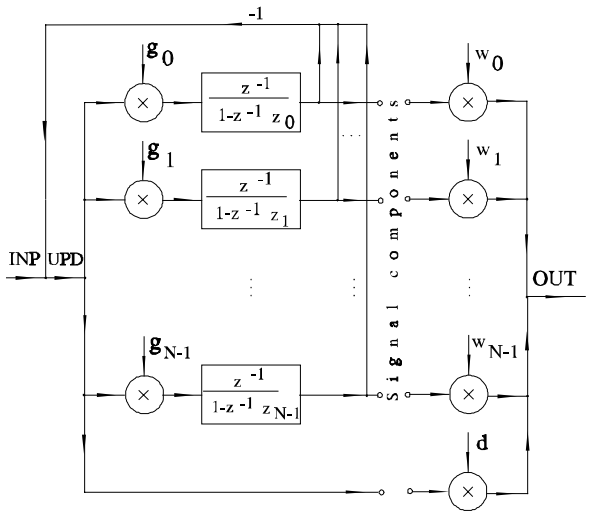

Figure 4.4.5: The common structure

The practical implementation requires first order sections for real resonators, and, second order sections for complex conjugate resonator pairs. In order to reach the above mentioned good filter parameters the resonator SFGs should be selected carefully. If the state variables in the resonator sections are orthogonal to each other and magnitude-truncation quantization strategy is applied, then the filter passivity is preserved [Pec89].

A possible second order resonator block is shown in Fig. 4.4.6. This arrangement fulfils the requirements and its SFG is uniquely suitable for the implementation on the selected FPGA device using bit-serial arithmetic.
This paper presents an implementation of the common structure using third generation XILINX XC4000 FPGA elements. The 2's-complement fixed point LSB first bit-serial arithmetic fully utilizes the architectural benefits of the selected device. The the most important features of the XILINX FPGA related to DSP type design

implementations can be found in Chapter 4.2, so it is not repeted here. The same is applies to the elements of the bit-serial operational primitive (OP) library defined to realize digital signal processing SFGs (see at Chapter 4.4.1). At the end of the Chapter, a short summary presents the implementation of an example 8th order elliptic bandpass filter.

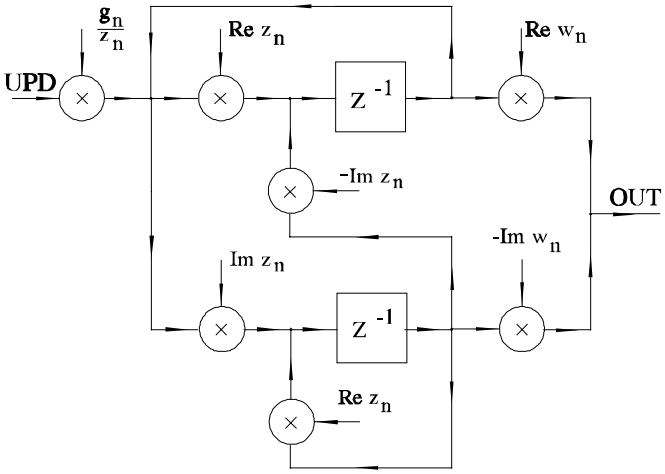

Figure 4.4.6: The second order resonator section

To support the RBDF design procedure a direct translator program is under development. This package will define the actual block diagram of the filter according to the input parameters, determine the contents and scale factors for memory based vector multipliers and synthesize the control logic. This filter implementation tool shortens the design time to an hour.

The prototype elliptic 8th order filter implementation

A prototype 8th order bandpass filter implementing the structure of Fig. 4.4.5. and Fig. 4.4.6. has been designed using conventional schematic entry tools. The selected device is the XC4005PG156, with its approximately 5000 gate density and 14 x 14 size CLB matrix. The data formats are 16 bit for the input and 12 bit for the coefficients. The four second order resonators are implemented with 8 multiplier module.The parallel adder/subtracter and accumulator unit in the multiplier was designed by the XACT Design Editor and defined as hard macro module. This

design method was selected because it provides full control of the hardware resources of the CLBs. Hard macros preserve their internal partitioning and relative placement information and they are the best way to design timing-critical functional modules. The hard macro CLBs are placed in one column of the array, following the predetermined carry signal routing. Three different CLB configuration are used. The LSB and MSB logic differ from the intermediate 2-bit configuration. The LSB CLB needs the incoming carry control. The MSB CLB on the other hand is a special 2-bit block without carry output. The netlist of the first multiplier module was generated by the design system from the schematic, and the later ones were copied and tailored from the netlist file using text editor. For all definable logic elements in these units direct placement constraints were given. ROM modules were placed directly onto the hard macro. This placement information and the regularity of the structure itself makes it possible for the XILINX Partition Placement and Route (PPR) program to finish the generation of the LCA configuration file in less than 30 minutes. The layout of the prototype filter is shown in Fig. 4.4.7.

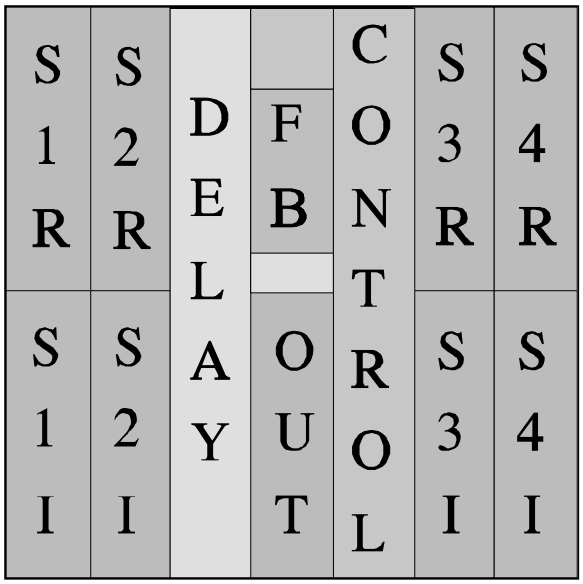

Figure 4.4.7: The layout of the prototype 8th order elliptic filter

The SxR and SxI blocks are the real and imaginary parts of the second order resonator sections, respectively. The synchronization bit delays, the truncation logic

and z^{-1} word delays are grouped together (DELAY) in the centre of the chip. The feedback serial adders (FB) and the output multiplier (OUT) are also placed near the resonator outputs. The prototype filter uses the simple LSB first I/O data format. The most important control signals (CLK, RES, CLR) are distributed on primary global nets. Memory address lines in one column use the vertical long lines.

The automatic filter sythesiser and netlist generator

The prototype filter implementation proved that using bit-serial arithmetic, a RBDF SFG can be realized on the target device. The regularity and modularity of the RBDF suggests an algorithmic block diagram and netlist generation. All filters of this type have the same generic architecture. According to the filter order N, there are N/2 parallel second-order resonators. Occasionally first-order resonators also can occur, but only in pairs. These two first order sections are handled in a similar way, as the real and imaginary halves of a second-order section. The filter-specific blocks are the feedback, control and I/O. They are synthesized in compliance with data format. The DSP library contains only a limited number of operational, interface and control primitives. The standard, common input for the XILINX development system is the Xilinx Netlist File (XNF) format [Xil91]. The development system supports the hierarchical design method, so functional blocks can be built up from the basic primitives. These functional blocks are treated as macro symbols and only their I/O connections should be given. All information related to the design is contained in XNF files and hard macro files, which have precompiled formats. In the case of the XC4000 family, the PPR program reads the main XNF netlist file and flattens the design according to its hierarchy.

The implementation process is shown in Fig. 4.4.8.

Any filter design tool can be used to determine the filter H(z) transfer function. Fig. 4.4.8. shows MATLAB, as an example only, the development sofware needs only the parameters of the filter transfer function to be realized. The next step is to derive the parameters of the resonators from the filter H(z) transfer function following the algorithm given in [Pec89]. The filter netlist generator reads these resonator parameters from a file, or in interactive mode, from the keyboard. It needs the B_c coefficient data to calculate the contents of the multipliers look-up tables. B_c data format is also used as a selector among the different sizes of adders/subtracters. There is no possibility of building up this modules from basic CLB primitives. The B_x input data wordlength determines the timing of the control signals, and is used with the I/O mode specification to generate the selected I/O module. On demand, special interface logic can be designed and merged to the filter if the FPGA resources allows it. The netlist generator checks the program input, selects a device from the XC4000 family, and generates the XNF files. From these, the PPR will generate the FPGA configuration LCA file.

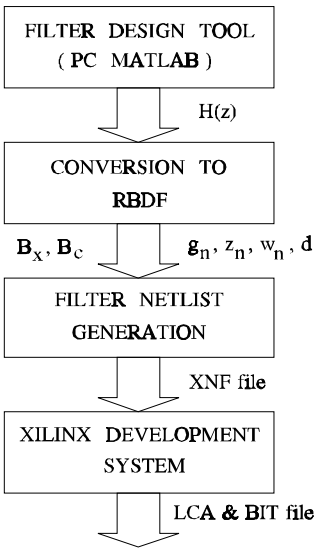

Figure 4.4.8: The automatic netlist generator program

The filter netlist generator program have been written in C, the implementation algorithm, the decomposition of the common resonator structure to second order sections and feedback summation modules are directly built in. The program generates the Xilinx technology specific netlist database and prints out all of the design parameter in a report file.