AUTOMATIC IMPLEMENTATION OF FIR FILTERS ON FIELD PROGRAMMABLE GATE ARRAYS

Satish Mohanakrishnan and Joseph B. Evans
Telecommunications & Information Sciences Laboratory
Department of Electrical Engineering & Computer Science
University of Kansas
Lawrence, KS 66045-2228

October 7, 1993

ABSTRACT

This paper describes a CAD system for automatic implementation of FIR filters on Xilinx Field Programmable Gate Arrays. Given the frequency specifications, this software package designs an FIR filter, optimizes the filter coefficients in the power of two coefficient space and implements it on an FPGA chip. The FPGA specific mapping techniques used to increase speed are discussed. The performance of the typical filters which were implemented is presented.

This research is partially supported by the Kansas Technology Enterprise Corporation through the Center for Excellence in Computer-Aided Systems Engineering and by the University of Kansas General Research allocation 3626-20-0038.

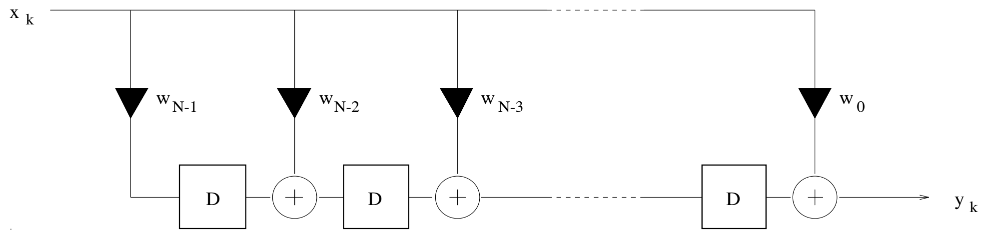


Figure 1: FIR Filter Structure

1 Introduction

Finite Impulse Response filters without full multipliers and their potential high speed VLSI implementations have received attention over the past decade [1, 2, 3, 4]. An efficient FIR filter architecture suitable for Field Programmable Gate Arrays (FPGA), which requires the coefficients to be a sum or difference of two power-of-two terms was discussed in [1]. In this paper, we present an improved filter tap structure and several mapping techniques which were used to increase the sampling rate. This paper also describes a CAD system which can be used for design of FIR filters, optimization of filter coefficients in the discrete coefficient space, and subsequent implementation on Xilinx XC3100 series FPGAs.

2 Background

In binary arithmetic, multiplication by a power-of-two is simply a shift operation. Implementation of systems with multiplications may be simplified by using only a limited number of power-of-two terms, so that only a limited number of shift and add operations are required.

In order to obtain good performance using a small number of such terms, the number of power-of-two terms used in approximating each coefficient value, the architecture of the filter, and the optimization technique used to derive the discrete space coefficient values must be carefully selected. It was demonstrated in [4] that an FIR filter with -60dB of frequency response ripple magnitude can be realized using two power-of-two terms for each coefficient value.

An inverted form FIR filter, which will be used in our FPGA implementations, is depicted in Figure 1. If the coefficient value is an integer power-of-two or a sum of two powers-of-two, the multipliers can be replaced by one or two shifters. Since the coefficients will be fixed for this class of filter, the coefficient values can be realized by appropriately routing the inputs to the full adders in the filter structure. That is, moving the adder inputs k places to the left achieves the same effect as would a coefficient value of 2^k.

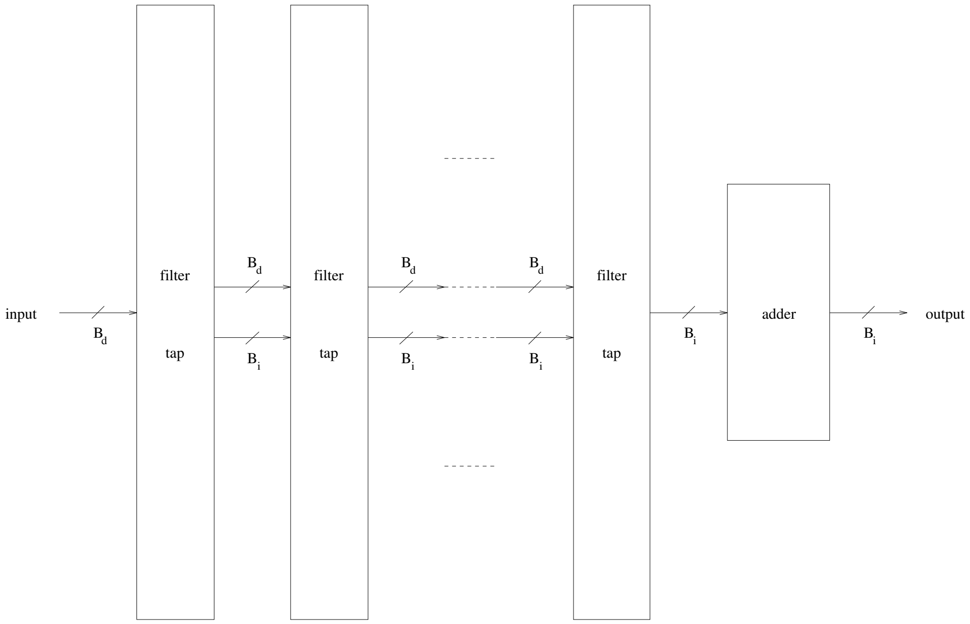


Figure 2: FIR Filter Architecture

3 Architecture

The overall filter structure is shown in Figure 2, where the filter taps and final adder stage are shown. The adder is required to resolve the carries that are generated and propagated through the pipeline. The structure of a portion of a typical filter tap is shown in Figure 3, where the internal pipeline is depicted. The two shifted versions of the data corresponding to the two power-of-two components of each coefficient are shown as dotted lines. Two adders are necessary for adding the sum and carry generated by the previous tap and the two shifted versions. The sign of the coefficients is controlled by inverters. The sum and carry signals from the full adders are pipelined using a carry-save addition (CSA) technique in order to increase the sampling rate and alleviate potential routing delays. The hardware requirements for a tap with B_d input datapath bits and B_i intermediate accumulation path bits are then $2B_i$ full adders and a minimum of $2B_i$ flip-flops.

4 CAD Tool

4.1 Filter Design and Optimization

The first stage in the design process is to obtain the filter coefficients. Given the frequency specifications, MILP3, written by Y.C. Lim [5] is used to obtain a continuous solution (which assumes infinite precision coefficient values). MILP3

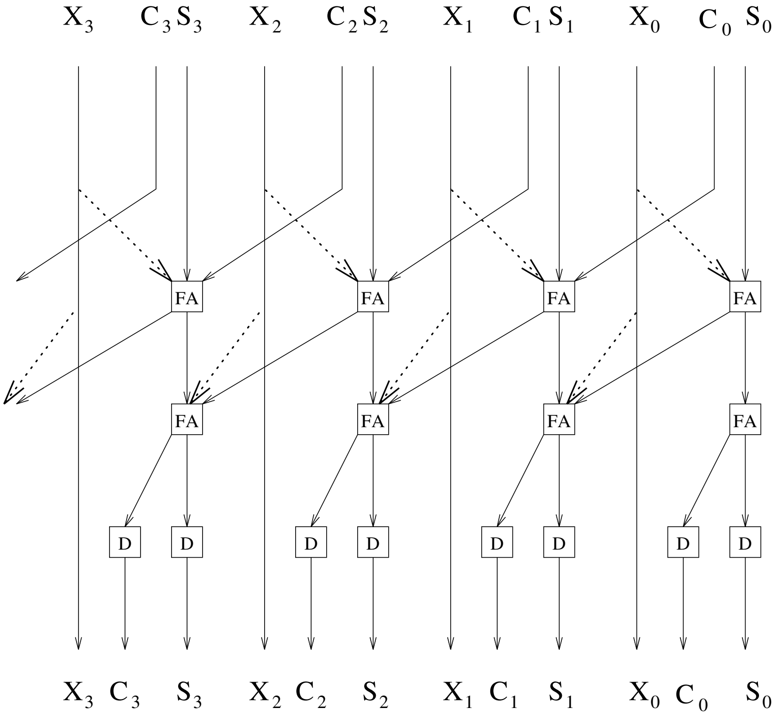


Figure 3: FIR Filter Tap Structure

uses standard integer programming techniques to optimize this continuous solution in the discrete powers-of-two coefficient space [2]. The resulting discrete solution has coefficients which are a sum or difference of power-of-two terms.

4.2 Xilinx Implementation

The output of the optimization stage is fed to code which maps the filter onto the FPGA. With the help of the Xilinx tools, the configuration details for the FPGA are then generated.

4.2.1 Place and Route

Due to the limited availability of global and local routing resources, placement of Configurable Logic Blocks (CLBs) and routing of nets are very critical in any FPGA design. The Automatic Place and Route (APR) program provided by Xilinx cannot be used to provide 100% placement for the following reasons. Due to the large of number of variables in the optimization problem, it takes an exorbitant amount of CPU time for placement. Since it is a general purpose package based on heuristic methods, it cannot always give the optimum placement for all the designs. For instance, when APR was given full freedom of placement for all of the 22 x 22

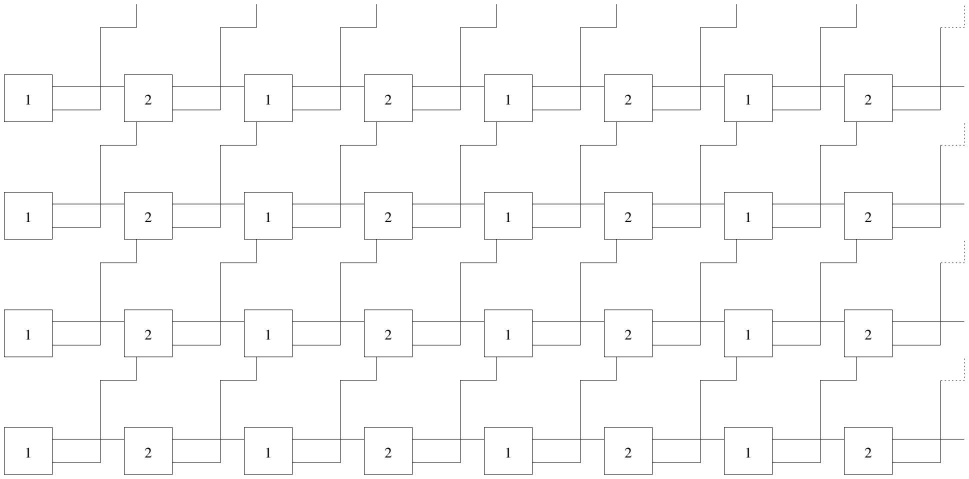


Figure 4: Mapping of the filter architecture on the Xilinx FPGA

array of CLBs for a 11 tap filter, it took 9 hours 2 minutes and 27 secs on a Sun SPARCstation–2 for the completion of placement and routing.

The mapping of the architecture in Figure 3 is shown in Figure 4, where each full adder is implemented in a Configurable Logic Block (CLB). The two rows of full adders map to alternate columns of the chip referred to as 1 and 2 as shown in Figure 4. To reduce congestion, the two shifted versions of the data are distributed among the two sets of full adders, whereas in the previous approach [1], they were routed to the first set of full adders. In the previous tap structure, the sum outputs of the second set of full adders in any tap are fed to the corresponding full adders in the next tap, which are two columns away. When the new structure is mapped onto the FPGA, the routing is only between CLBs which are in the adjacent columns. This makes more efficient use of the local routing resources. This structure has been found to achieve an improvement of 5 - 15% in the sampling rate for several typical filters.

The input data bus is distributed using horizontal long lines from one end of the chip to the other. By careful assignment of input pins and hence the horizontal long lines to the data, it is possible to reduce the maximum distance between any horizontal long line and a CLB where the data is needed. The assignment which gives the least distance and hence the delay varies from filter to filter. This optimization problem, with relatively a relatively small number of variables, is solved very effectively by APR to give an improvement of 20 - 30% in the sampling rate over the unoptimized placement.

4.3 Pin Constraints

For ease of design of printed circuit boards (PCBs), it may be necessary to have the input data pins in some particular order which will invariably not be the same as that of the optimum assignment found by the APR. As FPGAs are in-system reconfigurable, it is reasonable to impose pin constraints according to the existing PCB layout. Hence, inside the FPGA chip the the data lines are reordered by APR. This reordering inside the chip utilizes some routing resources and hence affects the sampling rate. The order of magnitude by which the sampling rate is affected is being investigated. It is possible to achieve performance close to that achieved by the optimum assignment using certain fixed assignments. To validate this, various assignments have been tried and their performances evaluated. Filters were implemented with data being distributed on alternate pins. Implementations were also done with assignments to alternate pins in the center of the chip. These fixed assignments achieved performance close to that of the optimum assignment by APR. Hence with any pin constraint, pin to fixed CLB and CLB to longline routings can be made without sacrificing speed.

4.4 User Interface

A Graphical User Interface (GUI) was designed using the Motif tool kit. As with other Motif GUIs, the interface has the basic menus, namely Design (File), Edit, and Help Menus. The Design menu has three options for the three main stages in the design process, that is, Frequency Specification, Discrete Space Optimization and Xilinx Implementation, as shown in Figure 5. The output of one stage is used as the input of the subsequent stage. The user can start at any stage, depending on the specification at hand. Thus a filter can be implemented from the frequency specifications or from an already existing infinite word length filter coefficient set or coefficients with power-of-two terms.

To begin a design, the Frequency Specification option in the Design menu is selected. Certain details such as the number of frequency bands and number of taps are entered in a dialog box. The input control option and symmetry option are selected using radio boxes. The input control option can be any one of minimal control (default), grid density control and band specification type control. The symmetry option can be either symmetrical (default) or antisymetrical. Depending on the details entered upto this point, a dialog box prompts for frequency specifications such as band gain, ripple ratio, starting and ending frequencies of the bands, and band specification type as shown in Figure 6.

By selecting the Optimization option in the Design menu, a radio box pops up which prompts the user to select any one of the six discrete space optimization control options as shown in Figure 7. Here again the default choice is the minimal control option. Depending on the option selected, some of the parameters such as

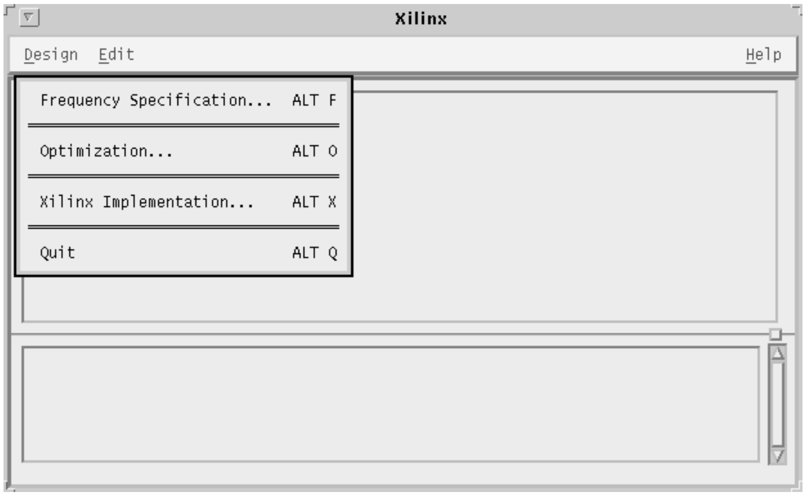
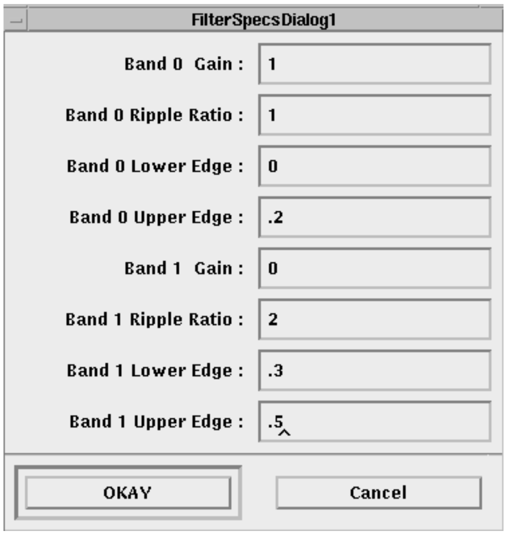


Figure 5: Main Window of the CAD Tool showing the various options in the Design Menu



Band 0 Gain : 1
Band 0 Ripple Ratio : 1
Band 0 Lower Edge : 0
Band 0 Upper Edge : .2
Band 1 Gain : 0
Band 1 Ripple Ratio : 2
Band 1 Lower Edge : .3
Band 1 Upper Edge : .5

Figure 6: Dialog box prompting for frequency specification for different bands

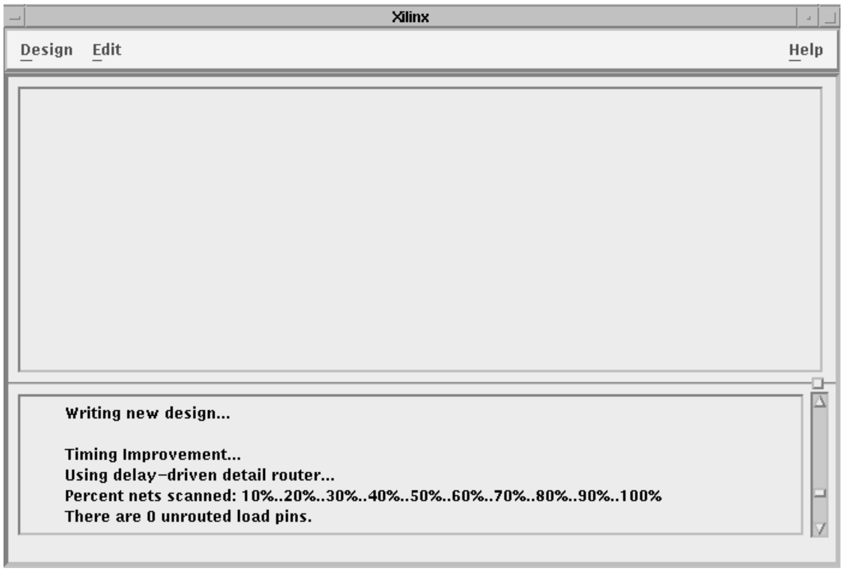
Figure 7: Radio Box for selecting one of the various Discrete optimization control options

passband gain weight, upper and lower limits of passband gain, objective function values,and maximum allowable coefficient value need to be specified in the Dialog box which follows. The output of this Design option is a set of filter coefficients in the discrete power-of-two space.

Once the optimization is done, the Xilinx Implementation option of the Design menu is selected. The input of this stage is either a set of user specified filter coefficients or the output of the optimization stage. The pin constraints can be specified in a dialog box or in a file. The Automatic Place and Route (APR) program typically requires 10 - 15 minutes for routing this type of FIR filter implementation. Messages are displayed in the message window regarding the progress of APR, as shown in Figure 8. The configuration details are output in a .lca file which can be used to determine the delay characteristics using the XACT^{TM} tools [6].

5 Performance

With the Xilinx XC3195, which has an array of 22 by 22 (484) CLBs, the maximum intermediate wordlength that can be accomodated is 22 bits. As a rule of thumb, the intermediate wordlength can be taken as slightly more than twice the number of input data bits. Hence, the maximum number of input data bits that can be supported is 10 bits. As each tap requires two columns of CLBs, up to 11 taps can be realized per chip.



```
Writing new design...

Timing Improvement...
Using delay-driven detail router...
Percent nets scanned: 10%..20%..30%..40%..50%..60%..70%..80%..90%..100%
There are 0 unrouted load pins.
```

Figure 8: Automatic Place and Route in progress - showing messages in the message window

Typical filter characteristics have been implemented on a Xilinx XC3195 FPGA using this tool. One of these is an eleven tap lowpass FIR filter (filter #0) with passband cut- off at $0.1f_s$, stop band beginning at $0.15f_s$ and -18 dB stopband rejection. Another example is an eleven tap highpass filter (filter #1) with the cut-off frequency at $0.1f_s$, the passband beginning at $0.15f_s$, and -18db stopband rejection. The final example is an eleven tap lowpass FIR filter (filter #2) with the passband cut-off at $0.2f_s$, the stopband beginning $0.3f_s$, and -27dB stopband rejection. The discrete-space impulse responses shown in Table 1 have been designed and implemented using this tool. An input data word size of 10 bits was used for all the examples; the 22 rows provide sufficient intermediate word width protection against overflow. All of the columns of the array were required to implement eleven taps. The final accumulation stage was not performed on the array. The sampling speeds of these filters attained using various mapping techniques on the present and the previous structures are listed in Table 2. The present structure with input pins and horizontal lines optimally assigned by APR gives the best performance with maximum sampling rates of 33.3 MHz, 31.8 MHz and 31.3 MHz. If placement is done by APR alone, with no prior placement, then the sampling speed attained for filter #0 is only 25.0 MHz.

The layout of the low pass filter mentioned above on an XC3195 is shown in

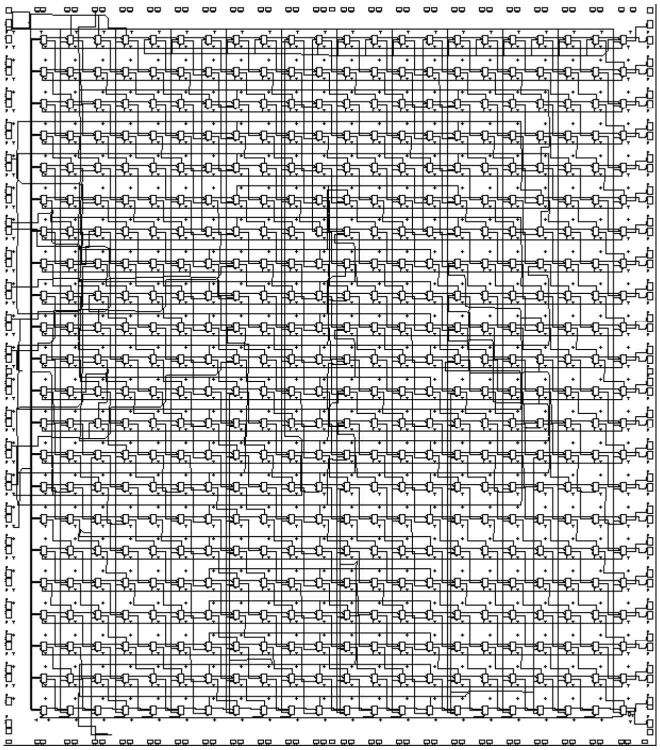


Figure 9: Layout of the Low Pass Filter on XC 3195 FPGA

Figure 9.

It is not cost effective to provide the final accumulation stage on the chip for the XC3000 and XC3100 series devices. A dedicated parallel adder can be used for that purpose. It is possible to implement the final adder stage in the XC4000 series of FPGAs, however, by virtue of the fast carry logic supported by these devices.

6 Conclusion

A CAD system for design and efficient implementation of FIR filters on Xilinx Field Programmable Gate Arrays was presented. Several generalized techniques which were used to reduce delay have been described and their effects on performance were evaluated. The present structure with the optimal assignment of long lines is found to be the best, but several other heuristic techniques can be used to obtain quality routings, better than those that have been previously proposed. Examples of typical filters designed and implemented using this tool were shown.

Table 1: Example FPGA Filter Impulse Responses

filter taps	filter #0	filter #1	filter #2
w_0, w_{10}	-30	12	36
w_1, w_9	6	7	-4
w_2, w_8	24	-33	-33
w_3, w_7	48	1	-56
w_4, w_6	65	96	-80
w_5	72	72	160

Table 2: Sampling speed (in MHz) attained for different techniques

Filter	*Previous Structure*		*Present Structure*			
	Unoptimized	Optimized	Unoptimized	Optimized	Alternate	Center - Alternate
filter #0	23.2	28.7	25.1	33.3	31.5	29.9
filter #1	26.3	27.3	26.4	31.3	30.7	30.7
filter #2	20.0	30.2	26.6	31.8	27.8	30.8

Acknowledgement

We would like to express our thanks to Professor Y.C. Lim of the National University of Singapore for the use of his MILP3 program in this work.

References

[1] J. B. Evans. An efficient FIR filter architecture. In *IEEE Int. Symp. Circuits and Syst.*, pages 627–630, May 1993.

[2] Y. C. Lim and S. R. Parker. FIR filter designed over a discrete power-of-two coefficient space. *IEEE Trans. Acoust., Speech, Signal Processing*, ASSP-31:583–591, Nov 1983.

[3] Y. C. Lim and A. G.Constantinides. Linear phase FIR digital filter without multipliers. In *IEEE Int. Symp. Circuits and Syst.*, pages 185–188 , 1979.

[4] Y. C. Lim and B. Liu. Design of cascade form FIR filters with discrete valued coefficients. *IEEE Trans. Acoust., Speech, Signal Processing*, ASSP-36:1735–1739, Nov 1988.

[5] Y. C. Lim. MILP3 Manual. University of Singapore., Mar 1988.

[6] XACT Reference Guide. Vol 1–2. Xilinx Incorporated. San Jose, California. 1992.

[7] The Programmable Gate Array Data Book. Xilinx Incorporated. San Jose, California. 1993.

[8] S. Y. Kung. *VLSI Array Processors*. Prentice-Hall, 1988.

[9] S. Y. Kung, H. J. Whitehouse, and T. Kailath, editors. *VLSI and Modern Signal Processing*. Prentice-Hall, Inc., 1985.

[10] A. Oppenheim and R. Schafer. *Digital Signal Processing*. Prentice-Hall, Inc., 1975.

[11] P. R. Cappello, editor. *VLSI Signal Processing*. IEEE Press, 1984.

[12] S. Y. Kung, R. E. Owen, and J. G. Nash, editors. *VLSI Signal Processing II*. IEEE Press, 1986.