Optimization of Signal Processing Chains: Application to Cascaded Filters

A. Hugeat,* J. Bernard,* J.-M. Friedt,* P.-Y. Bourgeois,* G. Goavec-Merou*
*FEMTO-ST Institute
Univ. de Bourgogne Franche-Comté, CNRS
Besançon, France

***Abstract*—The design of digital signal processing chains must meet competing requirements by maximizing performance (e.g. rejection in the case of a filter) while reducing resource consumption. In this paper, we explore a new methodology for designing chains assembled by cascading basic processing blocks. We apply this optimization strategy to the example of a cascade of Finite Impulse Response (FIR) filters. While the design of cascaded FIR filters generally focuses on low-level details, we provide a high-level model. This development strategy can be generalized for any signal processing chain made by assembling blocks whose resource consumption is qualified: a solver aims at meeting multiple objectives including minimizing resource consumption or optimizing performance. This result is then transformed into a synthesizable solution targeting a reconfigurable Field Programmable Gate Array (FPGA). The experiments show that this approach gives efficient results, both on the quality of the signal filtering and the processing resource used for the design.**

***Index Terms*—Field Programmable Gate Array, Finite Impulse Response filter, Optimization**

I. Introduction

Optimizing a software defined radio digital signal processing chain can be a complex problem. Each functional block in a chain can be optimized separately in order to aim for a global benefit. But this approach can show some limits. As a practical demonstration of this concept, we address Finite Impulse Response (FIR) filters as essential part of every digitial signal processing frontend responsible for low-pass filtering the signal before a decimation. Using multiple FIR filters may be more efficient than a single FIR filter [1].

The aim of this paper is to model resource usage of filter cascades and attempt to optimize parameters such as coefficient resolution (bits in an integer representation), number of coefficients and number of filters to maximize cutoff rejection in a given processing resource environment. We select FIR filters for their unconditional stability, despite the larger number of coefficients with respect to the Infinite Impulse Response (IIR) filters, introducing a longer lag which does not affect the processing bandwidth but allows for systematic analysis of cascaded filter chain. The result of our optimization is then implemented on a FPGA in order to test and verify that the identified solution is valid.

Section II describes background works related to FIR filters design and optimization. Section III defines our model of a cascaded filter thanks to a mixed integer quadratic program. Finally, section IV shows the results of our experiments.

II. Hardware Resources Design and Optimization

Despite increasing general purpose central processing unit computational power, real time radiofrequency signal processing remains the realm of FPGA aiming at pipelined processing data streams generated by current high grade analog to digital converters running at several hundreds of Msamples per second. While FPGA size and speed keep on increasing, so do the requirements of the user, with increasing rejection of filters or processing bandwith ambitions. Hence, optimizing available resources for a pipelined processing is needed and has been readily investigated by some authors.

A very interesting approach aimed at improving the reliability and development time consists in using the skeleton methodology, as described by Benkrid *et al.* [2]. This methodology aims at creating small elementary blocks that are generic and easily configurable. These blocks can be used by different projects, improving their quality and reliability over time. Then Benkrid *et al.* [3], [4] created a formal language (HIDE) to describe the placement of blocks inside the FPGA to optimize the occupied area. This language is derived from the Prolog language and describes how the skeleton blocks can be connected. Once the whole processing chain has been described, HIDE uses native built-in features of Prolog to seek an optimized placement. The placement optimization improves the resource consumption. Compared to our approach, this work only focuses on placement optimization, which is rather low-level and forgets about quality of the resulting chain.

Radiofrequency datastream digital processing is best improved by focusing on the FIR filters used in the frontend. Cen *et al.* [5] and Dey *et al.* [6] both use a genetic algorithm to design the FIR coefficients. They take the wanted shape of the frequency response and, generation after generation, have a coefficient set which better fits the ideal response. With our approach, we do not design the filters but consider sets of coefficients generated from a least square fitting approach or similar filter design tool. The free parameters are the cutoff frequency, rejection and quantization resolution in the context of computations restricted to integers: a collection of FIR filter with varying parameter is hence assembled and each element qualified in terms of cutoff band rejection. The aim is to globally optimize the problem rather than addressing each FIR individually.

Some works propose to improve the FIR algorithm. For

978-9-0827-9703-9/19/$31.00

example, Samueli *et al.* [7] explain that taking only power-of-two coefficients in the FIR allows to replace the Digital Signal Processing (DSP) multiplier by a computationnaly efficient bit shift. The speed is greatly improved but there is a hard constraint on the coefficients. Another example is given by Tsao *et al.* [8]. This time, the authors take advantage of the mathematical symmetry of convolution to substitute some multipliers by some adders. Both of these solutions are again focused on low-level hardware optimization.

Lim *et al.* [1] worked on the efficiency of multiple filters instead of one filter. Their research is focused on the impact of ripples on passband when they use a cascaded filter. To limit this impact they developed a Mixed Intger Linear Program (MILP) to search the best filter coefficients. Although their method aims at designing filter coefficients, it proves that we can use a cascade of filters to save some computation time or resources. Later Lim *et al.* improved their work on cascaded filters [9]. They developped another linear program to design two filters to reduce the ripples compared to a unique filter solution.

Young and Jones [10] proposed another MILP to design a cascaded filter. They considered the problem of resource consumption for ASIC design. Moreover, they also considered the rejection on the stopband. The limitation of their work is the time to solve the MILP et the scalability of the approach. Indeed, they were not able to create a solution for more than two sub-filters. Despite this limit, the approach confirms the relevance of multiple filters to improve the stopband rejection.

All these papers exhibit interesting optimizations to design FIR filters. In contrast, our work aims at providing a generic methodology to optimize any signal processing chain. We aim at a more abstract model of the blocks, and in the case of a cascasded FIR filter, of a simple FIR. The next section shows how to design a cascaded FIR filter with our methodology.

III. Model for Cascaded Filters

In order to get an efficient global design, our methodology is based on two steps. First, we create a model for each block, with high-level parameters, abstracting totally the implementation details. Second, we define the constraints and the objective as an optimization problem. Finally we use a solver to obtain an optimal theoretical result.

In this section, we apply this methodology to the problem of the cascade of FIR filters.

A. Model of a Filter

The cascade is composed of n stages. In stage i ($1 \leq i \leq n$), the filter is composed of a FIR and a shifter. The FIR has C_i coefficients of size π_i^C bits. Equation 1 defines the computation of output data (y) as a weighted average of input data (x). This equation is a discrete convolution. The shifter does a right shift of size π_i^S bits. The filter takes input data of size π_i^- bits and produces output data of size π_i^+. Figure 1 shows one stage of the cascade.

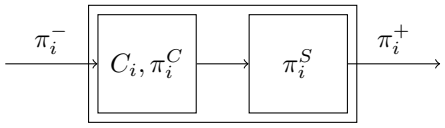

Fig. 1. A single filter is composed of a FIR (on the left) and a Shifter (on the right)

$$y_n = \sum_{i=0}^{C_i-1} \underbrace{b_i}_{\pi_i^C \text{ bits}} \times \underbrace{x_{n-i}}_{\pi_i^- \text{ bits}} \tag{1}$$

FIR i can reject $F(C_i, \pi_i^C)$ dB. F is determined numerically. To measure this rejection, we use GNU Octave software to design FIR filter coefficients thanks to two algorithms (`firls` and `fir1`). For each configuration (C_i, π_i^C), we first create a FIR with floating point coefficients and a given C_i number of coefficients. Then, the floating point coefficients are discretized into integers. In order to ensure that the coefficients are coded on π_i^C bits effectively, the coefficients are normalized by their absolute maximum before being scaled to integer coefficients. At least one coefficient is coded on π_i^C bits, and in practice only $b_{C_i/2}$ is coded on π_i^C bits while the other are coded on very fewer bits.

With these coefficients, the `freqz` function is used to estimate the magnitude of the transfer function of the filter. We consider the maximum rejection within the stopband minus the mean of the absolute value of passband rejection as efficiency criterion. This criterion gives us the value of $F(C_i, \pi_i^C)$ for each configuration.

B. Problem Description

The problem we address is to maximize the rejection under bounded silicon area and feasability constraints. Variable a_i is the area taken by filter i (in arbitrary unit). Variable r_i is the rejection of filter i (in dB). Constant $\mathcal{A}$ is the total available area. We model our problem as follows:

$$\text{Maximize} \sum_{i=1}^{n} r_i$$

$$\sum_{i=1}^{n} a_i \leq \mathcal{A} \tag{2}$$

$$a_i = C_i \times (\pi_i^C + \pi_i^-), \qquad \forall i \in [1, n] \tag{3}$$

$$r_i = F(C_i, \pi_i^C), \qquad \forall i \in [1, n] \tag{4}$$

$$\pi_i^+ = \pi_i^- + \pi_i^C - \pi_i^S, \qquad \forall i \in [1, n] \tag{5}$$

$$\pi_{i-1}^+ = \pi_i^-, \qquad \forall i \in [2, n] \tag{6}$$

$$\pi_i^+ \geq 1 + \sum_{k=1}^{i} \left(1 + \frac{r_j}{6}\right), \qquad \forall i \in [1, n] \tag{7}$$

$$\pi_1^- = \Pi^I \tag{8}$$

Equation 2 states that the total area taken by the filters must be less than the available area. Equation 3 gives the definition

of the area for a filter. More precisely, it is the area of the FIR as the Shifter does not need any circuitry. We consider that the FIR needs C_i registers of size $\pi_i^C + \pi_i^-$ bits to store the results of the multiplications of the input data and the coefficients. Equation 4 gives the definition of the rejection of the filter thanks to function F that we defined previously. The Shifter does not introduce negative rejection as we explain later, so the rejection only comes from the FIR. Equation 5 states the relation between π_i^+ and π_i^-. The multiplications in the FIR add π_i^C bits as most coefficients are close to zero, and the Shifter removes π_i^S bits. Equation 6 states that the output number of bits of a filter is the same as the input number of bits of the next filter. Equation 7 ensures that the Shifter does not introduce negative rejection. Indeed, the results of the FIR can be right shifted without compromising the quality of the rejection until a threshold. Each bit of the output data increases the maximum rejection level of 6 dB. We add one to take the sign bit into account. If equation 7 was not present, the Shifter could shift too much and introduce some noise in the output data. Each supplementary shift bit would cause 6 dB of noise. A totally equivalent equation is: $\pi_i^S \leq \pi_i^- + \pi_i^C - 1 - \sum_{k=1}^{i} \left(1 + \frac{r_j}{6}\right)$. Finally, equation 8 gives the global input's number of bits.

This model is non-linear and even non-quadratic, as F does not have a known linear or quadratic expression. We introduce p FIR configurations $(C_{ij}, \pi_{ij}^C), 1 \leq j \leq p$ that are constants. We define binary variable δ_{ij} that has value 1 if stage i is in configuration j and 0 otherwise. The new equations are as follows:

$$a_i = \sum_{j=1}^{p} \delta_{ij} \times C_{ij} \times (\pi_{ij}^C + \pi_i^-), \quad \forall i \in [1, n] \tag{9}$$

$$r_i = \sum_{j=1}^{p} \delta_{ij} \times F(C_{ij}, \pi_{ij}^C), \quad \forall i \in [1, n] \tag{10}$$

$$\pi_i^+ = \pi_i^- + \left(\sum_{j=1}^{p} \delta_{ij} \pi_{ij}^C\right) - \pi_i^S, \quad \forall i \in [1, n] \tag{11}$$

$$\sum_{j=1}^{p} \delta_{ij} \leq 1, \quad \forall i \in [1, n] \tag{12}$$

Equations 9, 10 and 11 replace respectively equations 3, 4 and 5. Equation 12 states that for each stage, a single configuration is chosen at most.

This modified model is quadratic, and it can be linearised if necessary. The Gurobi optimization software is used to solve this quadratic model, and since Gurobi is able to linearize, the model is left as is. This model has $O(np)$ variables and $O(n)$ constraints.

Note that we could also define a complementary problem: minimize the silicon area under a bounded rejection and feasability constraints. In this case, the objective function would be to minimize $\sum_{i=1}^{n} a_i$. Equation 2 would be replaced by equation 13 where $\mathcal{R}$ would be the guaranteed rejection.

TABLE I
CONFIGURATIONS (C_i, π_i^C, π_i^S), REJECTIONS AND AREAS (IN ARBITRARY UNITS) FOR PRN/500

n	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	Rejection	Area
1	(21, 7, 0)	-	-	-	-	32 dB	483
2	(3, 3, 15)	(33, 9, 0)	-	-	-	58 dB	486
3	(3, 3, 15)	(31, 8, 0)	(3, 3, 0)	-	-	67 dB	477
4	(3, 3, 15)	(19, 7, 2)	(11, 5, 2)	(3, 3, 0)	-	74 dB	496
5	(3, 3, 15)	(23, 8, 3)	(3, 3, 0)	(3, 3, 0)	(3, 3, 0)	78 dB	486

$$\sum_{i=1}^{n} r_i \geq \mathcal{R} \tag{13}$$

IV. EXPERIMENTAL RESULTS

To produce the following results, we have developed a workflow as described in this section. From a host PC, we solve the quadratic program and we generate a FPGA design with Xilinx Vivado software (version 2018.2). The synthesized bitstream is used to configure the Programmable Logic (PL) of the Xilinx Zynq 7010 SOC (xc7z010clg400-1) provided by a Redpitaya board under supervision of GNU/Linux running on the Processing System (PS). The response of the filter is addressed both on synthetic signals by feeding the FIR with the output of a pseudo-random (20-bit long linear feedback shift register) number generator. Finally, on the host PC fetches filtered datasets, the resulting transfer functions are normalized to obtain the Power Spectrum Density (PSD) aligned to zero and the performances of the various filters are compared by assessing the rejection following the criteria detailed earlier.

This section describes the experimental results obtained with the previous workflow. The goal is to check whether the results given by the filter solver are accurate enough in an experimental context *i.e.* the model of the filter is good enough to give solid results in terms of rejection as well as in terms of silicon area usage.

The experimental setup is composed of three cases. The raw input is generated by a Pseudo Random Number (PRN) generator, which fixes the input data size Π^I. Then the total silicon area $\mathcal{A}$ has been fixed to either 500, 1000 or 1500 arbitrary units. Hence, the three cases have been named: PRN/500, PRN/1000, PRN/1500. The number of configurations p is 1827, with C_i ranging from 3 to 60 and π^C ranging from 2 to 22. In each case, the quadratic program has been able to give a result up to five stages ($n = 5$) in the cascaded filter.

A. Comparison of Solver Results

This section presents the output of the filter solver *i.e.* the computed configurations for each stage, the computed rejection and the computed silicon area. This is interesting to understand the choices made by the solver to compute its solutions.

Table I shows the results obtained by the filter solver for PRN/500. Table II shows the results obtained by the filter solver for PRN/1000. Table III shows the results obtained by the filter solver for PRN/1500.

TABLE II
CONFIGURATIONS (C_i, π_i^C, π_i^S), REJECTIONS AND AREAS (IN ARBITRARY UNITS) FOR PRN/1000

n	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	Rejection	Area
1	(37, 11, 0)	-	-	-	-	56 dB	999
2	(3, 3, 15)	(51, 14, 0)	-	-	-	87 dB	975
3	(3, 3, 15)	(35, 11, 2)	(19, 7, 0)	-	-	99 dB	1000
4	(3, 4, 16)	(27, 8, 2)	(19, 7, 3)	(11, 5, 0)	-	103 dB	998
5	(3, 3, 15)	(31, 9, 2)	(19, 7, 2)	(3, 3, 0)	(3, 3, 0)	111 dB	987

TABLE III
CONFIGURATIONS (C_i, π_i^C, π_i^S), REJECTIONS AND AREAS (IN ARBITRARY UNITS) FOR PRN/1500

n	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	Rejection	Area
1	(47, 15, 0)	-	-	-	-	71 dB	1457
2	(19, 6, 17)	(51, 14, 0)	-	-	-	103 dB	1489
3	(3, 3, 15)	(35, 11, 2)	(35, 11, 0)	-	-	122 dB	1492
4	(3, 3, 15)	(27, 8, 2)	(27, 9, 2)	(19, 7, 0)	-	129 dB	1480

From these tables, we can first state that the more stages are used to define the cascaded FIR filters, the better the rejection. It was an expected result as it has been previously observed that many small filters are better than a single large filter [1], [9], [10], despite such conclusion being hardly used in practice due to the lack of tools for identifying individual filter coefficients in the cascaded approach.

Second, the larger the silicon area, the better the rejection. This was also an expected result as more area means a filter of better quality (more coefficients or more bits per coefficient).

Then, we also observe that the first stage can have a larger shift than the other stages. This is explained by the fact that the solver tries to use just enough bits for the computed rejection after each stage. In the first stage, a balance between a strong rejection with a low number of bits is targeted. Equation 7 gives the relation between both values.

Finally, we note that the solver consumes all the given silicon area.

B. Comparison of Filters

This section presents the rejection for real data on the FPGA. In all the figures of this section, the solid line represents the actual rejection of the filtered data on the FPGA as measured experimentally. The configurations are those computed in the previous section.

Figure 2 shows the rejection of the different configurations in the case of PRN/500. Figure 3 shows the rejection of the different configurations in the case of PRN/1000. Figure 4 shows the rejection of the different configurations in the case of PRN/1500.

In all cases, we observe that the actual rejection is close to the rejection computed by the solver.

C. Comparison of Resource Usage

In this section, we compare the actual silicon resources given by Vivado to the resources in arbitrary units. The goal is to check that our arbitrary units of silicon area models well enough the real resources on the FPGA. Especially we want

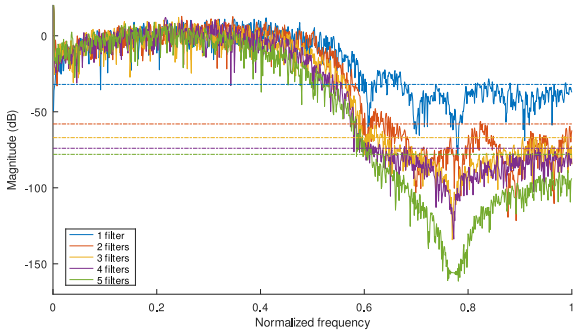

Fig. 2. Signal spectrum for PRN/500

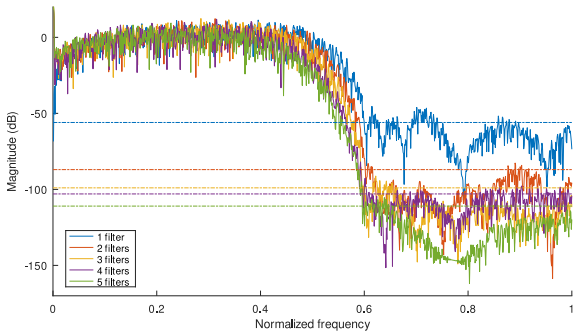

Fig. 3. Signal spectrum for PRN/1000

to verify that, for a given number of arbitrary units, the actual silicon resources do not depend on the number of stages n. Most significantly, our approach aims at remaining far enough from the practical logic gate implementation used by various vendors to remain platform independant and be portable from one architecture to another.

Table IV shows the resources usage in the case of PRN/500, PRN/1000 and PRN/1500 *i.e.* when the maximum allowed silicon area is fixed to 500, 1000 and 1500 arbitrary units. We have taken care to extract solely the resources used by the FIR filters and remove additional processing blocks including FIFO and PL to PS communication.

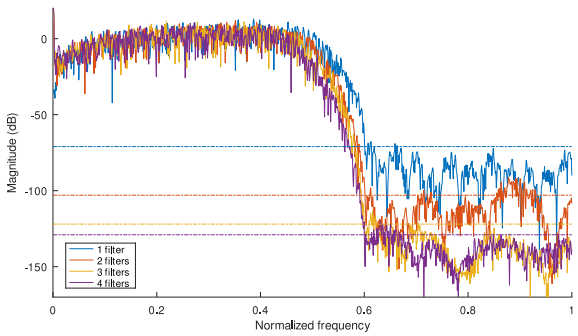

Fig. 4. Signal spectrum for PRN/1500

TABLE IV
RESOURCE OCCUPATION. THE LAST COLUMN REFERS TO AVAILABLE RESOURCES ON A ZYNQ-7010 AS FOUND ON THE REDPITAYA.

n		PRN/500	PRN/1000	PRN/1500	*Zynq 7010*
1	LUT	261	476	653	*17600*
	BRAM	1	1	1	*120*
	DSP	21	37	47	*80*
2	LUT	2595	5632	718	*17600*
	BRAM	2	2	2	*120*
	DSP	0	0	70	*80*
3	LUT	2495	3371	3616	*17600*
	BRAM	3	3	3	*120*
	DSP	0	19	35	*80*
4	LUT	2660	3830	2603	*17600*
	BRAM	4	4	4	*120*
	DPS	0	19	46	*80*
5	LUT	2456	3319	-	*17600*
	BRAM	5	5	-	*120*
	DPS	0	19	-	*80*

TABLE V
TIME TO SOLVE THE QUADRATIC PROGRAM WITH GUROBI

n	Time (PRN/500)	Time (PRN/1000)	Time (PRN/1500)
1	0.1 s	0.2 s	0.3 s
2	1.2 s	1.8 s	14.1 s
3	46 s	67 s	264 s ($\approx$ 4 min)
4	60 s	3888 s ($\approx$ 64 min)	5505 s ($\approx$ 1 h 30)
5	473 s ($\approx$ 8 min)	6268 s ($\approx$ 104 min)	*stopped after 40 h*

In some cases, Vivado replaces the DSPs by Look Up Tables (LUTs). We assume that, when the filters coefficients are small enough, or when the input size is small enough, Vivado optimized resource consumption by selecting multiplexers to implement the multiplications instead of a DSP. In this case, it is quite difficult to compare the whole silicon budget.

However, a rough estimation can be made with a simple equivalence. Looking at the first column (PRN/500), where the number of LUTs is quite stable for $n \geq 2$, we can deduce that a DSP is roughly equivalent to 100 LUTs in terms of silicon area use. With this equivalence, our 500 arbitraty units corresponds to 2500 LUTs, 1000 arbitrary units corresponds to 5000 LUTs and 1500 arbitrary units corresponds to 7300 LUTs. The conclusion is that the orders of magnitude of our arbitrary unit are quite good. The relatively small differences can probably be explained by the optimizations done by Vivado based on the detailed map of available processing resources.

D. Comparison of Computation Times

In this section we present the computation time to solve the quadratic problem. For each case, the filter solver software are executed with a Intel(R) Xeon(R) CPU E5606 cadenced at 2.13 GHz. The CPU has 8 cores that are used by Gurobi to solve the quadratic problem.

Table V shows the time needed to solve the quadratic problem when the maximal area is fixed to 500, 1000 and 1500 arbitrary units.

As expected, the computation time seems to rise exponentially with the number of stages. When the area is limited, the design exploration space is more limited and the solver is able to find an optimal solution faster. On the contrary, in the case of PRN/1500 with 5 stages, we were not able to obtain a result after 40 hours of computation so we decided to stop.

V. CONCLUSION

We have proposed a new approach to work with a cascade of FIR filter inside a FPGA. We have modeled the FIR filter operation and the data shift impact. With this model we have created a quadratic program to select the optimal FIR coefficient set to reject a maximum of noise.

Our experimental results are very promising in providing a rational approach to selecting the coefficients of each FIR filter in the context of a performance target for a chain of such filters. The FPGA design that is produced automatically by our workflow is able to filter an input signal as expected which validates our model and our approach.

A perspective is to model and add the decimators to the processing chain to have a classical FIR filter and decimator. The impact of the decimator is not so trivial, especially in terms of silicon area for the subsequent stages.

All The software used to demonstrate the concepts developed in this paper is based on the CPU-FPGA co-design framework available on OscIMP Digital github repository.

The project is supported by a Région Franche-Comté grant. The development of the software framework is supported by the OscillatorIMP ANR Equipex grant.

REFERENCES

[1] Y. C. Lim and B. Liu, "Design of cascade form fir filters with discrete valued coefficients," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 11, pp. 1735–1739, Nov 1988.

[2] K. Benkrid, D. Crookes, J. Smith, and A. Benkrid, "High level programming for fpga based image and video processing using hardware skeletons," in *The 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'01)*, March 2001, pp. 219–226.

[3] K. Benkrid, A. Benkrid, and S. Belkacemi, "Efficient fpga hardware development: A multi-language approach," *Journal of Systems Architecture*, vol. 53, no. 4, pp. 184 – 209, 2007.

[4] K. Benkrid, S. Belkacemi, and A. Benkrid, "Hide: A hardware intelligent description environment," *Microprocessors and Microsystems*, vol. 30, no. 6, pp. 283 – 300, 2006, special Issue on FPGA's.

[5] L. Cen, "A hybrid genetic algorithm for the design of fir filters with spot coefficients," *Signal Processing*, vol. 87, no. 3, pp. 528 – 540, 2007.

[6] A. Dey, S. Susmita, S. Avijit, and G. Shibani, "A method of genetic algorithm (ga) for fir filter construction: Design and development with newer approaches in neural network platform," vol. 1, 01 2011.

[7] H. Samueli, "An improved search algorithm for the design of multiplierless fir filters with powers-of-two coefficients," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 7, pp. 1044–1047, July 1989.

[8] Y. Tsao and K. Choi, "Area-efficient parallel fir digital filter structures for symmetric convolutions based on fast fir algorithm," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 2, pp. 366–371, Feb 2012.

[9] Y.-C. Lim, R. Yang, and B. Liu, "The design of cascaded fir filters," in *1996 IEEE International Symposium on Circuits and Systems. Circuits and Systems Connecting the World. ISCAS 96*, vol. 2, May 1996, pp. 181–184 vol.2.

[10] C. Young and D. L. Jones, "Improvement in finite wordlength fir digital filter design by cascading," in *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, March 1992, pp. 109–112 vol.5.