
Rajeev Jain, Member, IEEE, Paul T. Yang, and Toshiaki Yoshino, Member, IEEE

Abstract—This paper describes automatic architecture and floorplan generation techniques for integrated circuit fixed-coefficient FIR filters that can achieve high sample rates with compact layouts. These techniques have been implemented in a filter design system called FIRGEN that can automate the entire design from filter specifications to final chip layout. It can be retargeted to new cell libraries and place and route tools. Results of four chips designed with FIRGEN are presented. These achieve sample rates ranging from 25 to 112 MHz.

I. INTRODUCTION

HIGH performance FIR filters have applications in several video processing [1] and digital communications systems [2]. While compiler tools exist for low sample rate applications such as audio and telecommunication, techniques for automating the design of high sample rate FIR filters have only recently been emerging [3], [4]. These techniques have been shown useful for sample rates of 10–30 MHz. However, methods for higher rates such as required in high-speed digital data communications [2] or high rate video transmission systems [5] have not yet been reported. In this paper a new CAD system, FIRGEN, is presented that has been successfully used to generate compact FIR filter circuits operating at sample rates in the 25–112 MHz range in relatively mature technologies.

FIRGEN allows the user to generate the chip layout starting from filter specifications or from coefficient values. It consists of programs for a) filter synthesis, optimization, and simulation; b) architecture generation; c) floorplan generation; and d) chip layout generation. The architecture generation techniques are based on ideas proposed in [4], [6], [7], and the floorplanning techniques are based on results reported in [7]–[9]. The architecture and floorplanning generation tools are specifically aimed at achieving very high sample rates with compact layout and minimal pipeline latency.

Relative to existing filter compiler tools FIRGEN provides four advanced features: a) high performance with relatively compact layouts; b) the capability to retarget the design system to new technologies and associated layout tools; c) a framework for integrating filter synthesis, simulation, architecture generation, and layout generation tools; and d) compatibility with custom as well as semi-custom technologies. The CAD system can be retargeted to a user specified technology. Currently FIRGEN has been linked to two technologies: 1) a MOSIS compatible scalable (2.0-μm and 1.2-μm) CMOS custom macrocell library using the Lager layout generation tools [10] and 2) a commercial (0.8-μm) BiCMOS gate array cell library using the Tangata sea-of-gate layout generation tools [11]. Chips fabricated with these technologies have been presented in [12], [13].

A. CAD Tool Integration

Conventional filter compiler tools [1], [14]–[16] are monolithic programs which combine routines for architecture generation and layout generation into one package. This makes it difficult to retarget the compiler to different technologies and the tool becomes obsolete very quickly due to rapid advances in IC technology. To solve this problem FIRGEN has been developed as an integrated CAD system (Fig. 1) rather than a compiler tool. Each component can be replaced without having to change the other components. Clean interfaces have been defined between the filter synthesis, simulation, architecture generation, floorplan generation, and layout generation tools.

This approach is different from the conventional compilers that hard code the cell library information and the chip floorplan and use dedicated place and route routines. An overview of the CAD system is given in Section II.

B. Architectural Strategy

The bit-serial compilers FIRST [14], Impact [15], and Cathedral I [16] are applicable to both FIR and IIR filters but are mostly useful for low sample rate filters such as in speech, audio, and telecommunication applications [17]. Microcoded bit-parallel compilers [18], [19] have also been used for these low sample rate applications. Several techniques have been proposed for achieving higher sample rates in FIR filter circuits. These require varying amounts of hardware complexity and in many cases rely on heavily pipelining the architecture since pipeline latency can be introduced in FIR structures without affecting the input/output transfer function. However, the use of indiscriminate pipelining leads to substantial
increases in power dissipation as well as chip area due to the pipeline registers [4], [20]. The increase in power dissipation is of special concern at high speeds. The aim in developing FIRGEN has been to achieve high sample rates with minimum possible use of pipelining.

The compiler presented by Cappello et al. [21] achieves high performance by block processing of the input samples with highly parallel architectures. This is a powerful approach if the user is willing to invest substantial hardware complexity to achieve sample rates that are in excess of the maximum clock rates for the multiply-add circuits. The techniques proposed in this paper are aimed at achieving sample rates that are the same as the maximum clock rates without adding more hardware than is normally required by a one-to-one mapping of the algorithmic operators into hardware.

The FIR filter compiler Parsifal [3] offers the user the choice of combining multiple digit-serial data paths to achieve performance that is higher than that of pure bit-serial structures but lower than that of full bit-parallel architectures. The idea is to offer a spectrum of tradeoffs between bit-serial and bit-parallel architectures. In contrast FIRGEN is optimized for hardwired bit-parallel architectures only to get maximum performance for a given technology.

The FIR filter compiler proposed by Privat [1] is also targeted at hardwired bit-parallel architecture but uses carry-ripple addition which offsets the advantage of bit-parallel processing. Ruetz [4] also uses bit-parallel architecture but uses carry-save addition and additionally employs a tree structure (Fig. 5) for accumulating the tap products to reduce the critical path delay through the carry-save adders. A further reduction in critical path can be achieved by using carry-save bit-parallel processing together with the transpose form (Fig. 7) instead of the direct form (Fig. 4). Ulbrich and Noll [6] have proposed such an architecture with both carry-save addition and a fast vector-merge adder to get full advantage of bit-parallel processing, and Lin and Samueli [7] have shown a bit-level pipelined version of such an architecture. This is the approach used in FIRGEN and will be described in detail in Section III.

C. High Performance Layout Requirements

While most work reported on FIR filter design tools pays a lot of attention to automating high-speed architecture design for the multiply-add operations, there is less mention of designing the associated clock and data distribution network. Furthermore, less effort has been put into automation of floorplan and layout design for high performance. This is not a significant problem if the filter generator is tied into a specific technology and cell library. In this case the floorplan and layout generation are usually hard-coded into the compiler tool. However, if it is desired to retarget the design tools to arbitrary technologies, it is important to automate generation of the floorplan to direct the technology-specific layout tools.

In particular, to achieve high performance, the design and layout of data and clock distribution networks is very critical. Hatamian [22] and Noll [9] have reported design strategies for clock distribution for high performance. The floorplan generation tools in FIRGEN make use of these ideas. The floorplan generation techniques in FIRGEN, described in Section IV, have been used with custom layout [13] as well as gate-array layout tools [12]. In Appendix A, the cell library required by FIRGEN is presented and in Section V, the chip testing results are presented.

II. OVERVIEW OF FILTER DESIGN SYSTEM

The filter design system FIRGEN consists of three major subsystems: a) filter synthesis and coefficient quantization; b) filter architecture and floorplan generation; c) chip layout generation. Details of each subsystem are given below. Fig. 1 provides an overview of the entire filter generation process.

In the first step the user determines the filter coefficient values using the filter synthesis, simulation, and optimization software. The coefficient values are then entered in the architecture and floorplan generation programs. Both programs obtain information about the circuit cells from the cell library database using a mapping function. The architecture generator provides a netlist of the library cells to implement the filter. The floorplan generator creates a floorplan describing placement of the cells in the layout. The netlist and floorplan information is entered into the
layout generation tool (for the desired cell library) to obtain the final layout.

A. Filter Synthesis, Simulation, and Optimization

Frequency domain design specifications can be entered for low-pass, high-pass, bandpass, or arbitrary multiband filters. The specifications are entered interactively through a command window (Fig. 2). The user can then choose to use either the synthesis tool FIR [23] or WINDOW [24] to obtain floating point values for the filter tap coefficients. FIRGEN then processes these values through a coefficient quantization routine MKCSD to convert the coefficients to canonical signed digit form [25]. Next, the user specifies the number of CSD digits and the internal wordlength to be used. The CSD values are optionally optimized to obtain the best possible frequency response with the given number of non-zero canonical signed digits using the CSDOPT program [26]. FIRGEN then displays a plot of the frequency response obtained with the floating-point coefficient values as well as with the CSD quantized values (Fig. 2). At this point the designer can iterate on the quantization till the number of CSD digits used produces an acceptable response.

B. Architecture Generation

Once the CSD coefficient values have been decided on, FIRGEN can proceed with the architecture and floorplan generation (Fig. 1). Note that the user can also enter the filter coefficients directly. The architecture used in FIRGEN is described in Section III. FIRGEN generates an architectural schematic that consists of integrated circuit macrocells for the basic hardware functions addition, subtraction, and sample delay. Additionally, FIRGEN also generates schematics for the clock generation and distribution network as well as the data distribution network using various buffer macrocells. The complete set of macrocells required by FIRGEN to generate the schematic are described in Appendix A.

The schematic is created as a text file using the Structure Description Language, SDL, defined for the Lager layout generation system [10]. This description is stored in the OCT data base [27] using the design manager tool DMoct in Lager. The OCT data base then serves as the interface to the layout generation tools.

C. Floorplan Generation

The floorplan provides placement constraints to automatic place and route tools. In the floorplan, the place-
ment of all macrocells, including those of clock and data buffers, in the schematic, as well as the definition of routing channels are specified. The actual optimized floorplan is derived using a multi-step user-guided process which is described in Section IV. The floorplan is stored using the Floorplan Description Language, FDL [28]. In the FDL description, macrocells are clustered into modules that also contain routing channels. Associated with each of these modules (which can be hierarchical) are descriptions of 1) neighboring modules; 2) signal and power routing entering and/or exiting the module; and 3) preferred routing layer and routing direction. Thus, 1) specifies the relative (rather than the absolute) macrocell placement, while 2) and 3) serve to give routing hints to the place and route tools. Note, the floorplan description is symbolic in nature since no actual geometry or physical information is specified. This floorplan file can then be translated into alternative formats acceptable for different place and route tools.

The place and route tool is then expected to take this floorplan file (or a translated floorplan) and perform physical cell placement and generate physical connections for signal and power routing. To obtain a compact layout, it is expected that feedthroughs are provided in the macrocells (Section IV). These are exploited to divide the global routing into local channel routing problems [29]. Sea-of-gate routers, which are channelless routers, do not use the channel information and routing hints provided in the floorplan. However, module placement specified in the floorplan forces the router to generate the desired regular routing that is especially critical for the data and clock distribution network.

D. Layout Generation

The final layout generation can be performed using automatic macrocell generation and place and route programs. These programs are expected to take as input the netlist (Section II-B) and floorplan (Section II-C) information. Macrocell generators must exist for generating cells with parameters as shown in Appendix A. Currently two layout generation systems are interfaced to FIRGEN: 1) the Lager tools [10] consisting of the macrocell generator TimLager and the place and route tool Flint [28] for use with a MOSIS compatible scalable CMOS library [13], and 2) the Tangate gate-array place and route tool [11] for use with the Texas Instruments commercial gate array cell libraries [30].

E. Technology Retargeting

To enable the use of different cell libraries for different technologies, FIRGEN uses a technology mapping function to generate the architecture netlist (Fig. 1). To use this mapping function, the cell library must contain a set of parameterized primitive cells as described in Appendix A. These primitive cells include macrocells such as carry-save adder, carry-save subtractor, vector-merge adder, registers, buffers, and clock generator.

At run-time, FIRGEN reads the cell library information, and using the technology mapping functions, generic cells in the filter schematic are replaced by the cell library-specific macrocells. This new schematic can then be used in the technology-specific layout generator to produce the actual filter layout.

III. Architecture Generator

The architecture generator creates a high-speed architecture with only one stage of pipeline latency as described in this section. The speed of the filter is defined as the rate at which input samples can be processed. To increase the speed it is necessary to reduce the critical path delay between input and output.

A. The Transpose Structure

Fig. 3 shows the generic form FIR filter structure. When this structure is directly translated into a circuit, the critical path between the input and output ports of the filter is the sum of the delay through the first multiplier \( h(0) \) and the delay through the accumulator which sums the weighted tap signals. The accumulator can be physically implemented by an adder chain (direct form) as shown in Fig. 4. If \( T_m \) is the multiplier delay, \( T_a \) is the adder delay, and \( N \) is the filter order, then the adder chain has an estimated arithmetic delay of

\[
T_{\text{chain}} = T_m + (N - 1) \times T_a.
\]  

The accumulator can also be implemented as an adder tree (Fig. 5) as proposed by Reutz [4]. This structure has an estimated arithmetic delay of

\[
T_{\text{tree}} = T_m + (\log_{1.5} N) T_a.
\]  

From (1) and (2), the adder tree structure can be expected to be faster than the adder chain. Note that the estimated arithmetic delay does not include the RC delay due to circuit parasitics.

The direct form structure can be transformed by repositioning the \( z^{-1} \) delays to obtain the so-called transposed structure as shown in Fig. 6. This structure can be derived by applying retiming [31] to the direct form structure as shown in [32] or by using transposition theorem [33]. In the transposed form the critical path delay \( T_{\text{transpose}} \) is estimated to be:

\[
T_{\text{transpose}} = T_m + T_a
\]  

thus it is expected to be faster than the tree structure of Fig. 5. Using the available \( z^{-1} \) registers, it is difficult to reduce the critical path delay further. The transpose structure is the basis for the architecture generation in FIRGEN.

Two drawbacks of the transpose structure have been pointed out. First, as the number of taps increases the input signal bus becomes longer and also has to be distributed to a larger number of tap inputs leading to large load capacitances [1]. By actual designs it has been found that by using appropriate data buffers on the input bus and by
distributing the network in a tree-like structure, it is possible to significantly reduce the capacitive loading on each buffer (Section III-E). For example, a 64-tap filter chip [12] has been realized where the MSB of the input bus has a total load of 3.5 pF with one-level buffering. By inserting one more buffer level, the capacitance for the two buffers are reduced to 1.58 and 1.85 pF, respectively. These capacitive loads can be driven at 100 MHz.

The second drawback is that the registers for the \( z^{-1} \) delays have to be larger in the transpose structure since they hold the accumulated sum instead of the input signal [4]. In the test chips presented in Section V, it was found that the increased register length actually leads to better pitch matching with the adders in the layout. There is no area overhead in the gate-array design (Section V-B)—the bounding area of a tap is determined by the size of the carry-save adder and the register is exactly pitch matched with the carry-save adder. For the custom macorcell design (Section V-A), the adder cells takes up the majority of tap area. The area overhead of the larger register is approximately 10%.

A more serious problem resulting from the larger register in the transpose form is the increased loading on the clock buses. To reduce the loading, a tree structure similar to that used for the data bus is employed (Section III-E).

By appropriate buffering, the RC delay on the clock lines can be reduced such that it does not become the critical path.

The critical path delay in the transposed form (3), can be reduced by reducing the delay for both the multiplier \( T_m \) and the adder \( T_a \) in each tap.

**B. Carry-Save Addition**

The adder delay \( T_a \) can be reduced by using carry-save addition. In a direct circuit implementation of the transpose structure if conventional carry-ripple adders (CRA's) are used for each addition [1] the critical path is limited by the ripple delay which can be large for large number of bits, e.g., in 2-\( \mu \)m CMOS, the speed of an \( n \)-bit carry-ripple adder is approximately \( 2n \) ns. To avoid the long critical path of the carry-ripple adder, the adders in each tap position are converted to carry-save adders (Fig. 7). This implies that instead of only a sum signal, two signals (the sum and carry) propagate through the datapath (Fig. 7). The advantage is that at each tap position the critical path contains the delay of the sum bit of a single bit adder as opposed to the carry ripple of an \( n \)-bit adder.

The tradeoff is that two \( z^{-1} \) registers are required: one to delay the sum and one to delay the carry. This is an efficient tradeoff since it doubles the register hardware (which typically leads to an area increase of 10%) but results in a speed increase by a factor of \( F \):

\[
F = n \frac{T_c}{T_a}
\]

where \( n \) is the wordlength, \( T_c \) is the carry-ripple delay per bit and \( T_a \) is the sum delay per bit. Typically, \( T_c/T_a \leq 1.0 \), this means that the carry-save adders provide a speedup by a factor of up to \( n \) at the expense of chip area increase by a factor of about 10%.

A so-called vector-merge adder (VMA) has to be used in the final stage to add the final sum and carry (Fig. 7), this is discussed in Section III-C. Note that the VMA is really just a carry-propagation adder.

By using carry-save adders and a vector-merge adder, the expression in (3) becomes

\[
T_{\text{transpose}} = T_m + T_{\text{csa}} + T_{\text{vma}}(n)
\]

where \( T_{\text{csa}} \) is the delay through a carry-save adder, and \( T_{\text{vma}} \) is the delay for the vector-merge adder. Observe that \( T_{\text{vma}} \) will in general depend on the wordlength \( n \) of the digital filter. Thus, \( T_a \) has been broken down into \( T_{\text{csa}} \) and \( T_{\text{vma}} \).

**C. Vector-Merge Adder**

In (5), the main terms contributing to \( T_{\text{transpose}} \) are \( T_m \) and \( T_{\text{vma}}(n) \). This delay can be reduced by introducing a single pipeline register before the vector-merge adder (Fig. 8). At the cost of one cycle latency, the critical path
becomes
\[ T_{\text{transpose}} = \max \{ T_m + T_{\text{csa}}, T_{\text{vma}}(n) \}. \] (6)

Observe that if \( T_{\text{vma}} \) dominates, chip area can be invested just for the vector-merge adder to speed up the circuit. For small filters, it has been observed that \( T_{\text{vma}} \) dominates and by using fast adders such as carry-select adder or carry-lookahead adder, substantial speed-up can be achieved (Section V-A). Alternatively, if throughput is the overriding consideration rather than pipeline latency, then a pipelined carry-ripple adder can be used for the vector-merge adder [7].

D. CSD Multipliers

For fixed-coefficient filters the multiplier delay \( T_m \) can be substantially reduced using CSD coding for the coefficients [7], [16], [25]. The CSD multiplier is made up of combination of hard-wired shifts and carry-save adders (Fig. 9).

For example, a coefficient 0.875 has a binary representation of
\[ 0.875 = 2^{-1} + 2^{-2} + 2^{-3}. \] (7)

This can be recoded into an equivalent canonical signed digit representation of
\[ 0.875 = 1 - 2^{-3}. \] (8)

Thus, for multiplication by 0.875, only one addition and one subtraction is needed instead of three additions (Fig. 9).

If carry-save adders are used, the delay through a CSD multiplier \( T_{\text{csd}} \) can be expressed as
\[ T_m = T_{\text{csd}} = (N_{\text{csd}} - 1) \times T_{\text{csa}} \] (9)

where \( N_{\text{csd}} \) is the maximum number of nonzero canonical signed digits in a coefficient, and \( T_{\text{csa}} \) is the delay through a carry-save adder.

Substituting (9) into (6) results in
\[ T_{\text{transpose}} = \max \{ (N_{\text{csd}} - 1) \times T_{\text{csa}} + T_{\text{csa}}, T_{\text{vma}}(n) \} \] (10)

\[ T_{\text{transpose}} = \max \{ N_{\text{csd}} \times T_{\text{csa}}, T_{\text{vma}}(n) \}. \] (11)

This equation includes only arithmetic delays in the architecture. In the actual IC design, RC delay due to layout parasitics needs to be taken into consideration.

E. Clock and Data Distribution Network

The RC delay of the input data bus is dominated by that for the MSB data bus since this bit must be broadcasted to multiple number of bit positions for proper sign-extension (Fig. 9). For example, the capacitive loading on the MSB data bus for a 11-tap filter has been found to be 40 times larger than that of any other data bits.

Two strategies are used in FIRGEN to combat this data distribution problem: a) tree-structured data buffers are used (Fig. 10), and b) the MSB data driver is made to have a larger driving capability than those for other data bits. The combination of these strategies enable the RC delay of the data bus to be reduced to an acceptable level.

The RC delay problem of the clock distribution network is attacked in a similar manner. Namely, two-level clock buffers were created to buffer the global and local clock signals, respectively (Fig. 10).

Since the RC delay on the input data bus is a function of the chip size, it is thus a function of the filter order.
The RC delay can be thus written as
\[ T_{RC} = T_{RC}(N) \]  
\[ (12) \]

to indicate that it is a function of filter order \( N \).

Adding the RC delay to the delay expressed in (11), the overall delay (Fig. 11) becomes
\[ T_{\text{transp}} = \max \{ N_{\text{cd}} \times T_{\text{csa}} + T_{\text{RC}}(N), T_{\text{vms}}(n) \}. \]  
\[ (13) \]

**F. Critical Path Analysis**

Note, the first term in (13), \( N_{\text{cd}} \times T_{\text{csa}} + T_{\text{RC}}(N) \), depends on two factors. First is the maximum number of canonical signed digits \( N_{\text{cd}} \) allocated for each filter coefficient. Second is the RC delay through the data distribution line, \( T_{\text{RC}} \). This is a function of filter complexity since as the filter-order \( N \) increases, the RC delay typically also increases. Both \( N_{\text{cd}} \) and \( N \) can be large for a stringent frequency response requirement.

For the second term in (13), \( T_{\text{vms}} \), the speed of the vector-merge adder depends only on the internal wordlength \( n \) of the filter, this is related to the input/output requirement of the filter as well as the extra guard bits necessary for satisfying the SNR requirement. The actual speed of the vector-merge adder also depends on the actual implementation of the vector-merge adder. As mentioned in Section III-C, the speed of the vector-merge adder can be improved by using fast adder structures.

If we assume that the tree structure (Fig. 5) uses carry-save adders and CSD multipliers, its critical path delay can be written as
\[ T_{\text{tree}} = \max \{ N_{\text{cd}} \times T_{\text{csa}} + (\log_{1.5} N)T_{\text{csa}}, T_{\text{vms}}(n) \}. \]  
\[ (14) \]

If \( T_{\text{vms}} \) term dominates, then both architectures have identical critical path delay, and the main advantage of the transpose architecture is its regular structure.

If \( T_{\text{vms}} \) does not dominate then the transpose form will be faster than the tree structure if
\[ T_{\text{RC}}(N) < \log_{1.5} N \times T_{\text{csa}}. \]  
\[ (15) \]

**G. Netlist Generation**

Based on the transposed architecture shown in Fig. 11, FIRGEN generates a netlist of macrocells for the filter integrated circuit. The netlist consists of two main parts: 1) a list of the macrocells used in the filter schematic; 2) a list of nets describing the interconnections between the macrocells in the filter schematic.

As a first step, the macrocells used in the filter schematic are mapped to the cell library specific macrocells. This mapping is done by reading in a library mapping file, and then perform the following mapping: 1) the generic macrocell name (e.g., CSA) is replaced by the technology-specific macrocell name (e.g., carry-save-add); 2) the generic pin name for the macrocell (e.g., AIN) is replaced by the technology-specific pin name (e.g., in1).

After this mapping has been done, the architectural schematic is generated with reference to the generic macrocell/pin names. These serve as pointers to the actual macrocell/pin names. In the final netlist, the actual library-specific macrocell/pin names are used. The netlist is described using SDL [10].

**IV. FLOORPLAN GENERATOR**

The floorplan generator in FIRGEN defines the relative placement of all macrocells in the filter netlist as generated above. The techniques used by the floorplan generator have been tested with custom macrocell place and route tools as well as with a sea-of-gates place and route program. In both cases, directly providing the architecture net list to the automatic place and route programs without floorplanning results in inferior routing and is not sufficient to produce a high-performance circuit. It is essential to provide floorplanning constraints to the place and route software to achieve compact layout, efficient routing and more importantly a proper layout of the clock and signal distribution network. Providing floorplan constraints not only improves layout efficiency but also speeds up the place and route phase. This is especially critical for large gate-array filter designs such as the 64-tap filter discussed in Section V-B.

**A. Linear Placement**

From the architecture (Fig. 11) it can be seen that there are two kinds of nets that have to be routed: 1) global
nets: the input data bus is distributed to all the carry-save adder macrocells and the clock signals are distributed to all the registers; 2) local nets: the carry/sum signals are connected between neighboring adder or register macrocells. To make the layout compact, all adder and register macrocells are placed in a row in the same order that they appear in the schematic (Fig. 11). Thus all the local nets can be routed in channels between the macrocells. The floorplan consists of a linear placement of the macrocells with routing channels in between as shown in (Fig. 12).

The global databus net has to be shifted at each carry-save adder input to implement the coefficient scaling (Fig. 9). To avoid congestion in routing this net, it is routed by a combination of local channels between the macrocells and feedthroughs provided in the macrocells. The shifting of the bus takes place in the local channel and the connection to successive macrocells runs through the preceding cells. For the sea-of-gate designs, feedthroughs do not have to be explicitly provided in the macrocells since the router is channelless and routes over the macrocell by default.

Routing of the global clock net is explained in Section IV-E. The linear placement (Fig. 12) constructed by the floorplan generator can result in a long and thin floorplan for large order filters. In this case the floorplan generator provides the capability of folding the row of macrocells into several rows (Fig. 13) using a folding procedure.

B. Floorplan Folding

To produce a folded floorplan, the floorplan generator in FIRGEN employs a multistep floorplanning procedure.

In the first step, a trial linear placement (Fig. 12) is produced, this gives an initial estimate of the width and height of the filter as follows:

\[
W_i = \sum_{i=1}^{N} (W_{up} + W_{rout})
\]

\[
H_i = n \times H_{cwa} + H_{ckbuf}
\]

where \(W_i\) is the estimated linear width, \(H_i\) is the estimated linear height, \(W_{up}\) is the width of tap \(i\), \(W_{rout}\) is the estimated routing channel width in tap \(i\), \(H_{cwa}\) is the height of a single-bit carry-save adder (assuming that its height is the pitch limiting factor), \(H_{ckbuf}\) is the height of the clock buffer, and \(n\) is the internal wordlength. The estimated aspect ratio \(W_i/H_i\) is compared with a user-specified aspect ratio \(A\). If \(A < W_i/H_i\) then folding is performed as follows.

C. Coarse Grain Folding

In the first step, a "coarse grain" folded floorplan is produced to meet the desired aspect ratio \(A\). The single row of macrocells of width \(W_i\) and height \(H_i\) is divided into \(N_r\) rows stacked on each other (Fig. 13). The width of each row in this coarse floorplan is given by \(W_i/N_r\) and the total height of the coarse floorplan is given by \(N_r \times H_i\).

\[
H_i, \text{ Setting its aspect ratio equal to } A
\]

\[
A = \frac{W_i}{N_r \times H_i}
\]

From this, \(N_r\) is calculated as

\[
N_r = \sqrt{\frac{W_i}{H_i \times A}}
\]

The user can then modify the number of rows \(N_r\) to be used in the final floorplan if desired. The final floorplanning step performs a "fine grain" floorplanning with the number of rows chosen by the user.

D. Fine Grain Folding

The fine grain folding decides exactly which and how many macrocells will go in each row. Each macrocell,
such as an adder, has a discrete and finite width. Therefore the coarse estimation of the row width given by $W_i/N_r$ must be refined so that the width of each row is the actual sum of macrocell widths and channel widths. At the same time to minimize blank spaces at the end of each row (and therefore minimize the chip area) it is desirable that the maximum deviation between the width of any two rows is minimized. To achieve this objective an optimization problem is formulated as follows. If we define $\Delta W$ to be the maximum deviation between the width of any two rows:

$$\Delta W = \max_{i,j} (W_i - W_j), \quad i, j = 1, 2, \ldots, N_r - 1$$

(20)

where $W_i$ is the width of the $i$th row, the optimization problem can be stated as

minimize $(\Delta W)$ subject to the constraint

$$W_i = (m \times W_{add} + p \times W_{reg} + q \times W_{rout})$$

(21)

where $m, p, q$ are integers, $W_i$ is the width of the $i$th row, $W_{add}$ is the width of an adder, $W_{reg}$ is the width of a register, and $W_{rout}$ is the width of the routing channel.

An iterative optimization procedure [34] is applied to solve this problem until the maximum deviations between any two rows for two successive iterations remain the same. With this optimization the deviations between rows is typically reduced to less than 5%. The optimization results in an exact description of the macrocell placement for each row.

E. Data and Clock Distribution Network

As noted earlier, the data and clock distribution network and the floorplan for their placement is crucial in reducing clock and data skews in the filter. In FIRGEN, based on the ideas presented in [8], tree structures are used for both data and clock distribution networks. For a filter with linear structure, a single level data buffering scheme is used in which each data bit has its own data bus driver with the MSB data driver sized larger than those for other bits for the reasons mentioned earlier in Section III, a two-level buffering scheme is used for the clock distribution. In this scheme, a global clock driver buffers the output from the clock generator (or is built into the clock generator). This globally buffered clock signal is then distributed in a tree-like structure to the local clock buffers situated at each register macrocell (Fig. 12).

For larger folded filters, a two-level tree structure is used for data distribution. In these tree structures, global data drivers buffer the data bits, the signals from the global data drivers are in turn buffered at each row by local data buffers (Fig. 13). A variation of this local buffering scheme is to buffer the signal at each tap. In FIRGEN, the decision to use either of these data buffering scheme is determined by the availability in the cell library of sufficiently large and fast drivers, if such driver exists, the first scheme will be used.

The clock distribution in a folded filter is handled in a similar manner, except a three-level buffering scheme is used. The first-level drivers buffer the clock generator output. The second-level drivers buffer the global clock signals at each section of the folded filter structure, the final third-level drivers buffer the signals being broadcast to each register macrocell (Fig. 13). The final floorplan corresponds to either Fig. 12 or Fig. 13 and is generated using the FDL [10].

F. Custom Macrocell-Based Chip Floorplans

The 11-tap filter chips described in Section V-A were done using custom macrocells with the Laged system. The placement and route tool takes the floorplan and uses it to place the macrocell, creates the channels, taking advantage of the provided data feedthrough according to the routing hints, and actually only needs to physically implement the interconnections as was described earlier. Because the filter has only 11 taps, the floorplan is a linear placement of the macrocell. A comparison with a handcrafted design using the same cells have shown that the overhead due to the use of automatic place and route tool is less than 5%.

G. Gate-Array Based Chip Floorplans

Gate array place and route programs are typically geared toward random logic generation. When doing automatic placement and routing without any constraints simulated annealing algorithms are used for cell placements resulting in a discernible regular structure (Fig. 14(b)). This makes it difficult to generate the desired clock and data distribution layout. By imposing the floorplan, the desired structure is achieved (Fig. 14(a)); in addition, by fixing the placement of the cells, the run-time of the placement is completely eliminated, and the routing time is also considerably reduced.

A 64-tap BiCMOS chip was done using gate array cells (Section V-B). Due to the large number of taps, linear placement of all the macrocells results in an unacceptable aspect ratio. It is, therefore, necessary to fold the filter as described above. In the 64-tap filter, the filter is folded into 9 columns. As noted earlier, appropriate buffering of the data and clock signals are essential in a large and complex filter such as this. For this reason, multilevel BiCMOS buffers are used in both the clock and data lines.

V. Test Circuits

With the FIRGEN system, four chips have been generated, fabricated and tested. All of them have been tested for functionality and speed. A summary of the four chips is presented in Table I.

A. 11-Tap CMOS Filters

Three CMOS chips implementing an 11-tap pre-distortion filter for sinc compensation in a D/A converter [35] have been fabricated. A frequency response of the $x/sin$
Fig. 14. Gate-array placement results: (a) with and (b) without floorplan constraints.

<table>
<thead>
<tr>
<th>TABLE I</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SUMMARY OF FIRGEN CHIPS</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>BiCMOS</th>
<th>SINC 2.0 A</th>
<th>SINC 2.0 B</th>
<th>SINC 1.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology</td>
<td>0.8-μm BiCMOS</td>
<td>2.0-μm P-Well CMOS</td>
<td>2.0-μm N-Well CMOS</td>
<td>1.2-μm N-Well CMOS</td>
</tr>
<tr>
<td>Core Size</td>
<td>8.96 mm × 5.43 mm</td>
<td>5.02 mm × 1.86 mm</td>
<td>5.46 mm × 2.25 mm</td>
<td>3.27 mm × 1.35 mm</td>
</tr>
<tr>
<td>Chip Size</td>
<td>11.99 mm × 3.72 mm</td>
<td>7.30 mm × 4.23 mm</td>
<td>4.52 mm × 3.83 mm</td>
<td></td>
</tr>
<tr>
<td>Transistor Count</td>
<td>220 000</td>
<td>11 766</td>
<td>12 176</td>
<td>12 500</td>
</tr>
<tr>
<td>Maximum Speed</td>
<td>100 MHz</td>
<td>25 MHz</td>
<td>50 MHz</td>
<td>112 MHz</td>
</tr>
<tr>
<td>Power Dissipation</td>
<td>6 W @ 100 MHz</td>
<td>150 mW @ 25 MHz</td>
<td>341 mW @ 50 MHz</td>
<td>332 mW @ 112 MHz</td>
</tr>
</tbody>
</table>

(x) function is shown in Fig. 15, and a list of the canonical signed digit filter coefficients are presented in Table II. All three chips have been generated using the Lager silicon assembly system.

1) 11-Tap 2-μm CMOS Filter with Carry-Ripple Vector-Merge Adder: The first CMOS chip is a 2-μm p-well CMOS chip fabricated through MOSIS. It uses a simple ripple adder as the vector-merge adder and is generated with FIRGEN and the Lager system. The design has been tested to be functional at a speed of 25 MHz. The chip dissipates about 150 mW at maximum operating frequency.

2) 11-Tap 2-μm CMOS Filter with Carry-Select Vector-Merge Adder: The second CMOS chip is a 2-μm
n-well CMOS chip fabricated through MOSIS. The chip is identical to the above 2-μm chip but with a fast carry-select adder replacing the ripple adder as the vector-merge adder. The chip has been tested functionally up to 50 MHz and dissipates 341 mW at 50 MHz. The circuit speed represents a 2 X gain over the 2-μm chip discussed in Section V-A1. The chip has a slightly larger area due to the carry-select adder.

3) 11-Tap 1.2-μm CMOS Filter: The third CMOS chip (Fig. 16) uses the same fast carry-select adder as the 2-μm filter discussed in the previous section and is fabricated in 1.2-μm n-well CMOS technology through MOSIS. The chip has been tested functionally up to 112 MHz (Fig. 17) and dissipates 332 mW at maximum operating frequency. The higher operating speed implies a technology gain of 2.24 X over the 2-μm version of the chip (Section V-A2).

B. 64-Tap BiCMOS Filter

The fourth chip is a 64-tap filter chip built in Texas Instruments’ 0.8-μm BiCMOS gate array. It has been verified to be functional at clock rates up to 100 MHz and dissipates 6 W at the maximum operating frequency. The full detail of the chip can be found in [12].

C. Test Circuit Critical Path

For the three 11-tap filter chips, the critical path is the delay through the vector-merge adder T_{vm} (Fig. 11). The vector-merge adder for all three chips is a 14-b adder. By replacing the slow carry-ripple adder with fast carry-select adder, a gain in speed by a factor of 2.0 is achieved while using similar 2.0-μm technology. By going to an advanced 1.2-μm technology, the speed gain increases to a factor of 4.48.

The delay through the data bus, CSD multiplier and carry-save adder, T_{RC}(N) + N_{cs}/x × T_{cs}, (Fig. 11) is the dominant delay for large filters such as the 64 tap BiCMOS filter presented in Section V-B. Because the maximum number of nonzero CSD used in a filter tap is four, the critical path for this 64-tap filter is thus the RC data line delay plus the delay through four carry-save adders. Thus, T_{critical} is 4 × T_{cs} + T_{RC}(64).

D. Technology Factor

1) Speed: Improvement in the fabrication technology leads to faster transistor. This, in turn, will lead to faster adders for both carry-save adder and vector-merge adder. On the other hand, the RC delay will typically not scale with the technology. Thus, the RC delay will be reduced at a slower rate than the corresponding IC circuit delay. It is for this reason that as improved technology is employed the critical path is T_{critical} = N_{cs} × T_{cs} + T_{RC}(N) (Fig. 11). This is especially true for large order filters as is demonstrated by the BiCMOS filter in which the delay through the 14-b vector-merge adder is only 3.9 nS [12], which would have translated to a maximum operating speed of over 250 MHz; but actual circuit is limited to a maximum operation speed of 100 MHz by the RC data line delay.

If the RC line delay is not the dominant delay, then the improvement in fabrication technology can lead to substantial improvement in circuit speed. This is evident by the speed gain (2.24) of the 1.2-μm filter over the corresponding 2.0-μm filter in which the same filter schematic, including the carry-select adder for the vector-merge adder, is used.

2) Power: An important consideration in the high speed design is the power dissipation of the circuit. Improved technology leads to higher speed circuits which can also lead to higher power dissipation as the power in a CMOS circuit is [36]

\[ P_d = C_L V_{dd}^2 f \]  

(22)

where \( P_d \) is the dynamic power dissipation, \( C_L \) is the load capacitance, \( V_{dd} \) is the power supply, and \( f \) is the operating frequency.

Thus, the power dissipation is proportional to the operating frequency. It is known, however, that the parasitic capacitance \( C_p \) will be reduced as the technology is scaled down. Indeed, the 1.2-μm design consumes less power than the 2.0-μm design while operating at more than twice
the clock frequency. This indicates that the circuit parasitics, $C_L$, are reduced by a factor of more than 2.

3) BiCMOS Versus CMOS: In the BiCMOS design, only about 25% of the circuit is actually bipolar, mainly in buffers. This, however, has twofold benefits: first, the bipolar circuit can provide a better current driving capability than a pure CMOS circuit, leading to reduced delay time through the buffering circuits. Second, the BiCMOS buffer circuit simplifies circuit design task. For the complex data/clock distribution network present in high-order filters, the optimal buffer circuit design in CMOS is not a trivial task. Despite recent advances in automatic transistor sizing programs, an efficient automatic buffer generator has yet to be developed. Thus, the selection of CMOS buffer circuits rely on either having a variety of buffer circuits in the macrocell library or in using over-sized buffer to ensure that the load can be driven at the desired speed. Neither approaches is optimal. With the BiCMOS buffer circuit, this difficulty is removed and a near-optimal buffer can be used in a variety of load conditions.

VI. Conclusions

In this paper, we have described a top-down FIR filter design system called FIRGEN that is capable of generating high-performance digital FIR filter circuits directly from frequency domain specifications. The functional compiler employs a transposed form architecture with canonical signed digit multiplier and carry-save adders.

The compiler also automatically generates the floorplan files for the filter circuits so that the resultant design will give optimal layout for data and clock distribution trees. Appropriate buffering strategies are also used to break up long RC delays in the data and clock trees.

Finally, the compiler also has a retargeting capability that allows it to be adaptable to advances in technology or cell libraries. As a demonstration of this and other capabilities of FIRGEN, the system has been used to generate four chips in three different technologies using both custom and semi-custom macrocell libraries. All four chips have been found to be fully functional at frequencies ranging from 25 to 112 MHz.

APPENDIX A
Macrocell Library

The macrocells recognized and required by FIRGEN consists of several common arithmetic modules, registers, buffers, and clock generators. The arithmetic modules
consist of carry-save adder, carry-save subtractor and vector-merge adder. A full list of the required macrocells is presented in Table III.

1. Arithmetic Macrocells

The carry-save adder (CSA1) is simply a 3-input and 2-output full adder which produces both carry and sum bits. The carry-save subtractor (CSA2) is identical to the carry-save adder with the difference that one of the input is inverted. As such, the cell performs a one’s complement subtraction. In both of the CSA macrocells, feedthrough for the data line is built into the layout. In addition, the carry output is pre-shifted in the macrocell, e.g., the labelled carry output from cell i, is actually the pre-shifted carry output from bit position i – 1. Both of these layout requirements are used to promote efficient routing by the place and route tool and has been shown to be invaluable in contributing to the final compact chip layout. The vector-merging adder (VMA) is used to sum up the partial sum and carry vectors from the carry-save array, it can be implemented as either a simple ripple adder, a carry lookahead adder, a carry select adder, or a combination of any of the above. The actual implementation of the vector-merging adder depends on what type of adder is contained in the cell library.

2. Buffer and Register Macrocells

FIRGEN also required two kinds of register macrocells, a one-input register (for I/O), and a two-input register (for \( z^{-1} \) delays for sum and carry signals). Different size buffers are also recognized by FIRGEN and used appropriately in the architecture as required. The registers and buffers both contain built-in feedthroughs for the data line. The registers, in addition, have internal clock feedthroughs for distribution of clock signals. This serves to both constrain the routing and reduce the routing task faced by the place and route tool. Finally, a clock generator is optionally used by FIRGEN. The current SCUMOS MOSIS compatible cell library supports two clocking strategies, two-phase clocking and single-phase clocking. For the gate-array technology, a two-phase nonoverlapping clocking strategy is supported. The clocking strategy to be used by FIRGEN is encoded in the header for the cell library mapping function and is initialized appropriately when the library mapping function is read.

3. Macrocell Generation

The actual macrocell generation is done by the parameterizable macrocell generator in the target technology. In the Lager system, the macrocell generator program, TimLager, is used to generate the parameterizable cells.

ACKNOWLEDGMENT

The authors wish to thank Prof. H. Samueli, B. Wong, T. Lin, C. Chien, L. Tan, B. Chung, and K. Chan at UCLA for technical support and consultation; and T. Holloway, J. Graham, E. Born, and G. Stacey at Texas Instruments for BiCMOS device processing and testing.

REFERENCES


Rajiv Jain (S’83-M’84) received the B.Tech degree from the IIT Delhi, India, and the Ph.D. degree from the Katholieke Universiteit Leuven, Belgium.

He has worked at Siemens AG, Munich, on DSP design for an oversampled PCM CODEC. He was a group leader at IMEC, Leuven, for the ESPIART sponsored Cathedral silicon compiler project. During 1985-1988 he was a postdoctoral fellow at ERL, the University of California, Berkeley, where he worked on the IAG/IV silicon assembly system. Since 1988 he has been an Assistant Professor at UCLA where he has initiated research in the design of high-performance DSP circuits.

Paul T. Yang was born in Taiwan. He received the B.S.E.E. and M.S.E.E. degrees from the University of California, Los Angeles, where he is currently working towards the Ph.D. degree.

In 1986, he was with the Radiation Oncology Department of the University of California, San Francisco, as a Research Assistant working on cancer hyperthermia research. In 1987, he was with the Electronic Research Laboratory at the University of California, Berkeley, as a Research Assistant developing techniques for silicon micromachining. During the summer of 1989, he was a Member of the Technical Staff with the VLSI Design Laboratory, Texas Instruments. Since 1988, he has been a Graduate Student Researcher with the Department of Electrical Engineering at the University of California, Los Angeles, where he is presently pursuing the Ph.D. degree. His research interests include VLSI signal processing, digital communication systems, and CAD for ASIC designs.

Mr. Yang is a member of Tau Beta Pi and Eta Kappa Nu.

Toshiko Yoshino (M’83) was born in Japan. He received the B.S.E.E. degree in 1981 from Shinshu University, Nagano, Japan.

He joined Texas Instruments Inc. (Japan) in 1981 as an LSI Design Engineer. Since February 1986 he has been with the VLSI Design Laboratory of Texas Instruments, Dallas, where he has been working on the development of DSP functional compilers. His areas of research interest include the digital signal processing of speech, VLSI signal processing, and VLSI CAD.