International Journal of Computer Applications (0975 – 8887)
Volume 82 – No 18, November 2013

Area-Delay Estimation by Concurrent Optimization of FPGA Architecture Parameters using Geometric Programming

Y. Pandurangaiah
Head of the Department(ECE)
Vardhaman College of Engg
Hyderabad, India

J. Venkat Reddy
Student
Vardhaman College of Engg
Hyderabad, India

G. Kalyan Chakravarthy
Assistant Professor
Vardhaman College of Engg
Hyderabad, India

ABSTRACT
This paper presents the application of geometric programming for combined high-level and low-level architecture parameter exploration. This paper builds an geometric programming framework for reconfigurable architectures, and presents a full delay and area model of an FPGA. This optimization allows high-level architectural parameter selection and the transistor sizing to be done concurrently. The transistor values are derived using 45nm predictive technology model. CVX framework for MATLAB is used to run the geometric programming framework. The area and critical path delay are determined for given cost function by single-stage and multi-stage approach.

General Terms
CVX framework, Field Programmable Gate Arrays (FPGAs), Predictive Technology Model (PTM)

Keywords
Geometric Programming, Reconfigurable architectures

1. INTRODUCTION
In recent years, a considerable evolution in the architecture of FPGAs and this enhancements in the architectures, results in a expensive and time consuming experiments. Commercial FPGAs of every generation contains refined or new routing, memory, logic and embedded block structures. New or existing CAD tools are used by FPGA architects to map benchmark circuits to the architectures [3], [5]. Recent work, has suggested that the analytical techniques serve as the supplement to the experimental approach, in which the FPGA architectures are constructed using a model of relatively simple equations. Using this technique, the FPGA architect can investigate a much wider range of architectures.

The advantage of analytical approach is that the values for many architectural parameters can be optimized concurrently, which is different from experimental approach, in which typically one parameter is swept at a time. In [1], the simultaneous optimization of several parameters of routing structures of an FPGA by creating analytical equations and the impact of these parameters on the area of an FPGA is been shown, and a Geometric Programming (GP) framework is used to determine the values for these parameters.

In this paper, the simultaneous optimization of transistor sizing and architecture is done, which allows us in optimizing the both area and critical path delay (speed) of the FPGA. The work can be summarized as follows:

- A framework allowing, concurrent optimization of both low-level (transistor sizing) and high-level (architectural) parameters.
- Using geometric programming an area-delay model of an FPGA fabric is formulated.
- Concurrent optimization of low-level and high-level parameters will lead to a significantly different architectural conclusions compared to a traditional flow. Particularly, cluster size should be increased rather than decrease leading to delay improvements as delay becomes more important than area.

In [7], the general formulation for optimization of an electrical design based on an iteration process, involving successive routing and placement of circuits onto FPGA architectures is presented. In this work, the iterative refinement is removed, building an FPGA modeling technique and using a geometric program, a step of transistor sizing concurrently to a number of architectural parameters.

A geometric program is a constrained optimization problem of the following form:

Minimize : $f_0(x)$

Subject to : $g_i(x) = 1$, for $i = 1,2, \ldots, l$

where x is positive n-vector of real values, and functions f_i and g_i have special mathematical forms, known as posynomials and monomials, respectively. A monomial is a function

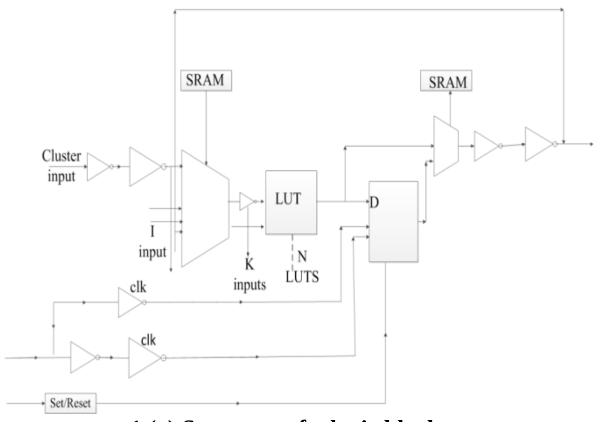
$$g(x) = cx_1^{a1}x_2^{a2} \ldots x_n^{an}$$

where the coefficient c must be positive. As an example, in this paper a monomial cost function $(T_{total})^z(A_{total})^z$ is used, where T_{total} and A_{total} represent the variables delay and area respectively and z is a constant. Geometric programming is used extensively for circuit design problems, which include wire sizing, transistor sizing and robust design. Use [6] for extensive review of geometric programming in the context of circuit design.

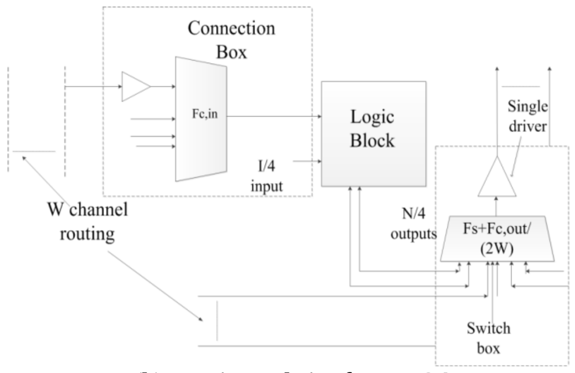
2. ARCHITECTURE FRAMEWORK
An island style FPGA is assumed in which an array of blocks are connected using tracks organized in vertical and horizontal channels with single driver routing, as represented in VPR 5.0 [3]. Figure 1(a) shows the structure of a logic block of an FPGA architecture which consists of configurable logic blocks (CLBs), which are tightly packed with K-input lookup tables (LUTs) connected with N LUTs and with I external inputs. A K-level pass transistor multiplexer tree is used to implement a K-input LUT.

Table 1. Model Parameters

Architectural Parameters :	
K	Number of inputs per lookup table
N	Number of lookup tables per logic block
I	Number of inputs per logic block
$F_{c,in}$	Number of tracks that connect to each logic input pin
$F_{c,out}$	Number of tracks each logic block can connect to
F_s	Number of track end points that connect to each track driver
Circuit Parameters :	
P	Rent parameter of a given circuit
n_2	Number of 2-LUTs in a given circuit
d_2	Depth of circuit netlist in number of 2-LUTs



1 (a) Structure of a logic block



1(b) Routing Fabric of an FPGA
Fig.1.Detailed view of FPGA architecture

Figure 1(b) shows the routing architecture, which consists of switch boxes and connection boxes. Three parameters are used to describe the routing blocks in the architecture :

$F_{c,in}$: number of tracks that connect to each logic block input,

$F_{c,out}$: number of tracks that each logic block output can connect and

F_s : number of track end points that connect to each channel driver.

In FPGA except for LUT multiplexer, all other multiplexers are implemented using a two stage pass transistor. The multiplexers are used to configure the signal routing paths around the device , and thus are connected to the SRAM configuration memory.

3. DELAY MODEL

It has been shown previously that geometric programming is capable of optimizing the transistor sizing for delay [15]. Here this type of delay optimization technique is employed to model the combination of pass transistor structures and CMOS present in FPGA devices. The delay of a critical signal path through a circuit implemented on FPGA is considered. The critical path will pass through CLBs, CLB feedback paths, LUTs, connection boxes and switch boxes. The formula in (1) is used, where each term is described below.

$$\begin{aligned} T_{total} = {} & T_{reg\ to\ OMUX} + D_i T_{LUT\ F/B\ path} + (D_k - 1)T_{LUT\ delay} \\ & + D_c T_{O/P\ CB\ delay} + D_c D_r T_{SB\ delay} \\ & + D_c T_{I/P\ CB\ delay} + D_c T_{input\ MUX\ delay} \\ & + T_{LUT\ to\ reg\ delay}. \end{aligned} \tag{1}$$

D_c represents the number of CLBs through which the critical path travels. D_k represents depth of the netlist when implemented in K-input LUTs.

$D_i = D_k - D_c$ represents the number of the internal feedback connections through which critical path traverses. D_r represents number of switch boxes through which the each external connection on the critical path propagates. To estimate the net list depths D_c, D_k and D_i the methods are employed from [19].

Each transistor in the circuit can also be represented as an RC network as shown in the figure 2. The capacitance and resistances values can be derived from the SPICE models of MOSFET devices. Here the values are derived using the 45nm predictive technology model [14]. Each resistance value for a transistor in architecture takes the form (2) and capacitance of form (3) or (4). R_C, C_D and C_G represent the channel resistance, diffusion capacitance and gate capacitance of a transistor respectively. S_i represents the width of transistor assuming all the transistors have minimum length. R_{nom} and $C_{nom,x}$ nominal values are dependent on the process technology, the type of transistor and in case of capacitance, whether it is nominal diffusion or gate capacitance.

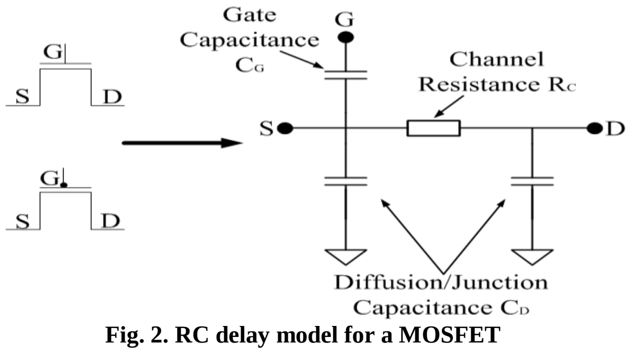


Fig. 2. RC delay model for a MOSFET

$$R_C = R_{nom}/S_i \tag{2}$$
$$C_G = C_{nom,G}S_i \tag{3}$$
$$C_D = C_{nom,D}S_i \tag{4}$$

The delays through each of the paths (1) are evaluated, by employing the Elmore delay model [13]. The Elmore delay model is used to represent the delay in the networks of RC trees and previously been shown to model the delay in the FPGA routing pass transistor networks [8]. The Elmore delay is calculated by evaluating the sum of each segment delay from signal to its sink, as shown in (5), where delay of each segment is the sum of the resistance along the path multiplied by capacitance of that segment.

$$T_{elmore} = \sum_{\text{paths i source to sink}} C_i R_{\text{source to } C_i} \qquad (5)$$

The expression of the total delay can be derived by breaking down the delay terms T_x in (1) into its constituent paths. Each path begins at VDD or GND and ends at a transistor gate input, which leads to a number of paths as given in figure 3(a-j).

The delay $T_{\text{reg to OMUX}}$ corresponds to the delay from the register producing critical signal path over the MUX selecting whether the LUT output is being registered or not, and over the two-level buffer as given in Figure 3(c).

The delay $T_{\text{LUT F/B path}}$ corresponds to the delay from the BLE output buffer over the pass transistor based MUX on the LUT input to its buffer, as given in figure 3 (b).

The delay $T_{\text{LUT delay}}$ corresponds to the delay from the LUT driver over all the levels of the multiplexer implementing the LUT, the 2:1 MUX and to the LUT output driver. This delay is the sum of the paths given in Figure 3(h) and 3(e).

The delay $T_{\text{O/P CB delay}}$ corresponds to the delay from the BLE output buffer over the switch box multiplier to its first inverting buffer, as given in Figure 3(b). In this case the Elmore delay is through the path to the switch box driver.

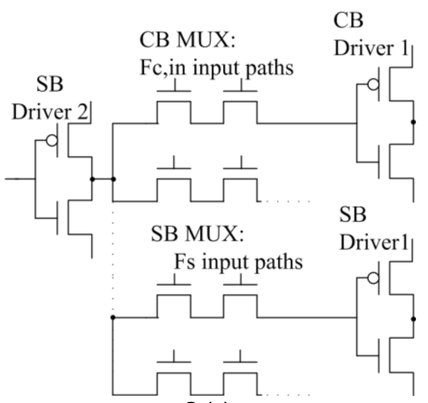
The delay $T_{\text{SB delay}}$ corresponds to the routing path signal between switchboxes. This represents sum of the driver delay and delay over the two level switchbox multiplier, as given in Figure 3(a) and 4(i).

The delay $T_{\text{input MUX delay}}$ corresponds to the delay from the connection box output driver over the LUT input select multiplexer to the LUT input driver.

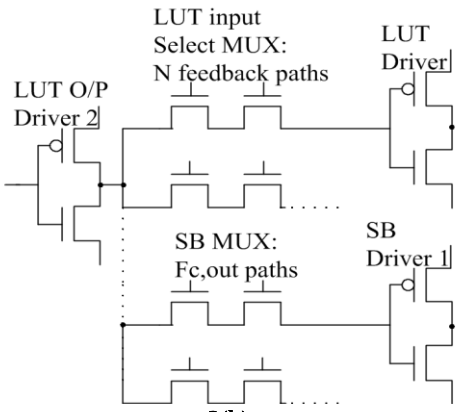
The delay $T_{\text{I/P CB delay}}$ corresponds to the path over the switchbox to the connection box, where the routed signal is consumed. This is the sum of the delay over the connection box, the driver delay and over the two inverting drivers in the connection box. These are given in Figure 3(a), 3(i) and 3(j) respectively.

Finally, the delay $T_{\text{LUT to reg delay}}$ corresponds to the delay over the multiplexer implementing the LUT to the register input where the critical path terminates, as in Figure 3(f).

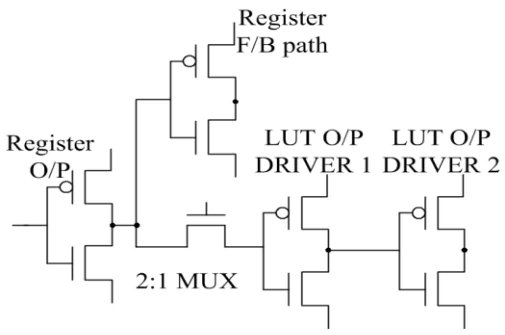
Since each driving gate has two paths to consider: the charge and discharge path from the driving gate, the terms representing delay are given as inequalities. As an example, the inequality for the charge path through the nMOS transistor which corresponds to the delay $T_{\text{O/P CB delay}}$ as given in (6), as in the Figure 3(b).



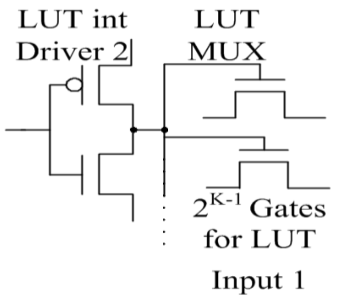
3 (a)



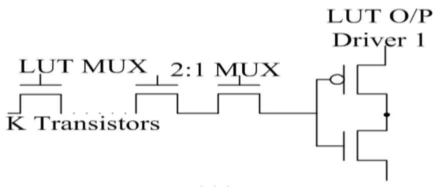
3(b)



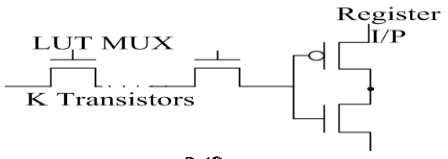
3(c)



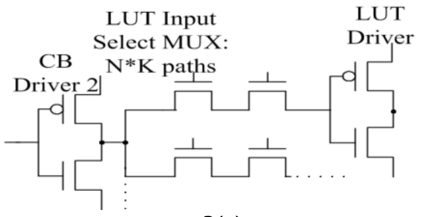
3(d)



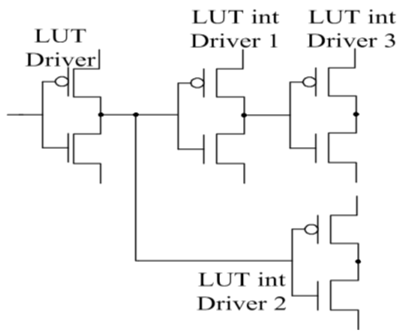
3(e)



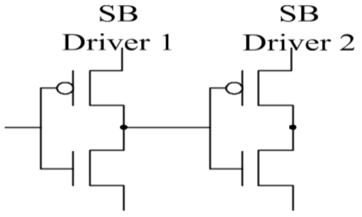
3 (f)



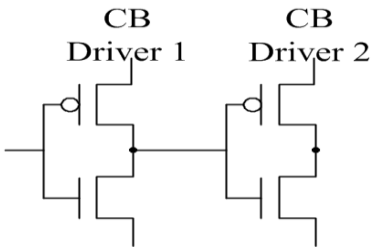
3(g)



3(h)



3(i)



3(j)

Fig. 3. RC delay models for circuit path

$$\begin{aligned} T_{O/P\,CB\,delay} \geq\ & R_{C,n,LOdrv2}(C_{D,n,LOdrv2} \\ & + NKC_{D,mux} + F_{c,out}C_{D,SB\_mux}) \\ & + (R_{C,LOdrv2} + R_{C,SB\_mux})2C_{D,SB\_mux} \\ & + (R_{C,LOdrv2} + 2R_{C,SB\_mux}) \\ & \times (C_{D,SB\_mux} + C_{G,SB\_driver}) \end{aligned} \quad (6)$$

4. AREA MODEL

An area model of an FPGA routing fabric alone was first presented in [1], and is summarized in section 4.2. In this work the model is extended to deal with variable buffer sizing and include both the routing and logic architecture. The area model is based on minimum width transistor sizing model employed in [5].

To evaluate how much logic area is consumed, use $N_c = [\sqrt{n_c}]^2$, where n_c is the number CLBs and estimated using the formula in [4]. The total area of an FPGA, A_{total} corresponds to the sum of the logic area A_l and routing area A_r, as in (7).

$$A_{total} = A_l + A_r \quad (7)$$

4.1 Logic Block Area

The area of the logic block is sum of the area dedicated to the following: the LUT input select multiplexer; the LUT, 2:1 multiplexer register and output buffer combination; the clock buffer and the set/reset logic. The clock buffer sizing and set/reset logic are assumed to be constant irrespective of the logic block architecture, the values are taken from [5]. Similarly, the size of the register size on the LUT output is assumed to be constant.

The LUT area is composed of pass transistors multiplexer cells, SRAM cells and internal drivers. The pass transistors size in the multiplexer ($S_{n,LM}$) is assumed to be equivalent. Similarly, the input buffer scheme is assumed to have the same size transistors. B_{li} represents the sum of these buffer areas. This leads to (8) as an expression for the area consumed by K-input LUT, where B_{li}, is the size of buffer driving LUT input and S_{SR} is the size of an SRAM cell.

$$A_{lut} = 2^K S_{SR} + KB_{li} + (2^{K+1} - 2)S_{n,LM} \quad (8)$$

The 2:1 multiplexer consists of one level pass transistor multiplexer. The area A_{21mux}, given by (9), where $S_{n,21mux}$

corresponds to the size of each two pass transistors implementing 2:1 multiplexer and S_{SR} corresponds to the one bit configuration memory required.

$$A_{21mux} = S_{SR} + 2S_{n,21mux} \tag{9}$$

The sum of the transistor areas for the two inverters implementing the driver gives the output buffer combination. The area consumed by these combination of inverters is given by (10).

$$B_{lo} = S_{n,LOdrv1} + S_{p,LOdrv1} + S_{n,LOdrv2} + S_{p,LOdrv2} \tag{10}$$

where $S_{n,LOdrv*}$ and $S_{p,LOdrv*}$ represents the size of each nMOS and pMOS transistors respectively in a CMOS inverter.

An expression for the approximation of multiplexer area is given by (11) and (12). The S_n corresponds to the size of pass transistor and E corresponds to the number of inputs.

$$A_{mux} = S_n(E+[\sqrt{E}]) + S_{SR}([\frac{E}{\sqrt{E}}]+[\sqrt{E}]) \tag{11}$$

$$\approx S_n(E+\sqrt{E}) + 2S_{SR}\sqrt{E} \tag{12}$$

Each input select multiplexer is fully connected; every output feedback path and every input from the connection box can reach any LUT input, which leads to the expression in (13), which gives the area devoted to each of these multiplexers. $S_{n,ISmux}$ corresponds to the size of the pass transistors implementing the input select multiplexer. Since there are I+N inputs to the multiplexer, in (14) $E_{IS,tree}$ corresponds to the number of pass transistors in the multiplexer tree and in (15) $E_{IS,tree}$ corresponds to the number of SRAM bits.

$$A_{ISmux} = E_{IS,tree}S_{n,ISmux} + E_{IS,RAM}S_{SR} \tag{13}$$

$$E_{IS,tree} = N + I + [\sqrt{N+I}] \tag{14}$$

$$E_{IS,RAM} = [\sqrt{N+I}] + [\sqrt{N+I}] \tag{15}$$

Combining the above constituent parts of the logic block leads to an expression in (16). The total area of the FPGA fabric is given in (17).

$$A_{LB} = NA_{LUT} + NA_{reg} + NA_{21mux} + KNA_{ISmux} + NB_{lo} + A_{clkB} + A_{rst} \tag{16}$$

$$A_l = N_c A_{LB} \tag{17}$$

4.2 Routing Area

The amount of silicon area devoted to the routing fabric to consist of all the connection box and switch box multiplexers, in addition to their configuration memories and output buffers. Thus, the routing area will depend on the size of the multiplexers used to connect the signals to and from the logic blocks and I/O pins, the transistor sizing, the channel width and the size of the grid of logic cells.

The estimation of multiplexer sizes in the connection and switch boxes is based on the observation that the expression for the area of two level multiplexer in (11) can be approximated as given in (12). The size of these multiplexers will depend on the channel width of the device.

The model developed in [2] is used to estimate the channel width. The model for architectures with the wires that span one logic block is shown in (18), where the minimum channel width W_{min} is described by (19), and β, $\alpha_{in,}$ α_{out} and p_f are empirical constants. In (19), λ corresponds to the average number of inputs used on each logic block and $\bar{R}$ corresponds to the average point-to-point wire length. The methods given in [9] are used to calculate the value of point-to-point wire-length for different logic parameters.

$$W = W_{min} + \frac{1}{\beta}(\frac{W_{min}}{F_s})(\frac{W_{min}}{F_{c,in}})^{\alpha_{in}}(\frac{W_{min}}{F_{c,out}})^{\alpha_{out}} \tag{18}$$

$$W_{min} = p_f \frac{\lambda\bar{R}}{2} \tag{19}$$

In routing fabric there are two types of multiplexers: connection box multiplexers and switch box multiplexers. Using the approximation of (12) gives the expression of multiplexer area for the connection box as given in (20) and (21) gives the approximation of switchbox area. $F'_{c,in} = \frac{F_{c,in}}{W}$ corresponds to the number of tracks that connect to the each logic block input and $F'_{c,out} = \frac{F_{c,out}}{W}$ corresponds to the proportion of routing tracks that the each logic block output connects to. Combining the above models lead to (22) for the routing area.

$$S_{cb} = S_{n,cb}(WF'_{c,in} + \sqrt{WF'_{c,in}}) + 2S_{SR}\sqrt{WF'_{c,in}} \tag{20}$$

$$S_{sb,m} = S_{n,sb}\left(\frac{N}{2}F'_{c,out} + F_s + \sqrt{\frac{N}{2}F'_{c,out} + F_s}\right) + 2S_{SR}\sqrt{\frac{N}{2}F'_{c,out} + F_s} \tag{21}$$

$$A_r = IN_c(S_{cb} + B_{cb}) + 4I_{io}\sqrt{N_c}(S_{cb,io} + B_{cb,io}) + 2N_{s,m}W(S_{sb,m} + B_{sb,m}) + 6N_{s,e}W(S_{sb,e} + B_{sb,e}) \tag{22}$$

5. GEOMETRIC PROGRAMMING FORMULATION

This section show that the model can be expressed in a form conformable to GP. It is essential to express the model as posynomial terms less than or equal to one or as monomial terms with equality to one. The cost function is considered first, which takes the form of a monomial (23). It is possible to by varying the exponent weight z, for example, targeting only delay by setting $z = 1$, or an equal weighting by setting

$z = 0.5$. The exponent weight must be constant for each run of the GP.

$$\min : T_{total}^{z} A_{total}^{1-z} \quad (23)$$

The model presented in before section is not in a form that is conformable to GP. The GP representation of the routing architecture model was given in [1], here the focus is on presenting the logic area constraints in the correct form.

The sum of nMOS and pMOS transistors for each inverting stage gives the area of its each respective buffer in the FPGA , for example B_{lo} in (10). The expressions (24) and (25) gives the transformation into posynomial form for buffer area, where B_x corresponds to the area of the buffer x.

$$B_x = \sum_{\text{all inverters in x}} S_n + S_p \quad (24)$$

$$\Rightarrow \sum_{\text{all inverters in x}} S_n B_x^{-1} + S_p B_x^{-1} \le 1 \quad (25)$$

The expressions (26)-(30) gives the area constraints in a standard form GP representation. The sum of logic area and routing area in (7) maps directly to the inequality constraint in (31), which gives an example, how the model maps to the constraints.

$$2^K S_{SR} A_{lut}^{-1} + K B_{li} A_{lut}^{-1} + \left(2^{K+1} - 2\right) S_{n,LM} A_{lut}^{-1} \le 1 \quad (26)$$

$$S_{SR} A_{21mux}^{-1} + 2 S_{n,21mux} S_{SR} A_{21mux}^{-1} \le 1 \quad (27)$$

$$E_{IS,tree} S_{n,ISmux} A_{ISmux}^{-1} + E_{IS,RAM} S_{SR} A_{ISmux}^{-1} \le 1 \quad (28)$$

$$NA_{LUT} A_{LB}^{-1} + NA_{reg} A_{LB}^{-1} + NA_{21mux} A_{LB}^{-1} + KNA_{ISmux} A_{LB}^{-1} + NB_{lo} A_{LB}^{-1} + A_{clkB} A_{LB}^{-1} + A_{rst} A_{LB}^{-1} \le 1 \quad (29)$$

$$N_c A_{LB} A_l^{-1} = 1 \quad (30)$$

$$A_{total}^{-1} A_l + A_{total}^{-1} A_r \le 1 \quad (31)$$

The mapping of delay constraints is straightforward, as they take the posynomial form. The expressions (32)-(41) gives an example how delay constraints are represented in GP. The example shows the delay and the related constraints between the inverters used in the switch box buffer. The expressions in (32) and (33) represent the charge/discharge path of first inverter through pMOS and nMOS transistors respectively, where $T_{SB\,inv1\,inv2}$ corresponds to the variable representing the delay. C_{G,SB_inv2} corresponds to the load capacitance of the second inverter gate and is the sum of two transistor gates used to make up the inverter as expressed in (34). The expressions (35)-(40) gives the required capacitance and resistance values, where $R_{nom,*}$ and $C_{nom,*}$ correspond to the nominal values for a minimum gate length transistor in given technology.

$$T_{SB\,inv1\,inv2}^{-1} R_{C,n,SB\_inv1} C_{D,n,SB\_inv1} + T_{SB\,inv1\,inv2}^{-1} R_{C,n,SB\_inv1} C_{G,SB\_inv2} \le 1 \quad (32)$$

$$T_{SB\,inv1\,inv2}^{-1} R_{C,p,SB\_inv1} C_{D,n,SB\_inv1} + T_{SB\,inv1\,inv2}^{-1} R_{C,n,SB\_inv1} C_{G,SB\_inv2} \le 1 \quad (33)$$

$$C_{G,SB\_inv2}^{-1} C_{G,p,SB\_inv2} + C_{G,SB\_inv2}^{-1} C_{G,n,SB\_inv2} \le 1 \quad (34)$$

$$R_{nom,nMOS} R_{C,n,SB\_inv1}^{-1} S_{n,SB\_inv1}^{-1} \le 1 \quad (35)$$

$$R_{nom,pMOS} R_{C,p,SB\_inv1}^{-1} S_{p,SB\_inv1}^{-1} \le 1 \quad (36)$$

$$C_{nom,D,nMOS} C_{D,n,SB\_inv1}^{-1} S_{n,SB\_inv1} \le 1 \quad (37)$$

$$C_{nom,D,pMOS} C_{D,p,SB\_inv1}^{-1} S_{p,SB\_inv1} \le 1 \quad (38)$$

$$C_{nom,G,nMOS} C_{G,n,SB\_inv2}^{-1} S_{n,SB\_inv2} \le 1 \quad (39)$$

$$C_{nom,G,pMOS} C_{G,p,SB\_inv2}^{-1} S_{p,SB\_inv2} \le 1 \quad (40)$$

$$S_{TECH} S_{n,SB\_inv1}^{-1} \le 1 \quad (41)$$

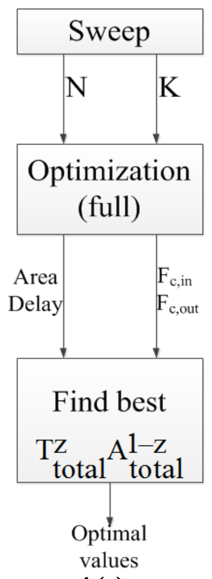
The expression in (41) used to ensure that the transistor size does not violate the smallest feature size possible in the process technology, where S_{TECH} corresponds to the constant representing the minimum feature size. The final constraints must be applied to all transistors in the GP.

The Geometric program takes approximately 43 seconds to run on a Intel Dual Core i5-2450M 2.5GHZ running windows 7. This is for each K and N logic parameters sweep.

6. RESULTS

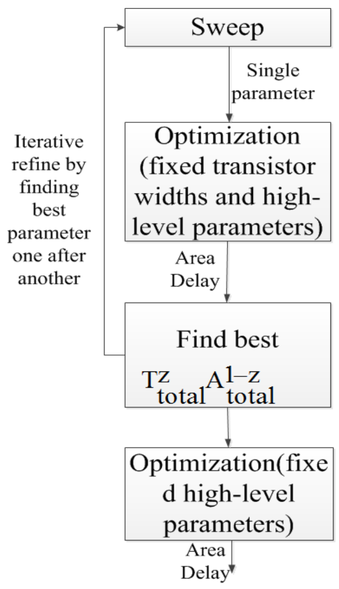
To demonstrate the power of Geometric Programming framework, the framework is run using CVX framework in MATLAB [11]. Two different flows using Geometric Programming framework are modeled, to demonstrate the impact of concurrently optimizing the low-level and high-level parameters.

In the first experimental approach, the K and N logic parameters are fixed for each run of the optimization tool. To find the optimal set of parameters, it requires to sweep across the values of interest. Each run of the tool reports the value of the total area, critical path delay and the objective function. The best architecture is selected for which, the values contributes the best value of objective function. Figure 4(a) shows the first experimental flow.



4 (a)

In the second experimental approach, each of K, N, $F_{c,in}$ and $F_{c,out}$ is successively chosen. This experimental flow is as shown in Figure 4(b). The sweep across $K = 2-7$, will determine the best LUT for an randomly chosen value of N, $F_{c,in}$ and $F_{c,out}$. Then K is fixed and then sweep across the CLB size $N = 2-12$. Similarly, the routing flexibility parameters are chosen from a sweep of different values.


Sweep
Single parameter
Iterative refine by finding best parameter one after another
Optimization (fixed transistor widths and high-level parameters)
Area
Delay
Find best
$T^{z}_{total} A^{1-z}_{total}$
Optimization(fixed high-level parameters)
Area
Delay

4(b)

Fig.4.The two flows used in the experiment

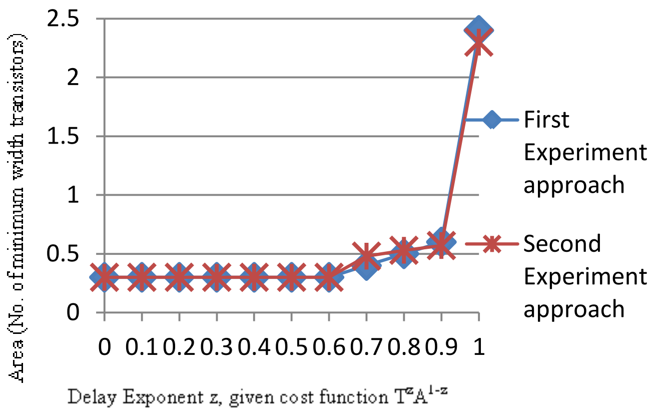


Figure.5 (a). Area of each approach

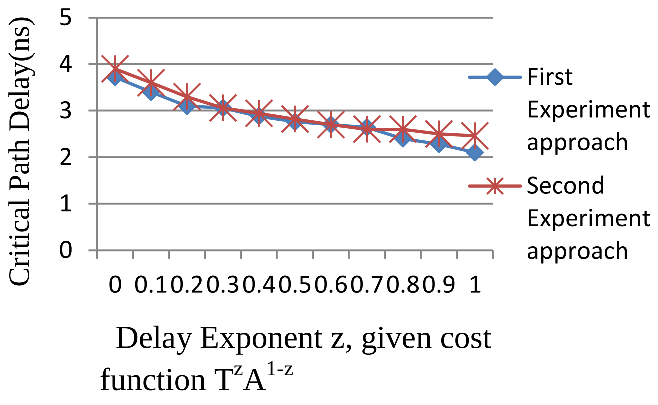


Figure.5(b) Critical path delay of each approach

During the exploration the exponent parameter z was varied to check how each approach performs depending on the weight of cost function. Figure 5(a) shows the area in minimum width transistors when varying the exponent z in the objective function of $T^{z}_{total} A^{1-z}_{total}$ and Figure 5(b) shows the critical path delay of each architecture. In single stage approach the delay improves when the cost function is weighted towards delay as objective. The multi-stage heuristic performs worse for both metrics - around 1% in area and 6% in delay compared to the best architecture.

The geometric programming approach can be rewarded in terms of run time. The geometric program framework takes approximately 40 seconds to solve, whereas, VPR tool takes approximately 5 minutes to run an architecture file.

7. CONCLUSION

This paper presents the use of Geometric Programming for fast and early stage exploration of configurable architectures. This approach allows the concurrent optimization of high-level and low-level architecture parameters, and shows that it is possible to gain in performance. In this experiment, the transistor values are derived using 45nm predictive technology model (PTM). The graphs are plotted for area and delay of the architectures for both single and multi-stage approach by varying the exponent z in the objective function $T^{z}_{total} A^{1-z}_{total}$.

In future work, spice is used to extract accurate delay information and wire delay model can also be included to improve the accuracy of the modeling approach.

8. REFERENCES

[1] A. M. Smith, G. A. Constantinides, and P. Y. K. Cheung, "Area estimation and optimization of FPGA routing fabrics," in *Int'l Conf. on Field-Programmable Logic and Applications*, Sep. 2009.

[2] W. Fang and J. Rose, "Modeling FPGA routing demand in early-stage architecture development," in *Int'l Symp. on Field-Programmable Gate Arrays*, Feb. 2008, pp. 139–148.

[3] J. Luu, I. Kuon, P. Jamieson, T. Campbell, A. Ye, M. Fang, and J. Rose, "Vpr 5.0: FPGA CAD and architecture exploration tools with singledriver routing, heterogeneity and process scaling," in *Int'l Symp. on* Field-Programmable Gate Arrays, Feb. 2009, pp. 133–142.

[4] A. Lam, S. J. Wilton, P. Leong, and W. Luk, "An analytical model describing the relationships between logic architecture and FPGA density," in *Int'l Conf. on Field-Programmable Logic and Applications*, Sep. 2008.

[5] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep- submicron FPGAs*. Kluwer Academic Publishers, 1999.

[6] S. P. Boyd, S.-J. Kim, D. D. Patil, and M. A. Horowitz, "Digital circuit optimization via geometric programming," *Operations Research*, vol. 53, no. 6, pp. 899–932, Nov-Dec 2008.

[7] I. Kuon and J. Rose, "Area and delay trade-offs in the circuit and architecture design of FPGAs," in *Int'l Symp. on Field-Programmable Gate Arrays*, Feb. 2008, pp. 149–158.

[8] M. Lin, A. E. Gamal, Y.-C. Lu, and S. Wong, "Performance benefits of monolithically stacked 3-D FPGA," *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, vol. 26, no. 2, pp. 216–229, Feb. 2007.

[9] A. M. Smith, J. Das, and S. J. E. Wilton, "Wirelength modeling for homogeneous and heterogeneous FPGA architectural development," in Int'l Symp. on Field-Programmable Gate Arrays, Feb. 2009, pp. 181– 190.

[10] J. Das, S. J. Wilton, P. Leong, and W. Luk, "An analytical model describing the relationships between logic architecture and FPGA density," in *Int'l Conf. on Field-Programmable Logic and Applications*, Sep. 2009.

[11] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming (web page and software)," Feb. 2009, http://stanford.edu/ ∼boyd/cvx.

[12] S. Joshi and S. Boyd, "An efficient method for large-scale gate sizing," *IEEE Trans. on Circuits and Systems I: Regular Papers*, vol. 55, no. 9, pp. 2760–2773, Oct. 2008.

[13] W. C. Elmore, "The transient analysis of damped linear networks with particular regard to wideband amplifiers," *J. Appl. Phys.*, vol. 19, no. 1, pp. 55–63, Jan. 1948.

[14] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45nm design exploration," in *Int'l Symposium on Quality Elec*tronic Design, 2006. ISQED '06., March 2006, pp. 6 pp.–590.

[15] S.-J. Kim, S. P. Boyd, S. Yun, D. D. Patil, and M. A. Horowitz, "A heuristic for optimizing stochastic activity networks with applications to statistical digital circuit sizing," *Optim Eng*, vol. 8, no. 4, pp. 397– 430, 2007.