FPGA Architecture Optimisation Using Geometric Programming

Alastair M. Smith, *Member, IEEE,* George A. Constantinides, *Senior Member, IEEE,* and Peter Y. K. Cheung, *Senior Member, IEEE*

***Abstract*—This paper is concerned with the application of Geometric Programming to the design of homogeneous FPGA architectures. The paper builds on an increasing body of work concerned with modelling reconfigurable architectures, and presents a full area and delay model of an FPGA. We use a Geometric Programming framework to show how transistor sizing and high-level architecture parameter selection can now be solved as a concurrent optimisation problem. We validate the model through the use of SPICE models and the VPR FPGA architecture simulation tool. Not only does the optimisation framework allow architectures to be optimised orders of magnitude faster than previous work, but the combined optimisation can lead to different architectural conclusions compared to conventional methods by exploring the coupling between the two sets of optimization variables. Specifically, we show that as delay takes more significance in the objective of the optimisation, there should be more the same, or even more lookup tables in a logic block, whereas conventional techniques suggest that there should be fewer lookup tables in an FPGA logic block.**

[1]

***Index Terms*—FPGA, Reconfigurable Architectures, Geometric Programming, Convex optimisation.**

I. Introduction

Recent years have seen considerable evolution in the architecture of FPGAs. Each generation of commercial FPGAs contains new or refined routing, logic, memory and embedded block structures. These architectural enhancements are the result of time consuming and expensive experiments, in which FPGA architects (in both industry and academia) use existing or new CAD tools to map benchmark circuits to the architectures under investigation [1], [2].

Recent work, however, has suggested that this experimental approach can be supplemented by analytical techniques, in which FPGA architectures are modelled by relatively simple equations, and powerful optimisation tools are used to "prune" the architecture space, allowing the FPGA architect to investigate a much wider range of architectures than previously possible [3], [4], [5], [6], [7], [8], [9].

Much of the effort in analytical modelling of FPGA architectures has been on ascertaining FPGA performance or utilisation given a set of high-level parameters describing the logical fabric of the device [3], [4], [5], [6], [7]. However, it is also possible to model effects due to low-level details of the fabric such as transistor sizing [8], [9]. In the context of FPGA design, the application of transistor-level modelling techniques is particularly interesting, as there are a small number of different resource types which are replicated across the fabric. This means that the entire device architecture can be described by a concise set of low-level parameters, aiding the application of formal optimisation.

In this paper, we combine high-level and low-level models and show how they can be made amenable to Geometric Programming (GP), a form of convex optimisation. This allows many logical parameters and most physical parameters of the device architecture to be optimized concurrently.

Specifically, the contributions of this work can be summarized as follows:

- Full details of the framework in [5] and [6] that allows concurrent optimisation of both high-level (architectural) and low-level (transistor sizing) parameters.
- Formulation of an area-delay model of FPGA fabrics as a geometric program.
- Quantification of the area model accuracy using the VPR FPGA simulator and the delay model using the HSPICE simulation environment.

The remainder of this paper is organised as follows. Section II details related work in the field of FPGA architecture modelling and exploration, and provides a brief overview of GP within the domain of digital circuit design. Section III provides details of the FPGA architecture framework studied in this paper. The area and delay models used in this paper are given in Sections IV and V respectively. These models are then mapped into a GP in Section VI. The GP formulation is studied in Section VII, in which we examine the accuracy of the models used and show how GP can be used to make new conclusions about FPGA architectures. The paper is concluded in Section VIII.

II. Related Work

Many research works have been concerned with optimisation of FPGA architectures. The majority of this work has taken an empirical approach to the problem [10], [2], [11]. A typical approach is to develop a parameterisable architecture template and explore the design space by specifying the parameter set for each architecture. The academic tool VPR [12], [1] has been used extensively for this purpose. One of the limiting factors of this approach is that to evaluate the performance of each architecture, synthesis, placement and routing must be performed for a number benchmark circuits, which takes a considerable amount of time; it is common for the CAD process of mapping to a commercial FPGA to take in the order of hours.

[1]This work has been funded by the EPSRC (UK) under grant numbers EP/C549481/1 and EP/E00024X/1. Support from Xilinx is gratefully acknowledged.

TABLE I
SUMMARY OF WORK ON FPGA MODELLING AND TRANSISTOR SIZING.

	Input	Output
[4]	Routing architecture parameters Average wirelength	Estimate of channel width
[3]	Logic architecture parameters Circuit Rent parameter #2-LUTs	Estimate of # FPGA Logic resources
[6]	Logic architecture parameters Circuit Rent parameter, #2-LUTs	Estimate of average wirelength
[7]	Logic architecture parameters Circuit Rent parameter #2-LUTs, 2-LUT depth	Estimate of FPGA circuit depth
[9]	All high-level arch. parameters Transistor sizes Circuit Rent parameter #2-LUTs, 2-LUT depth	Delay estimate of FPGA
[13]	All architecture parameters Circuit netlist	Accurate Area & Delay Transistor Sizing
[14]	Routing architecture parameters	Routing architecture transistor sizing
[8]	Logic architecture parameters Circuit Rent parameter #2-LUTs, 2-LUT depth	Area & Delay Estimate Routing architecture transistor sizing

Recent advances in the FPGA community have sought to mitigate the need for computationally intensive CAD flows by developing closed form equations to model FPGA performance. This work is summarised in Table I, and detailed as follows. In [3], an analytical model was presented that estimates the number of FPGA resources required to implement a benchmark circuit given a number of logic architecture parameters. The work is based on describing a benchmark circuit by its Rent parameter [15], and the number of 2-input logic functions required to implement it. [7] builds on this model by introducing the depth of the circuit netlist between registers as an extra variable in the model. This extra parameter is used to estimate the number of computational resources on the critical path of a benchmark circuit implemented on an FPGA.

Work on the routing architecture has also received some interest, for example [4] presented a model that estimates the number of routing tracks (channel width) required to successfully route a design on an FPGA. The model requires parameters that describe the routing architecture as an input, and assumes all designs have the same wirelength. In [6], a model was developed to estimate the wirelength used by designs implemented on heterogeneous FPGAs and also increased the accuracy of the channel width model in [4]. Throughout the rest of this paper, we combine the routing models with those in [3] and [7] to develop a complete area and delay model of FPGA fabrics, and include details such as optimal transistor sizing.

In [9] a model for the delay of FPGAs was presented. The model uses a transistor-level delay model on top of the circuit depth models presented in [7]. The transistor sizing is assumed constant regardless of the high-level architecture details, however some experimentation is done to find a good value. The details of the model are very similar to the delay model in [8] and the work we present here. However, we allow transistor sizing to be optimised for each set architecture parameters, include an area model and allow the designer to target a combination of area and delay. Moreover, we allow some architecture parameters to be optimised concurrently to the transistor sizing.

Transistor sizing for FPGAs has also recently received some attention from the research community. In [13] a tool was developed that optimizes the electrical design (transistor sizing) of FPGA architectures given a set of high-level architecture parameters. [13] presents a heuristic for optimising the electrical design based on an iterative procedure involving successive placement and routing of benchmarks onto FPGA architectures. The inclusion of benchmark information on top of the architecture specification during this procedure means that the performance of designs implemented on the architecture are taken into account during the optimisation. However, this successive refinement requires the use of CAD tools, and thus takes a considerable amount of time (a maximum of 12 hours is reported in the paper). In our work, we remove the need for iterative refinement involving CAD flows by leveraging recent advances in FPGA modelling techniques and introduce a geometric program to perform the step of transistor size optimisation concurrently to a number of architectural parameters.

Transistor sizing for FPGA interconnect has also been studied in [14]. The paper presents a model for the purpose of transistor sizing of interconnect buffers and a heuristic method was developed to optimize the sizing. However, it has been shown in [16] that transistor sizing, including that presented in [14] can be solved efficiently and to optimality through use of geometric programming. We employ such techniques in this paper.

A. Geometric Programming

A *Geometric Program* (GP) is a constrained optimisation problem of the following form:

$$\begin{aligned} &\text{Minimize :} && f_0(x) \\ &\text{Subject to :} && \\ & && f_i(x) \leq 1, \text{ for } i = 1, 2, ..., m \quad (1) \\ & && g_i(x) = 1, \text{ for } i = 1, 2, ..., l \quad (2) \end{aligned}$$

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

A monomial is a function

$$g(x) = cx_1^{a_1}x_2^{a_2}\dots x_n^{a_n}$$

where the coefficient c must be strictly positive and all exponents a_i are real valued constants. A posynomial is simply a sum of a finite number of monomials.

Geometric programming has been used extensively for many circuit design problems. Example problems addressed by GP include transistor sizing [17], wire sizing [18] and robust design in the presence of statistical variations [19]. We refer the reader to [16] for an extensive review of GP in the context of circuit design. One of the particularly attractive features of GP is that it has excellent tractability. Interior point methods are used to solve GP with polynomial run-time with respect to the number of variables. By comparison, parameter-sweep methodologies for exploring design spaces have exponential run-time with respect to the number of variables. This means

that the GP approach deployed in this paper offers a significant advantage in compute time over traditional methodologies.

Our previous work in [5] made the observation that some high-level models of FPGA fabrics also fit into the GP framework. For example, we made the observation that FPGA routing fabrics consist predominantly of multiplexers, and that the area and number of these multiplexers can be expressed as a GP, leading to modest area savings. Furthermore, in [8] we showed that by employing GP transistor sizing techniques such as those discussed above, area, delay and a combination of the two can be optimised in conjunction with some high-level architectural parameters. In this work, we present these models in their complete form for the first time, and include comprehensive experimental verification of their accuracy.

III. Model Framework

The modelling framework we present consists of a number of parts. In Section III-A, we define the target architecture style, which is based on lookup tables as the basic logic element, and with a number of variable parameters. The basic representation of the circuit is defined in Section III-B, which is used as the input to our model. The model has to use the information about each circuit in conjunction with the information about the architecture to estimate the number of computational resources used by the FPGA architecture, and the number of resources on the critical path. This information is used along with transistor-level details to obtain accurate area information in Section IV and timing information in Section V. Details of how the model is cast as a GP are given in Section VI.

A. Architecture Framework

Throughout this paper we assume an island-style FPGA in which an array of blocks is connected using tracks organized in horizontal and vertical channels with single-driver routing, as represented by VPR 5.0 [1]. There are five high-level architecture parameters that we study in this work, which are summarised in the top half of Table II and are explained below. The logic blocks in the architecture consist of K-input lookup tables (LUTs) packed into tightly connected configurable logic blocks (CLBs), each with N LUTs and with I external inputs, as shown in Figure 1(a). Further details of this logic block architecture are available in [2], Section 3.1.1. A K-input LUT cab be implemented using a K-level pass transistor multiplexer tree, as shown in Figure 1. Further details of this structure can be found in [2], Section B 1.2.

The logic architecture parameters N, K and I impact the number of logic blocks required to implement a circuit. For example a 7-input LUT has a considerably larger logic density per block than a 2-input LUT, hence fewer CLBs will be required in an architecture containing 7-LUTs. Similarly a large value of N implies fewer architecture blocks are required, as the CLBs have increased capacity. Due to the internal structure of a CLB, the tradeoff between these three parameters is not straightforward; the most recent study of this tradeoff is in [1].

TABLE II
Model Parameters.

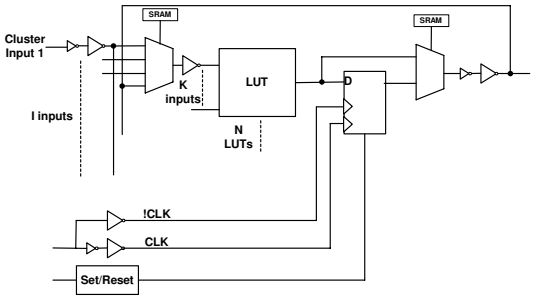
High-level Architectural Parameters:	
K	Number of inputs per lookup table
N	Number of lookup tables per logic block
I	Number of inputs per logic block
$F_{c,in}$	Number of tracks that connect to each logic input pin
$F_{c,out}$	Number of tracks each logic block can connect to
F_s	Number of track end-points 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

The routing architecture is used to connect signals between logic resources, as shown in Figure 1(c). Routing architectures in FPGAs have changed considerably since [2], and modern commercial FPGAs use single-driver routing [20]. These routing architectures consist of connection boxes and switch boxes that drive and terminate wire tracks. The number of tracks in each channel (vertical or horizontal) of the routing architecture is known as the channel width, W. We assume a single driver routing architecture [20] in which channels are directional, hence in each channel half of the tracks are directed in one direction (north to south, or east to west) and half in the opposite direction (south to north or west to east). The routing blocks in the architecture can be described by three parameters: the number of tracks that can connect to each logic block input, $F_{c,in}$; the number of tracks that each logic block output can connect to, $F_{c,out}$; and the number of track end-points that connect to each channel driver, F_s. The routing flexibility parameters $F_{c,out}$, $F_{c,in}$ and F_s impact the number of tracks required, the relationship is discussed in Section IV-B.

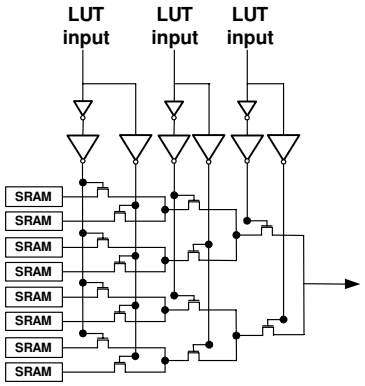
All connection box multiplexers in the FPGA are implemented using a two-stage pass transistor structure, as used in VPR 5.0. This type of multiplexer provides a balance between area and delay: relative to a single stage pass transistor multiplexer, SRAM bits are saved by encoding the multiplexers this way, however the delay in a two stage multiplexer is lower than the binary tree implementation used in the LUT multiplexer. The two-stage multiplexer, including a transistor-level implementation is shown in Figure 2. In each case, the multiplexers are used to configure signal routing paths around the device, and thus the select lines are connected to SRAM configuration memory.

B. Circuit Representation

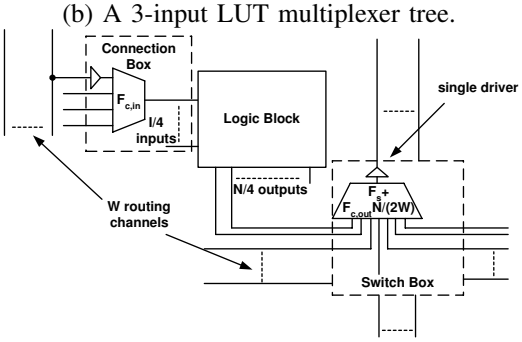
Each benchmark circuit we use in our experimentation is defined by a circuit netlist. However a model is also needed. In this work three parameters are used to describe the circuit model: n_2, the number of 2-LUT primitives that can be used to implement the circuit, d_2 the maximum number of gates between flip-flops and p, the Rent parameter. The Rent parameter is from a well-known empirical rule that defines the complexity of connections between circuit primitives [15]. This parameter is obtained through an initial placement of the 2-LUT netlist, and the same value is used for all experiments. Benchmark circuit parameters are given in the lower half of Table II.



(a) Structure of a configurable logic block.



(b) A 3-input LUT multiplexer tree.



(c) Routing Fabric of an FPGA.

Fig. 1. Detailed view of FPGA architecture.

TABLE III
NOTATION USED IN THE AREA MODEL, '*' DENOTES A NUMBER, DEPENDING ON THE NUMBER OF INVERTERS USED IN A DRIVER.

Variables	
Symbol	Meaning
A_{total}	Total FPGA area
A_l, A_r	Total FPGA logic and routing areas
A_{CLB}	Total area of a CLB
A_{lut}	Area of LUT multiplexer, buffers and config memory
A_{21mux}	Area of the 2:1 multiplexer in the logic block
A_{LB}	Combined area of a logic element within a CLB
A_{rst}	Area of the CLB reset logic
B_{li}, B_{lo}	Area of LUT input/output buffer
$B_{cb,clb/io}$	Area of buffer for CLB/IO input connection MUX
$B_{sb,m/e}$	Buffer Area - switch box MUX (middle/edge)
W	Channel width of device
W_{min}	Minimum nominal channel width of device [4]
D_r	Average point-to-point wirelength
n_k,n_c	Number of k-LUTs/CLBs required for benchmark
N_c	Number of CLBs required in the FPGA grid
$S_{n,21mux}$	Scaling of pass transistors in the 2:1 MUX
$S_{n,LM}$	Scaling of pass transistors in the LUT multiplexer
$S_{n,SB}$	Scaling of pass transistors in the switch box MUX
$S_{n,CB}$	Scaling of transistors in the connection box MUX
$S_{n/p,LIdrv*}$	Scaling of n/pMOS transistor in LUT I/P driver
$S_{n/p,LOdrv*}$	Scaling of n/pMOS transistor in LUT O/P driver
$S_{n/p,SBdrv*}$	Scaling of n/p transistors in switch box driver
$S_{n/p,CBdrv*}$	Scaling of n/p transistors in connection box driver
$E_{LS,tree}$	No. of pass transistors in the LUT input MUX tree
$E_{LS,RAM}$	SRAM bits required for the LUT input MUX tree
A_{CB}	Area of all input connection boxes
$A_{CB,clb/io}$	Area of CLB/IO input connection box MUX
A_{SB}	Area of all switch boxes
$A_{SB,m/e}$	Switch box area inc. buffer (middle/edge)
$N_{s,m/e}$	No. switching points on middle/edge of the array
Constants	
f_p	empirical constant [4]
β	empirical constant from [4]
α_{in}	empirical constant from [4]
α_{out}	empirical constant from [4]
λ	average number of used inputs per logic block [4]
A_{reg}	Area of a single D flip-flop
A_{clkB}	Area of the clock buffer in a CLB
S_{SR}	Size of an SRAM cell

TABLE IV
A COMPARISON OF THE NUMBER OF UNUSED LUT INPUTS γ AND THE APPROXIMATION USED IN THIS WORK.

K	2	3	4	5	6	7
γ	0	0.261	0.466	0.701	0.996	1.232
$0.25K - 0.5$	0	0.25	0.5	0.75	1.0	1.25

IV. AREA MODEL

The area model is based on evaluating the total minimum width transistor area, as used in [12]. This is the same metric as used in GP area minimisation work [16]. The total area of an FPGA, A_{total}, can be represented as the sum of the routing area A_r and logic area A_l, as in (3). The routing area is the sum of all transistor sizes in the switch box and connection boxes, while the logic area consists of all transistors in the CLBs. Resources implementing routing signals on to and off chip are not considered, but the connection boxes to these resources are. For reference, the variables used in the area model are given in Table III.

$$A_{total} = A_l + A_r \tag{3}$$

In our experiments in Section VII, we apply the model on a per benchmark basis by creating an architecture for each benchmark, a commonly employed method in FPGA architecture evaluation [2]. The size of the FPGA is assumed to be the smallest square that fits the benchmark circuit. Thus, the grid size $N_c = \lceil \sqrt{n_c} \rceil^2$, where n_c is the number of CLBs in the benchmark circuit. This quantity can be estimated using the formula in [3]. Throughout this paper we use this model to estimate the number of CLBs. For the majority of the design space, this model can be stated as in (4) and (5), where n_2 represents the number of 2-input LUT primitives that describe the benchmark circuit, n_k represents the number of k-input lookup tables required to implement the benchmark circuit, and p is the Rent parameter of the circuit. γ is an empirically derived parameter that represents the average number of unused LUT inputs. These values are given in Table IV. A good approximation to this term is $\gamma = 0.25K - 0.5$, which is also given in Table IV and in turn leads to the simplification in (5).

$$n_c = \frac{n_k}{N} \tag{4}$$

$$n_k = n_2 \left(\frac{3}{K + 1 - \gamma} \right)^{\frac{1}{p}} \approx n_2 \left(\frac{3}{1.25K + 0.5} \right)^{\frac{1}{p}} \tag{5}$$

We discuss the constituent parts of the logic block area in Section IV-A. The routing area model used in this work was

first presented in [5]. We consider the amount of silicon area devoted to the routing fabric to consist of all switch box and connection box multiplexers, in addition to their output buffers and configuration memories.

A. Logic Block Area

The combined area of all logic blocks in the architecture can be stated as in (6), and is simply the number of CLBs multiplied by the area that each CLB consumes, A_{CLB}.

$$A_l = N_c A_{CLB} \tag{6}$$

The area of the logic block is the sum of the area devoted to the following: the LUT, 2:1 multiplexer, register and output buffer combination, the LUT input select multiplexer, the set/reset logic and the clock buffer. We assume the set/reset logic and clock buffer sizing to be constant regardless of the logic block architecture, with values from [2]. Similarly, the size of the register on the LUT output is assumed to be a pre-defined constant.

A K-input LUT is constructed as a K-level multiplexer, as shown in Figure 1(b). The area is composed of the SRAM cells, the pass-transistor multiplexer cells and the internal drivers. The widths of the pass transistors in the multiplexer are given by $S_{n,LM}$, each pass transistor has the same width. Similarly, each LUT input buffer is assumed to have the same transistor sizing. The sum of these buffer areas is given by B_{li}. This leads to (7) as an expression for the area consumed by a K-input LUT, where S_{SR} is the size of an SRAM cell and B_{li} is the size of the buffer driving each LUT input and is the sum of six transistors used to implement the inverting buffers as shown in (8), where the area of $S_{n,LIdrv*}$ represents the size of each nMOS transistor in the CMOS inverter and $S_{p,LIdrv*}$ represents the size of each pMOS transistor.

$$A_{lut} = 2^K S_{SR} + K B_{li} + (2^{K+1} - 2) S_{n,LM} \tag{7}$$

$$B_{li} = S_{n,LIdrv1} + S_{p,LIdrv1} + S_{n,LIdrv2} + S_{p,LIdrv2} + S_{n,LIdrv3} + S_{p,LIdrv3} \tag{8}$$

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

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

The 2:1 multiplexer in the CLB consists of a one level pass transistor multiplexer. This consumes area A_{21mux}, given by (9), where $S_{n,21mux}$ represents the size of each of the two pass transistors implementing the 2:1 multiplexer, and S_{SR} is the one bit configuration memory required.

The CLB output buffer combination is the sum of the transistor areas for the two inverters implementing the driver. The combination of these inverters consumes the area given in (10), where the area of $S_{n,LOdrv*}$ represents the size of each nMOS transistor in the CMOS inverter and $S_{p,LOdrv*}$ represents the size of the pMOS transistor.

The LUT input select multiplexer is implemented with the two-level multiplexing scheme. An example of a 16:1 multiplexer using this scheme is shown in Figure 2(a). In such a scheme, the first and second level of multiplexing is balanced

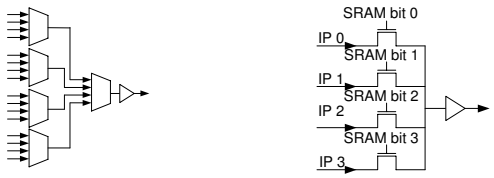


Fig. 2. Multiplexing schemes in VPR 5.0: (a) a two-level multiplexing scheme for a 16:1 multiplexer, (b) a pass transistor-based 4:1 multiplexer.

such that each stage has multiplexers with approximately the same number of inputs. Since the output of one of the multiplexers in the first stage is fed to the output, the SRAM configuration cells can be shared across each multiplexer in order to save area. Multiplexers can be efficiently implemented using pass transistors, as shown in Figure 2(b). This structure is taken from VPR 5.0, in which one-hot encoding of configuration bits is used. These observations lead to the expression for multiplexer area given in (11); S_n represents the size of the pass transistors and E is the number of inputs. An approximation for this area is given in (12), which is used for the routing fabric multiplexers.

Each of the input select multiplexers is fully connected; every input from the connection box and every output feedback path can reach any LUT input [2], Section 3.1.1. Using the exact expression for multiplexer area leads to the expression in (13) for the area devoted to the input select multiplexer, where $S_{n,LSmux}$ represents the size of the pass transistors implementing the input select multiplexer. Since there are $I + N$ inputs to the multiplexer, $E_{LS,tree}$ in (14) represents the number of pass transistors in the multiplexer tree and $E_{LS,tree}$ in (15) represents the number of SRAM bits.

Combining the above expressions for the constituent parts of the logic block leads to the expression for logic block area in (16).

$$A_{mux} = S_n \left(E + \lfloor \sqrt{E} \rfloor \right) + S_{SR} \left(\left\lceil \frac{E}{\lfloor \sqrt{E} \rfloor} \right\rceil + \lfloor \sqrt{E} \rfloor \right) \tag{11}$$

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

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

$$E_{LS,tree} = N + I + \lfloor \sqrt{N + I} \rfloor \tag{14}$$

$$E_{LS,RAM} = \left\lceil \frac{N + I}{\lfloor \sqrt{N + I} \rfloor} \right\rceil + \lfloor \sqrt{N + I} \rfloor \tag{15}$$

$$A_{LB} = N A_{LUT} + N A_{reg} + N A_{21mux} + K N A_{LSmux} + N B_{lo} + A_{clkB} + A_{rst} \tag{16}$$

B. Routing Area

The routing area is given by (17) and is the sum of the area devoted to switch box multiplexers A_{SB} and the area devoted to connection boxes A_{CB}.

$$A_r = A_{CB} + A_{SB} \tag{17}$$

Each logic resource on the FPGA has multiplexers to connect signals from the routing tracks to each of the logic block input pins. We assume that each CLB input pin has an identical size connection box multiplexer. Similarly, each I/O

block has the same size multiplexer, which may be different to that of the CLB connection box. The combined area of connection boxes is thus given by (18), where I_{clb} and I_{io} are the number of CLB and I/O inputs respectively, $A_{CB,clb}$ and $A_{CB,io}$ are the areas of each connecting multiplexer and $4\sqrt{N_c}$ represents the number of I/O blocks, which are on the perimeter of the FPGA grid.

$$A_{CB} = N_c I_{clb} A_{CB,clb} + 4\sqrt{N_c} I_{io} A_{CB,io} \tag{18}$$

The switch boxes consist of multiplexers that are responsible for routing signals around the FPGA and driving the wire tracks. The outputs of logic resources, I/O pins and the end points of routing tracks are all fed into the switch box multiplexers. There is one multiplexer and one driver for each track [1]. Since I/O blocks are spread around the edge of the array, the switch box multiplexers around the perimeter of the logic array have a different number of inputs to those in the center of the array. $N_{s,e}$ and $N_{s,m}$ represent the number of points in the array where switch boxes occur. These numbers are dependent on the FPGA size, and as the routing grid is one unit of width larger than the logic grid $N_{s,e} = 4\left(1 + \sqrt{N_c}\right)$ and $N_{s,m} = (\sqrt{N_c} - 1)^2$.

The combined area of all switching resources is given by (19). In this equation, $2W$ represents the number of channels in the center of the grid (horizontal plus vertical), $1.5W$ represents the number of channels on the edge of the grid (reduced by one direction of four) and the areas of each multiplexer and its output driver are given by $A_{SB,e}$ and $A_{SB,m}$ for the edge and middle respectively. The areas of these multiplexers and their drivers are detailed below.

$$A_{SB} = 1.5W N_{s,e} A_{SB,e} + 2W N_{s,m} A_{SB,m} \tag{19}$$

The model developed in [4] is used to estimate channel width. This model is shown in (20) for architectures with wires that span one logic block, where the nominal minimum channel width W_{min} is described by (21), and β, α_{in}, α_{out} and f_p are empirically derived constants. In (21), λ represents the average number of inputs used on each logic block and D_r represents the average point-to-point wirelength. We note that this model is easily extendable to architectures with wires that span more than one logic block through use of an extended form of the channel width expression given in (20). However, we are restricted to channels with a single segment length and single-driver routing. The channel width model would have to be re-derived for more detailed routing architectures.

$$W = W_{min} + \frac{1}{\beta}\left(\frac{W_{min}}{F_s}\right)\left(\frac{W_{min}}{F_{c,in}}\right)^{\alpha_{in}}\left(\frac{W_{min}}{F_{c,out}}\right)^{\alpha_{out}} \tag{20}$$

$$W_{min} = f_p \frac{\lambda D_r}{2} \tag{21}$$

The methods described by [6] are used to calculate the value of point-to-point wirelength for different logic parameters. It is based on [21], but differs by taking into account the differing number of logic blocks as a result of technology mapping to varying LUT and CLB sizes. This wirelength is given by D_r in (22).

$$D_r = \frac{2\sqrt{2}(3+3p)}{(1+2p)(2+2p)} n_c^{(p-0.5)} \tag{22}$$

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

Using this approximation the area of a connection box multiplexer can be expressed by (23). The number of inputs to each connection box multiplexer can be specified by the architecture parameter $F_{c,in}$, or alternatively $F'_{c,in} = \frac{F_{c,in}}{W}$, which represents the proportion of tracks that can connect to each logic block input. A different multiplexer flexibility parameter may be given for I/O blocks and CLBs, however in this work we consider them to be the same.

The approximation for the area of a single switch box multiplexer is given by (24). F_s represents the number of routing tracks that are inputs to each multiplexer and $F_{c,out}$ is the architecture parameter that describes how many tracks each logic block (or I/O block) can reach. Alternatively $F'_{c,out} = \frac{F_{c,out}}{W}$ represents the proportion of routing tracks to which each logic block output connects. The factor $\frac{N}{2}$ represents the fact that the N outputs from each CLB are spread around four sides of the device, and that each multiplexer has inputs from two adjacent logic blocks due to the track directions. The switch boxes on the edge of the device have a slightly different expression to reflect that I/O blocks may have a different number of outputs to CLBs.

The buffer area at the output of each multiplexer must also be considered. Each buffer is constructed of two cascaded inverters meaning that the area is the sum of the four transistors in the same way as (10). Thus the areas of the multiplexer/buffer combinations are given by (25-28). These are used in (18) and (19) to evaluate the total routing area.

$$A_{CBmux,clb} = S_{n,cb}\left(WF'_{c,in} + \sqrt{WF'_{c,in}}\right) + 2S_{SR}\sqrt{WF'_{c,in}} \tag{23}$$

$$A_{SBmux,m} = S_{n,sb}\left(\frac{N}{2}F'_{c,out} + F_s + \sqrt{\frac{N}{2}F'_{c,out} + F_s}\right) + 2S_{SR}\sqrt{\frac{N}{2}F'_{c,out} + F_s} \tag{24}$$

$$A_{CB,clb} = A_{CBmux,clb} + B_{cb,clb} \tag{25}$$

$$A_{CB,io} = A_{CBmux,io} + B_{cb,io} \tag{26}$$

$$A_{SB,m} = A_{SBmux,m} + B_{sb,m} \tag{27}$$

$$A_{SB,e} = A_{SBmux,e} + B_{sb,e} \tag{28}$$

V. Delay Model

GP has previously been shown to be capable of optimising transistor sizing for delay [19]. We employ this type of delay modelling technique here to represent the combination of CMOS and pass transistor structures present in FPGA devices. In this work we focus on the delay of the transistor elements.

TABLE V
NOTATION USED IN THE DELAY MODEL, '*' DENOTES A NUMBER, DEPENDING ON THE NUMBER OF INVERTERS USED IN A DRIVER.

Variables	
Symbol	Meaning
D_k	Max. no. of LUTs on paths between registers
D_c	Max. no. of CLBs on paths between registers
D_i	No. of internal connections on critical path
T_{total}	Total critical path delay
Resistances/capacitances, $X = D$ implies the diffusion value and $X = G$ implies a gate capacitance	
$R/C_{X,CBmux}$	R/C of pass transistor in connection box MUX
$R/C_{X,SBmux}$	R/C of pass transistor in switch box MUX
$R/C_{X,LSmux}$	R/C of pass transistor in LUT input select MUX
$R/C_{X,21mux}$	R/C of pass transistor in the 2:1 MUX
$R/C_{X,Lmux}$	R/C of pass transistor in the LUT MUX
$R/C_{X,CB\_dr*}$	R/C of connection box driver
$R/C_{X,SB\_dr*}$	R/C of switch box driver
$R/C_{X,LO\_dr*}$	R/C of LUT output driver
$R/C_{X,LI\_dr}$	R/C of LUT input driver
$R/C_{X,LDint*}$	R/C of internal LUT driver
$R/C_{X,reg\_op}$	R/C of DFF output
C_{X,reg_ip}	R/C of DFF input
Constants	
$R_{nom,D,p/n}$	Nominal diffusion res. of p/nMOS transistor
$C_{nom,D,p/n}$	Nominal diffusion cap. of p/nMOS transistor
$C_{nom,G,p/n}$	Nominal gate cap. of p/nMOS transistor
S_{TECH}	Minimum feature size of process

We have simplified the problem by assuming that track and wiring delays are not accounted for, however, we note that the techniques employed here can also be used for wiring capacitance calculation, and also fit into a GP framework. In addition to the variables in Table III, variables used in this section are given in Table V.

Each transistor in the circuit can be represented as an RC network as in Figure 3. We use a commonly employed model for the resistance and capacitance values within a MOSFET [16]: each resistance value for a transistor in the architecture takes the form (29) and each capacitance takes the form (30) or (31). $R_{D,x}$ represents the channel resistance of transistor x, and $C_{G,x}$ and $C_{D,x}$ represent the gate and diffusion capacitances respectively. In each of these equations S_i refers to the width of the transistor assuming all transistors have minimum length. The nominal values $R_{nom,D}$, $C_{nom,D}$ and $C_{nom,G}$ are dependent on the type of transistor (nMOS/pMOS), the process technology and in the case of the capacitance, whether it is the nominal gate or diffusion capacitance.

$$R_D = \frac{R_{nom,D}}{S_i} \tag{29}$$

$$C_G = C_{nom,G} S_i \tag{30}$$

$$C_D = C_{nom,D} S_i \tag{31}$$

$$T_{elmore} = \sum_{paths\ i\ source\ to\ sink} C_i R_{source\ to\ C_i} \tag{32}$$

The nominal values of resistance and capacitances are be derived from SPICE models of MOSFET devices. In this work, we derive the values using 65nm predictive technology models [22], making approximations from the BSIM4 modelling framework [23].

To evaluate the delays through the pass transistor networks, the Elmore delay model is employed [24]. The Elmore model is used to represent delay in networks of RC trees and has previously been shown to model delay in FPGA routing pass transistor networks [14]. As shown in (32), Elmore delay is calculated through the evaluation of the sum of each segment delay from the signal source to its sink, where the delay of each segment is the sum of the resistance along the path multiplied by the output capacitance of that segment. Expressions of this form are posynomial, and naturally fit into a GP framework, allowing transistor sizes to be optimised.

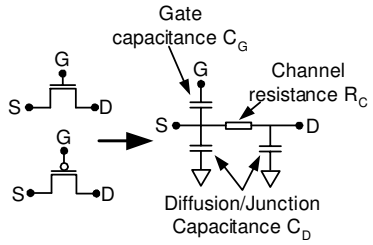


Fig. 3. RC delay model for a MOSFET.

A. Delay in FPGAs

Consider the delay of the critical signal path through a circuit implemented on an FPGA. The critical path will emanate from a register and typically passes through a number of LUTs, CLBs, switch boxes, connection boxes, CLB feedback paths and the multiplexers and buffers associated with each of these. We model the critical path delay as a weighted sum of the various circuit components; the value of each weight depends on the expected number of each of these components along the path. We use the formula in (33), where each term is as described below. D_k represents the depth of the netlist when implemented in K-input LUTs: we assume the critical path is through the deepest part of the netlist. D_c represents the number of CLBs through which the critical path travels. $D_i = D_k - D_c$ represents the number of internal feedback connections through which the critical path traverses. Finally, D_r represents the number of switch boxes through which each external connection on the critical path propagates.

$$\begin{aligned} \mathrm{T_{total}} = {} & T_{reg\ to\ ODrv} + D_i T_{LUT\ F/B\ path} \\ & + (D_k - 1) T_{LUT\ delay} + D_c T_{O/P\ CB\ delay} \\ & + D_c D_r T_{SB\ delay} + D_c T_{I/P\ CB\ delay} \\ & + D_c T_{input\ MUX\ delay} + T_{LUT\ to\ reg\ delay} \end{aligned} \tag{33}$$

Each of the scaling factors can be determined through use of analytically derived mathematical expressions. The average wirelength is assumed to determine the number of switch boxes through which each external CLB connection travels, which is represented by D_r and is the average wirelength, the equation for which is given in (22). The equations governing D_k, D_c and D_i, which are taken directly from [7] are given in (34)-(36).

$$D_k = \frac{2 d_2}{K - 1 - \gamma + \log_2(K - \gamma)} \tag{34}$$

$$D_c = d_k \left[1 - \frac{(N-1) + \frac{N}{n_k}\left[N(K-\gamma) - N + 1\right]}{c(K-\gamma)} \right] \tag{35}$$

$$D_i = D_k - D_c \tag{36}$$

In order to derive an expression for the total delay, we must break down each of the delay terms T_x in (33) into its constituent paths. Each path starts from VDD or GND and terminates at a transistor gate input. This leads to a number of paths, given in Figure 4(a-i).

The value $T_{reg\ to\ ODrv}$ represents the delay from the register producing the critical path signal through the MUX selecting whether the LUT output is registered or not, and through the two-level inverting driver. This is given in Figure 4(f).

The term $T_{LUT\ F/B\ path}$ represents the delay from the BLE output buffer through the pass transistor-based MUX on the LUT input to its buffer, as shown in Figure 4(b). The Elmore delay path in this case terminates at the LUT driver input gate.

The delay $T_{LUT\ delay}$ represents the delay from the LUT driver through all levels of the multiplexer implementing the LUT, the 2:1 select MUX and to the LUT output driver. The delay is thus the sum of the paths these paths, for which transistor-level diagrams of these are given in Figure 4(g), and Figure 4(c) respectively.

The delay $T_{O/P\ CB\ delay}$ represents the delay from the logic element's output buffer, through the switch box multiplexer to its first inverting buffer, as represented by the transistor diagram in Figure 4(b), where in this case the Elmore delay is through the path to the switch box driver.

$T_{SB\ delay}$ represents the routing path signal between switch boxes. This is represented by the sum of the driver delay and delay through the two-level switch box multiplexer, as in Figure 4(h) and (a).

The term $T_{I/P\ CB\ delay}$ represents the path through the switch box to the connection box where the routed signal is consumed. This is the sum of the initial driver delay, the delay through the connection box multiplexer and through the two inverting drivers in the connection box. These are shown in Figure 4(h), (a) and (i) respectively.

$T_{input\ MUX\ delay}$ is the delay from the connection box output driver through the LUT input select multiplexer to the LUT input driver, as represented in Figure 4(e).

Finally, $T_{LUT\ to\ reg\ delay}$ is the delay through the LUT driver, then the multiplexer implementing the LUT and to the register input where the critical path terminates, as in Figure 4(g) and (c).

Since there are two paths to consider for each driving gate: the charge and discharge path from the driving gate, the terms representing delay are given as inequalities. Due to space limitations it is not possible to list all of the Elmore delay equations. However, as an illustrative example, the inequality for the delay path that represents $T_{O/P\ CB\ delay}$ is given by (37) and (38), represented in Fig. 4(b). In this case (37) represents the path through the nMOS transistor in the driver and (38) the path through the pMOS transistor.

$$\begin{aligned} T_{O/P\ CB\ delay} \geq & R_{D,n,LOdrv2}(C_{D,n,LOdrv2} \\ & + NKC_{D,LSmux} + F_{c,out}C_{D,SBmux}) \\ & + (R_{D,n,LOdrv2} + R_{D,SBmux})2C_{D,SBmux} \\ & + (R_{D,n,LOdrv2} + 2R_{D,SBmux}) \\ & \times (C_{D,SBmux} + C_{G,SBdr1}) \end{aligned} \tag{37}$$

$$\begin{aligned} T_{O/P\ CB\ delay} \geq & R_{D,p,LOdr2}(C_{D,p,LOdrv2} \\ & + NKC_{D,LSmux} + F_{c,out}C_{D,SBmux}) \\ & + (R_{D,p,LOdrv2} + R_{D,SBmux})2C_{D,SBmux} \\ & + (R_{D,p,LOdrv2} + 2R_{D,SBmux}) \\ & \times (C_{D,SBmux} + C_{G,SBdr1}) \end{aligned} \tag{38}$$

VI. Geometric Programming formulation

In this section, the model presented above is translated into a form that is amenable to GP. Specifically, it is necessary to show that the model can be expressed as constraints in posynomial and monomial form, as in (1) and (2). The GP we present must be solved separately for each combination of I, K and N. It has not yet been possible to express constraints involving these three terms, which is predominantly due to the existence of negative coefficients in the expressions for circuit depth (35), used in the delay model.

We use the cost function (39) to minimise a monomial function of area and delay. By varying the exponent weight z it is possible to target, for example, delay only by setting $z = 1$, or area-delay product by setting $z = 0.5$.

$$\min : T_{total}^{z} A_{total}^{1-z} \tag{39}$$

A. Area Constraints

We focus here on presenting the area constraints that do not immediately appear to be in posynomial form. Many of the equations in the model presented thus far are easily expressed in posynomial form, for example the equation describing channel width (20) can be expressed as a posynomial by the simple rearangement (42). This is equivalent to an equality constraint because the problem is one of minimisation.

Buffer area is not included in (42), however, the expression for the area of buffer in each case resembles (40), where x represents any of the buffers, for example the internal LUT buffers in Figure 4(g), which is the sum of the area of three inverters. These buffer constraints is mapped into a GP constraint in a straightforward manner, as in (41).

$$B_x = \sum_{\text{all inverters in } x} S_n + S_p \tag{40}$$

$$\implies \sum_{\text{all inverters in } x} S_n B_x^{-1} + S_p B_x^{-1} \leq 1 \tag{41}$$

The expression for the area of a switch box multiplexer (24) cannot be expressed directly in posynomial form, as there is a square root over a sum. However, the terms in the sum can be replaced by a single variable Q, as in (44) and a new constraint can be introduced to represent the sum, as in (43).

$$W_{min}W^{-1} + \beta^{-1}F_s^{-1}W_{min}^{(1+\alpha_{in}+\alpha_{out})}W^{-1}F_{c,in}^{-\alpha_{in}}F_{c,out}^{-\alpha_{out}} \leq 1 \tag{42}$$

$$\frac{N}{2}F'_{c,out}Q^{-1} + F_sQ^{-1} \leq 1 \tag{43}$$

$$S_{n,SB}A_{SB,m}^{-1}Q + S_{n,SB}A_{SB,m}^{-1}Q^{\frac{1}{2}} + 2S_{SR}A_{SB,m}^{-1}Q^{\frac{1}{2}} \leq 1 \tag{44}$$

$$\frac{N}{4}F'_{c,out}P^{-1} + I_{io}F'_{c,out}P^{-1} + F_sP^{-1} \leq 1 \tag{45}$$

$$S_{n,SB}A_{SB,e}^{-1}P + S_{n,SB}A_{SB,e}^{-1}P^{\frac{1}{2}} + 2S_{SR}A_{SB,e}^{-1}P^{\frac{1}{2}} \leq 1 \tag{46}$$

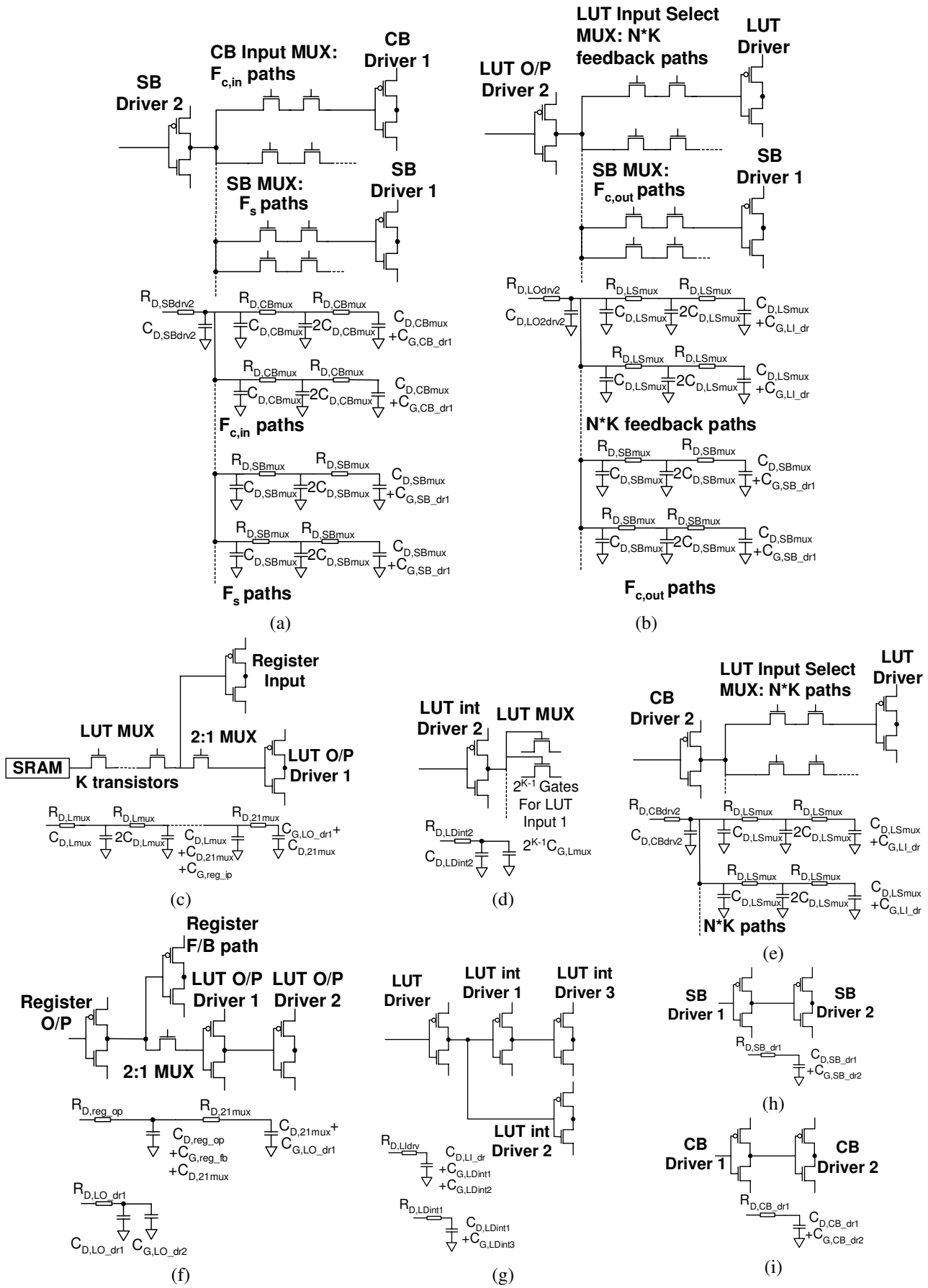


Fig. 4. RC delay models for circuit path, transistor variables are summarised in Table V.

B. Delay Constraints

The mapping of delay constraints is relatively straightforward, as they take posynomial form. An example of how delay constraints are represented in a GP are given in (47)-(56). The example shows the delay and related constraints between the inverters used in the switch box buffer in Figure 4(h). (47) and (48) represent the charge/discharge path of the first inverter through pMOS and nMOS transistors respectively, where $T_{SB\ inv1\ inv2}$ is the variable representing the delay. C_{G,SB_inv2} is the load capacitance of the second inverter gate, and is the sum of the two transistor gates used to make up the inverter - the sum is expressed in (49). The capacitance and resistance values required are given by (50)-(55). (56) is used to ensure the transistor size does not violate the smallest feature size possible in the process technology, where S_{TECH} is the constant representing the minimum feature size. This final constraint must be applied to all transistors in the GP.

$$T^{-1}_{SB\ dr1\ dr2} R_{D,n,SBdr1} C_{D,n,SBdr1} + T^{-1}_{SB\ dr1\ dr2} R_{C,n,SBdr1} C_{G,SBdr2} \leq 1 \quad (47)$$

$$T^{-1}_{SB\ dr1\ dr2} R_{D,p,SBdr1} C_{D,n,SBdr1} + T^{-1}_{SB\ dr1\ dr2} R_{C,n,SBdr1} C_{G,SBdr2} \leq 1 \quad (48)$$

$$C^{-1}_{G,SBdr2} C_{G,p,SBdr2} + C^{-1}_{G,SBdr2} C_{G,n,SBdr2} \leq 1 \quad (49)$$

$$R_{nom,D,n} R^{-1}_{D,n,SBdr1} S^{-1}_{n,SBdr1} \leq 1 \quad (50)$$

$$R_{nom,D,p} R^{-1}_{D,p,SBdr1} S^{-1}_{p,SBdr1} \leq 1 \quad (51)$$

$$C_{nom,D,n} C^{-1}_{D,n,SBdr1} S_{n,SBdr1} \leq 1 \quad (52)$$

$$C_{nom,D,p} C^{-1}_{D,p,SBdr1} S_{p,SBdr1} \leq 1 \quad (53)$$

$$C_{nom,G,n} C^{-1}_{G,n,SBdr2} S_{n,SBdr2} \leq 1 \quad (54)$$

$$C_{nom,G,p} C^{-1}_{G,p,SBdr2} S_{p,SBdr2} \leq 1 \quad (55)$$

$$S_{TECH} S^{-1}_{n,SBdr1} \leq 1 \quad (56)$$

C. Summary of the GP

Despite an FPGA requiring in excess of 1 million transistors for our test circuits, are only 150 transistor types in the design that are optimized, as many buffers and pass transistors drive the same size loads, so have the same size. This is due to the regular structure of the FPGA - there are only a small number of functional blocks that are replicated across the array. The 150 transistors in the problem equate to 150 transistor widths and 450 variables to represent the resistance and capacitances. In addition to the high-level parameters, the Geometric Program has in excess of 600 variables. The GP takes approximately 1 minute to run on a Quad Core Pentium 2.6GHz running Windows XP using CVX [25] within MATLAB.

As yet, it has not been possible to all of the constraints for the full critical path delay in posynomial form. The critical path delay in (33) involves variables D_k, D_c, which are a function of architecture parameters N and K. The presence of $\log_2$ and negative coefficients in (34) and (35) respectively are the reason for not being able to express the delay model such that the GP allows N and K to be optimised in a single run of the optimisation. Instead, our experimentation employs a parameter sweep over these values to find an optimal point. Nevertheless, the use of GP reduces a 600+ dimensional design space to two dimensions.

VII. Results

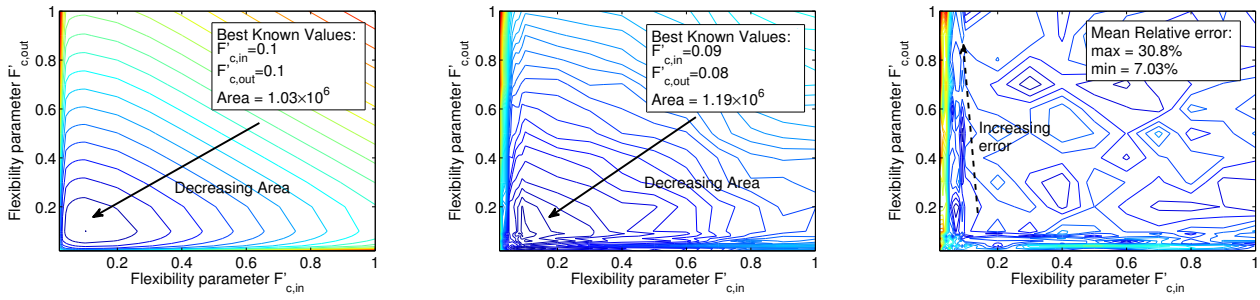
A. Routing Area Model Verification

Several parts of the model have been verified previously. The model for estimating the grid size was verified in [3], and the depth model in [7]. Analysis of the routing area model was performed in [5] with fixed buffer sizing. We extend the analysis here to observe the routing area when variable buffer sizes are used. We focus on the routing area rather than the logic area because the model used in the GP will give exact transistor counts for an individual logic cell in all cases. The only difference in the logic area model and any experiment will be due to the modelling of grid size, this effect is also present in the routing area model. The accuracy of the routing model is affected by the approximation for multiplexer area and the fact that a real multiplexer will have an integral number of inputs.

To evaluate the GP framework, we performed a comparison of our model to VPR 5.0 [1]. The latter tool constructs multiplexer models for specified architecture files, which include information such as the logic and routing parameters and the transistor sizes. VPR also chooses the minimum channel width that allows each design to be routable. In order to implement the GP framework, we employed `CVX`, a free plug-in toolset for solving convex programs [25], [26]. To verify our model is a good approximation relative to VPR, we used a parameter-sweep approach: whilst GP can be used to derive the optimal parameters of interest, variables can be fixed in the GP framework to evaluate the model as a closed form equation and determine the accuracy of the model across a sweep of parameters. In these experiments, we assume a fixed logic architecture: $K = 4$, $N = 10$ and $I = 22$. The cost function is set with $z = 0.5$, *i.e.* area-delay product is chosen for the optimisation objective.

Figures 5(a) and 5(b) show the results of our parameter sweep across the entire range of $F'_{c,in}$ and $F'_{c,out}$ for the model and experiment respectively. In the figures, each contour represents a difference of 30k transistors. To generate the data, 20 MCNC circuits have been employed, with an architecture generated for each and the plots show the geometric mean across all circuits. The results show that our model gives an accurate representation of the total FPGA area; the absolute values of area and trends are present in both the GP model and VPR.

Figure 5(c) shows the difference between the model and the VPR experimentation. The model is particularly inaccurate for extremely low values of $F'_{c,out}$. This is because the channel width model from [4] breaks down in this region: VPR must architect the routing multiplexers that connect to a very small number of tracks, for example one in a hundred, and this discretisation of $F'_{c,out}$ is difficult to make consistent across the array. Nevertheless, the mathematical model demonstrates that the trend tracking is sufficient to detect the correct region of the design space for determining the best routing architecture.



(a) (b) (c)

Fig. 5. A comparison of routing area model and experiment. (a) is the area model implemented as a GP, (b) is the VPR experimental method. Each contour represents a difference of 30k transistors. (c) shows the mean relative error graph, where each contour represents a 1% difference.

The GP framework has a significant advantage in terms of compute time required. Each point in the VPR experimentation can take between two minutes and four hours to obtain for a single benchmark dependent on benchmark size and interconnect complexity. Conversely, the GP framework takes approximately one minute to run, independent of the benchmark size, a speedup of up to 240×. Moreover, the optimal values for F_c in the model can be found in a single run of the GP rather than a parameter sweep, reducing the execution time of any sweeping approach by two orders of magnitude.

B. Delay Model Verification

In this paper we focus on verifying the circuit delay model. This is done through the use of HSPICE [27]. The SPICE model is constructed by specifying the transistors and connections in the critical path. The transistor sizings are specified by the results of the GP optimisation. The optimally derived high-level architecture parameters such as channel width and the flexibility parameters must be quantised for the SPICE model in order to obtain multiplexers with an integral number of inputs. Certain parts of the critical path are replicated, such as the switch box to switch box delay, and are only specified once. Each of the constituent parts of the SPICE model are extracted so that they can be scaled up by the factors in (33) and compared to the critical path reported by the GP model. The critical delay for each path is the maximum of the rise and fall times to half VDD extracted from the SPICE simulation. Figure 6(a) shows the geometric mean of the delay for a variety of LUT size and CLB size, as calculated by the GP optimisation framework. Figure 6(b) shows the same metric when delay components are extracted from SPICE. Each figure shows contours which indicate a separation of 50ps, the dotted contours close to the optimal point represent 10ps. The results show that both methods of evaluating delay give similar conclusions - that delay is optimised by selecting a large LUT size and that the delay is less sensitive to cluster size when the LUT size is greater. Whilst the trends are tracked within a reasonable degree of accuracy, the absolute values of the critical path model are approximately 20% lower than those of the SPICE model. This under-estimation is due to the large number of nMOS pass transistors in the circuit: the charge path is not to full VDD for nMOS transistors, and the Elmore model does not account for this.

Figure 6(c) shows the geometric mean of the area-delay product for a variety of LUT size and CLB size, as calculated by the GP optimisation framework. Similarly Figure 6(d) shows the area-delay metric when delay components are extracted from SPICE. The contours here represent a difference of 5×10^4 second transistor widths and the dashed contours represent 1×10^4 second transistor widths. In this case the same conclusions would be made about what constitutes the optimal architecture - in order to minimise the area-delay product, the number of LUT inputs should be 5 and there should be 4-5 LUTs in the CLB.

C. Cost Function Dependent Architectural Tradeoffs

The GP formulation allows a combination of area and delay to be targeted for determination of an optimal architecture. In order to observe the architectural tradeoffs that happen as the cost function is varied, we varied the exponent z in the cost function $T_{total}^{z} A_{total}^{(1-z)}$ and swept across the logic architecture parameters to determine the optimal architecture for each value of z. The architecture tradeoffs can be seen in Figure 7, in which the average number of transistors in each design is given with the average total transistor area. The graph shows that transistor area increases as delay takes more significance in the cost function. It also shows that little changes in terms of the architecture until $z > 0.6$; the same architecture parameters are chosen, and it is only the transistor sizes that change. However, beyond this point, modifications to the structure of the architecture happen. There are a number of observations that can made from examining the architectures. First, in order to improve the delay the amount of capacitance within the routing architecture is minimised by the GP by lowering the routing flexibility parameters, this is at the expense of increased channel width. In terms of the logic architecture, the capacity of a logic block is increased by employing LUTs with more inputs and packing more LUTs within a CLB. This effect requires a tradeoff: the amount of capacitance in the CLB feedback network increases delay within a logic block due to the full crossbar multiplexing (see Figure 1(a)), but the interconnect within a logic block may be faster than employing external connections, hence the number of LUTs within a CLB should be approximately 7-8.

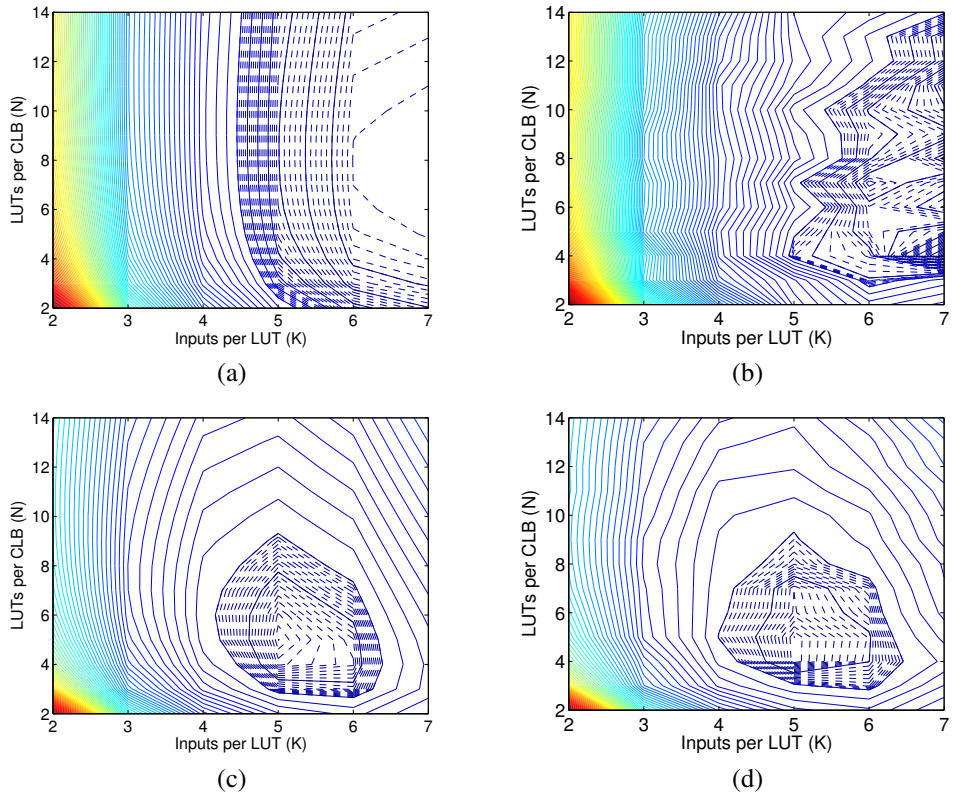


Fig. 6. A comparison of delay and area delay product of the combined area delay model for $z = 0.5$. Figure (a) is the modelled delay, (b) is the delay when delays are extracted from SPICE simulations, in both cases contours represent a difference in delay of 50ps and dotted contours around the optimal point represent 10ps. Figure (c) is the modelled area-delay product, (d) is the area delay product with delays extracted from SPICE simulations

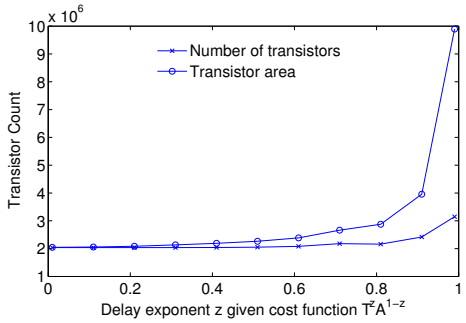


Fig. 7. A comparison of the number of transistors and the sum of their areas. The cost function is varied between delay and area oriented design.

D. Application of the Geometric Program

Our tool allows the high and low-level architecture parameters to be evaluated within the same framework. This has previously not been possible, and past studies have employed a two-stage approach in which architecture parameters are fixed before transistor sizing [13], [9] or vice versa. To demonstrate the impact of optimising the high-level and low-level parameters concurrently, we modeled three different flows using our framework.

The tool requires that the logic parameters N and K are fixed for each run of the optimisation tool. In order to find the optimal set of all parameters, it is necessary to sweep across a range of values of interest. Each run of our tool reports the transistor sizing, switch and connection box flexibility as well as the value of the objective function, the total area and the critical path delay. The best architecture is selected as the one which gives the best value of the objective function. This is the first experimental flow.

The second experiment also employs our framework, but models a two stage approach to optimisation. In the first stage, the GP optimisation procedure is performed, but all transistors are fixed to be minimum width - this is done through the introduction of equality constraints in the GP. A full sweep across all combinations of N and K is performed to find the optimal architecture parameters under the minimum width condition. The values of $F_{c,out}$ and $F_{c,in}$ are determined from the results of the GP optimisation. The objective function is reported for each N and K and the architecture parameters are selected for the device for the best value of the objective. After selecting the high-level parameters, the optimal parameters N, K, $F_{c,out}$ and $F_{c,in}$ chosen in the first stage are then fed into a new GP as constants, and the constraints on transistor widths are removed. The final delay, area and objective function are then reported and the transistor widths are determined.

The final experiment is a naïve approach in which each of K, N, $F_{c,out}$ and $F_{c,in}$ is chosen successively assuming fixed transistor sizes. We sweep across $K = 2-7$, determining the best LUT size for an arbitrarily chosen value of N, $F_{c,out}$ and $F_{c,in}$. K is then fixed and the CLB size N is chosen from $N = 2-12$. The routing flexibility parameters are chosen similarly

from a sweep of ten different values. Finally, the transistor sizing is determined given the best values found during the sweep procedure.

The exploration was performed on 20 MCNC circuits. Figure 8(a) shows the geometric mean area in minimum width transistors when varying the exponent z in the objective function $T_{total}^{z} A_{total}^{1-z}$, and Figure 8(b) shows the critical path delay of each architecture. The single stage approach improves the delay when the cost function is weighted towards delay as an objective, the difference between the dual stage approach in which the transistor sizing is considered after high-level parameter selection and the one stage approach is approximately 4% in terms of critical path delay. Whilst this difference is relatively small, an interesting observation is that the intuition supplied by the two methods is significantly different. Figure 8(c) shows how the optimal LUT size K and cluster size N vary as the cost function is varied. These results show that the single-stage and two-stage approaches head in different directions as delay takes more importance. The reason for this is that increasing the cluster size adds many capacitive loads on feedback paths (between blocks mapped into the same CLB). In the two-stage approach the drivers of these paths are not being sized simultaneously and therefore the cost of routing delay within a cluster appears worse than if you had the flexibility to simultaneously size the drivers.

In the case where the cost function is weighted towards area, the single stage optimisation and two stage optimisation give very similar results in terms of area. The difference in delay is similarly small. Unsurprisingly, the multi-stage heuristic performs worse for both metrics in this region - around 1% in area and 6% in delay compared to the best architecture.

These results are interesting from a design perspective: high-level architectural decisions can be made independently of transistor sizing with only a small effect on the metrics of interest. The reason for these two approaches being close together is that in the two-stage approach, every transistor is assumed to be minimum size prior to selecting the high-level parameters. This assumption turns out to be close to optimal because the second inverter in each buffer is the most critical to delay and is sized accordingly, however these are the only transistors differing from minimum width in the resulting optimized architectures.

The geometric programming approach has a significant advantage in terms of run-time over existing work. The geometric program takes approximately one minute to solve and includes optimisation of the routing flexibility parameters and the channel width. This is a significant improvement over previous work: up to 12 hours was reported in [13] for a single set of high-level architecture parameters. We note that [13] is likely to be more accurate due to the use of the SPICE simulator, the inclusion of wire delay and the feedback loop that involves full placement and routing of designs using VPR. However, our experimentation in Section VII-B indicated similar conclusions about the optimal selection of high-level architecture parameters would be gained were SPICE used for accurate delay measurement.

VIII. Conclusion

An accurate model that can be used to optimise any posynomial combination of area and delay has been presented for FPGAs. Whilst the absolute values reported by the framework are not exact, the framework allows us to conclude that the model is sufficiently good at tracking the trends present in the architecture space. The model could be further improved by deriving a more accurate version of the Elmore delay model. However, the same conclusions from the experiments should be reached.

The model is fast to evaluate and can be used to reduce the amount of time taken to explore FPGA architectures early in the design process. This is because only three parameters are used to specify each benchmark circuit, whereas many existing benchmarking techniques require synthesis, placement and routing of complex netlists. Furthermore, the small number of parameters used in our approach means that larger benchmark circuits do not affect the run-time performance.

One of the key results of this work has been to prove that significantly different architectural decisions may be made when architectural parameters are optimised concurrently. This provides strong support for the use of analytical modelling techniques in conjunction with formal optimisation. There are many further opportunities for modelling FPGA architectures, for example heterogeneous devices, complex routing structures and statistical delay models. All of these could be possible using geometric programming techniques. The MATLAB models and code employed in this work are publicly available at: http:/cas.ee.ic.ac.uk/people/as999/GP.

References

[1] 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 single-driver routing, heterogeneity and process scaling," in *International Symposium on Field-Programmable Gate Arrays*, Feb. 2009, pp. 133–142.

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

[3] A. Lam, S. J. E. Wilton, P. W. L. Leong, and W. Luk, "An analytical model describing the relationships between logic architecture and FPGA density," in *International Conference on Field-Programmable Logic and Applications*, Sep. 2008.

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

[5] A. M. Smith, G. A. Constantinides, and P. Y. K. Cheung, "Area estimation and optimisation of FPGA routing fabrics," in *International Conference on Field-Programmable Logic and Applications*, Sep. 2009.

[6] A. M. Smith, J. Das, and S. J. E. Wilton, "Wirelength modeling for homogeneous and heterogeneous FPGA architectural development," in *International Symposium on Field-Programmable Gate Arrays*, Feb. 2009, pp. 181–190.

[7] J. Das, S. J. E. Wilton, P. W. L. Leong, and W. Luk, "Modeling post-techmapping and post-clustering FPGA circuit depth," in *International Conference on Field-Programmable Logic and Applications*, Sep. 2009.

[8] A. M. Smith, G. A. Constantinides, S. J. E. Wilton, and P. Y. K. Cheung, "Concurrently optimizing FPGA architecture parameters and transistor sizing: implications for FPGA design," in *International Conference on Field-Programmable Technology*, December. 2009.

[9] H. L. Yu, E. Hung, T. Chau, and P. Leong, "A detailed delay path model for FPGAs," in *International Conference on Field-Programmable Technology*, Dec. 2009.

[10] E. Ahmed and J. Rose, "The effect of LUT and cluster size on deep-submicron FPGA performance and density," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 3, pp. 288–298, Mar. 2004.

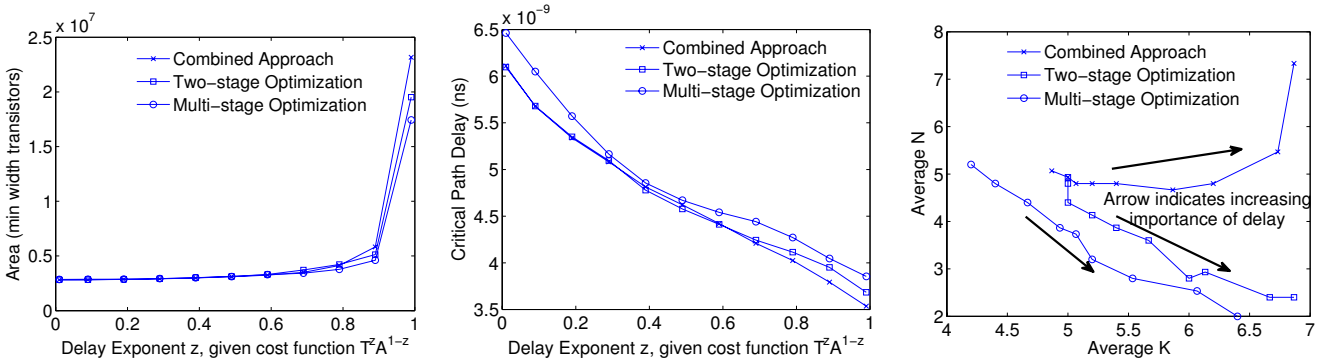


Fig. 8. Comparison of combined optimisation, dual-stage and multi-stage optimisation procedures: (a) gives the mean area of each approach, (b) shows the critical path delay and (c) shows what happens to N and K as the exponent is varied.

[11] D. Lewis *et al.*, "The stratixTM routing and logic architecture," in *International Symposium on Field-Programmable Gate Arrays*, Feb. 2003, pp. 12–20.

[12] V. Betz and J. Rose, "VPR: A new packing, placement and routing tool for FPGA research," in *Field-Programmable Logic and Applications*, W. Luk, P. Y. K. Cheung, and M. Glesner, Eds. Springer, Berlin, 1997, pp. 213–222.

[13] I. Kuon and J. Rose, "Area and delay trade-offs in the circuit and architecture design of fpgas," in *International Symposium on Field-Programmable Gate Arrays*, Feb. 2008, pp. 149–158.

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

[15] B. Landman and R. Russo, "On a pin versus block relationship for partitions of logic graphs," *IEEE Transactions on Computers*, vol. C-20, no. 12, pp. 1469–1479, Dec. 1971.

[16] 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.

[17] S. Sapatnekar, V. Rao, P. Vaidya, and S.-M. Kang, "An exact solution to the transistor sizing problem for CMOS circuits using convex optimization," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 12, no. 11, pp. 1621–1634, Nov 1993.

[18] S. Sapatnekar, "Wire sizing as a convex optimization problem: exploring the area-delay tradeoff," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 8, pp. 1001–1011, Aug 1996.

[19] 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, Dec 2007.

[20] G. Rose, E. Lee, M. Tom, and A. Yu, "Directional and single-driver wires in FPGA interconnect," in *International Conference on Field-Programmable Technology*, Dec. 2004.

[21] M. Feuer, "Connectivity of random logic," *IEEE Trans. on Computers*, vol. CAS-26, no. 4, pp. 29–33, Jan. 1982.

[22] W. Zhao and Y. Cao., "New generation of predictive technology model for sub-45 nm early design exploration," *IEEE Transactions on Electron Devices*, vol. 53.

[23] T. H. Morshed *et al.*, "Bsim4.6.4 MOSFET model - user's manual," Feb. 2009, http://www-device.eecs.berkeley.edu/~bsim3/BSIM4/BSIM464/BSIM464_Manual.pdf.

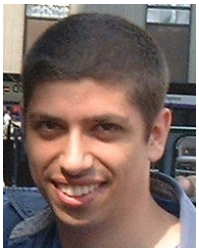
[24] W. C. Elmore, "The transient analysis of damped linear networks with particular regard to wideband amplifiers," *Journal of Applied Physics*, vol. 19, no. 1, pp. 55–63, Jan. 1948.

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

[26] ——, "Graph implementations for nonsmooth convex programs," *Recent Advances in Learning and Control (a tribute to M. Vidyasagar), Lecture Notes in Control and Information Sciences*, pp. 95–110, 2008.

[27] Synopsis Inc., San Jose, "HSPICE. http://www.synopsys.com/products/mixedsignal/hspice/hspice.html."

Alastair M. Smith received an M.Eng. (Hons) degree in Electrical and Electronic Engineering in 2003 from Imperial College (U.K). Thereafter, he joined the Circuits and Systems research group in the same department, receiving his Ph.D. in 2007. Dr. Smith worked as a post-doctoral fellow at The University of British Columbia, Vancouver, Canada from 2007-08, and is currently a Research Associate in the Circuits and Systems research group at Imperial College. His research interests include Computer Architecture, Reconfigurable Computing, convex optimisation and Integer Linear Programming. He currently serves on the programme committees of FPL, ReConFig, and ARC.



George A. Constantinides (S96-M01-SM08) is Reader in Digital Systems and Head of the Circuits and Systems research group in the department of Electrical and Electronic Engineering at Imperial College London. He is a recipient of the Eryl Cadwaladar Davies Prize for the best doctoral thesis in Electrical Engineering at Imperial College (2001), an Imperial College Research Excellence Award (2006), and a best paper prize at ARC 2010. He is Associate Editor of the IEEE Transactions on Computers and the Journal of VLSI Signal Processing. He was Programme Co-Chair of the IEEE International Conference on Field-Programmable Technology (FPT) in 2006 and Field Programmable Logic (FPL) in 2003. He currently serves on the programme committees of several international conferences, including FPL, DATE, DAC, and FPT. He has published over 90 research papers in peer refereed journals and international conferences. Dr Constantinides is a member of the IEEE, the ACM, and SIAM.

Peter Y. K. Cheung (M85SM04) received the B.S. degree with first class honors from Imperial College of Science and Technology, University of London, London, U.K., in 1973. Since 1980, he has been with the Department of Electrical Electronic Engineering, Imperial College London, where he is currently a Professor of digital systems and head of the department. He runs an active research group in reconfigurable circuits and systems. His research interests include VLSI architectures for signal processing, asynchronous systems, reconfigurable computing using FPGAs, and architectural synthesis.