Evaluating Area and Performance of Hybrid FPGAs with Nanoscale Clusters and CMOS Routing

REZA M.P. RAD
University of Maryland
and
MOHAMMAD TEHRANIPOOR
University of Connecticut

Advances in fabrication technology of nanoscale devices such as nanowires, carbon nanotubes and molecular switches provide new opportunities for implementing cluster-based FPGAs. Extensive research is needed to evaluate area and performance of FPGAs made from these devices and compare with their CMOS counterparts. In this work, we propose a hybrid FPGA that uses nanoscale clusters with a functionality similar to the clusters of traditional CMOS FPGAs. The proposed cluster is constructed by a crossbar of nanowires and can be configured to implement the required LUTs and intracluster MUXes. A CMOS interface is also proposed to provide configuration and memory elements for the nanoscale cluster. In the proposed architecture, inter-cluster routing remains at CMOS scale. We have developed models for area and delay of clusters and interconnects of the proposed hybrid FPGA. FPGA tools are configured with these models and used to synthesize and configure the benchmark circuits onto the hybrid FPGAs with NiSi nanowires or nanotubes. Experiments are conducted to evaluate and compare area and performance of the hybrid FPGA and traditional CMOS FPGA (scaled to 22 nm). Up to 82% area reduction was obtained from implementing MCNC benchmarks on the hybrid FPGA. Performance of the hybrid FPGA is shown to be close to that of CMOS FPGA.

Categories and Subject Descriptors: B.7.2 [**Integrated Circuitst**]: Types and Design Styles

General Terms: Design

Additional Key Words and Phrases: Nanotechnology, FPGA, CMOS, reliability, performance

ACM Reference Format:

Rad, R. M. P. and Tehranipoor, M. 2007. Evaluating area and performance of hybrid FPGAs with nanoscale clusters and CMOS routing. ACM J. Emerg. Technol. Comput. Syst. 3, 3,

This article is an extended and revised version of the paper presented at the 2006 IEEE/ACM Design Automation Conference (DAC) © ACM 2006.

Authors' addresses: R. M. P. Rad, Department of CSEE, University of Maryland Baltimore County; email: reza2@umbc.edu, M. Tehranipoor, Department of CSEE, University of Connecticut; email: tehrani@engr.uconn.edu.

Article 15 (November 2007), 00 pages. DOI = 10.1145/1295231.1295235 http://doi.acm.org/10.1145/1295231.1295235

1. INTRODUCTION

Research in all areas of emerging nano-electronic devices has had an exciting rate of growth that raises the hopes for expanding semiconductor era beyond CMOS in few years. Researchers have accomplished the basic fundamental steps that are required for this to happen. However, there are still many milestones that must be passed. In the following we briefly describe the recent advances in device technology, assembly techniques, interface systems between nano and CMOS devices and architectures proposed based on emerging nano-devices.

Research on molecular electronic components is an increasingly growing field and molecular scale parts are expected to be used as circuit components in the near future [Semiconductor Industry Association 2005]. Molecules with configurable switching and rectifying properties are reported in Chen et al. [2003] that can be used to develop diode-logic circuits. These molecular switches are programmable and can retain their connected or disconnected state, that is, it is possible to create re-configurable nanodevices using such switches. Carbon nanotubes (NTs) with less than one nanometer diameter and micrometers length have been synthesized [Dekker 1999]. Also nanowires (NWs) with diameters as small as 3 nanometers and lengths of few hundred micrometers were reported [Cui et al. 2001; Morales and Lieber 1998]. Techniques for applying different doping on these nanowires have also been developed and field-effect transistor (FET) operation was observed [Huang et al. 2001; Cui et al. 2003]. These synthesized nanowires or nanotubes are the basic elements used in implementation of nanoscale crossbars. Electrical properties of these nanoscale wires are under extensive theoretical and experimental investigation. Various research groups have reported the parasitic values of the synthesized nanowires or nanotubes. In this work we have obtained basic parasitics (resistance and capacitance) of the nanowires according to theoretical estimations and assumptions made in DeHon [2005] and also results of experimental measurements reported in Yu and Burke [2004]. These values are used in evaluations we have performed for delay/performance of the proposed architecture.

Assembly of nanoscale components will be one of the major challenges in the nanoscale circuit design. Experiments to use self-assembly and self-alignment techniques for arranging nanowires in array structures are reported in Huang et al. [2001] and Whang et al. [2003]. The main limitation however is that these techniques can only provide array-like regular structures and are not as flexible as today's advanced lithography. *Nano-Imprint* [Chen et al. 2003; Chou et al. 1996; McAlpine et al. 2003] technology is another approach proposed for making nanoscale structures. It has been suggested that this technology can provide more flexibility in designing nonregular structures in nanoscale regime. A hybrid technology which uses both self-assembly and nano-imprint may become the dominant technology at this scale.

Nanoscale circuits implemented by emerging molecular devices will need a CMOS-scale support to provide inputs, outputs and configuration circuitry for the molecular-scale parts. An interface based on modulated doping of nanowires and their use as field effect transistors to create decoder and demultiplexer (DMUX) interfaces was presented in DeHon [2005] A DMUX structure based on random deposition of gold particles on a nanowire array structure was proposed in Kuekes and Williams [2000]. In the proposed architecture in this paper we have assumed the use of self-assembly techniques for implementation of the crossbars. We have also assumed that contacts between nanowires and CMOS metals can be provided through one of the various techniques named previously, for example, nano-imprint, to create the required CMOS-Nano interface. Detailed discussions on these topics are given in Section 2.

Various architectures for nanoscale circuits have been proposed in literature and different research groups have reported their evaluations on the area and performance of the proposed architectures. An array architecture for nanoscale devices was suggested in Goldstein and Budiu [2001]. It is an island style architecture in which clusters of nanoblocks and switchblocks are connected in an array structure. The architecture assumes rich interconnect between clusters that improves accessibility and routability of the fabric and hence its performance. Each nanoblock is a grid of nanowires with molecular switches at the junction of each pair of nanowires that can be configured either as a diode or as a disconnection. A PLA-based FPGA-like architecture is presented in DeHon [2005] that uses modulated-doping nanowires for CMOS-Nano interface. Different aspects of this architecture including its area and power are discussed and theoretically analyzed.

Snider et al. [2004] proposed a CMOS-like logic based on nanoscale FETs created in crossbar architectures. This logic style is proposed to be usable in array structures to create reconfigurable architectures. However, creating each gate requires routing signals between N-plane and P-plane. This complicates the circuit synthesis process and routing area may become a bottleneck. A cell-based architecture and interface scheme using special metal pins implemented on the surface of the substrate to provide the contacts with nanowires is proposed in Strukov and Likarev [2005]. The above architectures need completely new set of synthesis, placement and routing tools for performing more accurate evaluations on their area and performance efficiency based on implementation of benchmark circuits on them. Gayasen et al. [2005] proposed an architecture where the routing interconnects of FPGAs were implemented by nanowires and logic blocks of the FPGAs were kept the same as traditional CMOS FPGAs. The authors used available FPGA synthesis tools in their experiments and it was shown that nanowire-based routing can reduce area of the FPGA up to 70%. Although the area of the proposed architecture is significantly lower than CMOS FPGAs, but using long nanowires can also significantly lower the reliability of such FPGAs.

1.1 Contribution and Article Organization

More accurate and extensive evaluation of architectures designed from nanoscale devices is only possible through the use of experimental approaches, implementing benchmarks on these architectures, and analyzing their advantages and shortcomings. However, performing experiments on the architectures discussed will require a new set of computer-aided design (CAD) tools.

In this article, we propose a new nanowire-based structure for the FPGA clusters. The proposed nanoscale cluster has the advantage of being functionally compatible with clusters of traditional CMOS FPGAs. This allows the use of available and well-developed FPGA tools, for example, partitioning, packing, placement and routing tools, for performing architectural evaluation and analysis of the proposed architecture. Another advantage of the proposed cluster is that the fabrication techniques for nanowire arrays and nanowire-metal contacts are being well developed. The advances made in the assembly and contact techniques are the basis for implementing the proposed cluster. The proposed cluster is composed of nanoscale crossbars working as LUTs, CMOS latching and inversion support for the nano part and the CMOS-Nano interface and configuration logic. The routing structure of the FPGAs is assumed to remain at CMOS scale.

Section 2 reviews the traditional CMOS-scale logic cluster architecture and presents the proposed nanowire-based cluster and its CMOS support circuitry. The experimental setups are discussed in Section 3. The experiments were performed in order to investigate the effects of the proposed nanoscale cluster on area and delay of the MCNC benchmark circuits. Additionally, an analysis was done in order to determine how the values of N and K must be tuned to obtain area and performance-efficient nanoscale clusters. Results of our experiments demonstrate that correct selection of these parameters makes the proposed hybrid FPGA capable of outperforming traditional CMOS FPGAs. In our experiments we used area and delay models of different parts of the architecture based on calculations presented in Section 4. We have implemented MCNC benchmarks on both the new hybrid FPGA and CMOS FPGA. The experimental results are reported and discussed in Section 5. The concluding remarks are in Section 6.

2. LOGIC CLUSTER ARCHITECTURE

2.1 Logic Clusters in CMOS FPGAs

Figure 1 shows the structure of a *Basic Logic Element (BLE)* and cluster architecture traditionally used in FPGAs [Ahmed and Rose 2004]. As shown, each cluster contains N BLEs each of which consists of one K-input LUT and one flip-flop (*DFF*). The cluster contains I inputs, and it has been demonstrated that $I < N \times K$ will result in optimum area and performance for FPGAs [Ahmed and Rose 2004]. In CMOS, LUTs and MUXes used to provide interconnections between BLEs (intra-cluster routing) can take up to 70% of the cluster area [Gayasen et al. 2005]. Therefore, new implementation methods that can reduce the size of these two parts, that is, LUTs and intracluster MUXes, will significantly reduce the area of clusters in FPGAs. Hence, implementation of the clusters by employing nanowire based crossbars as LUTs and MUXes could significantly reduce the cluster area.

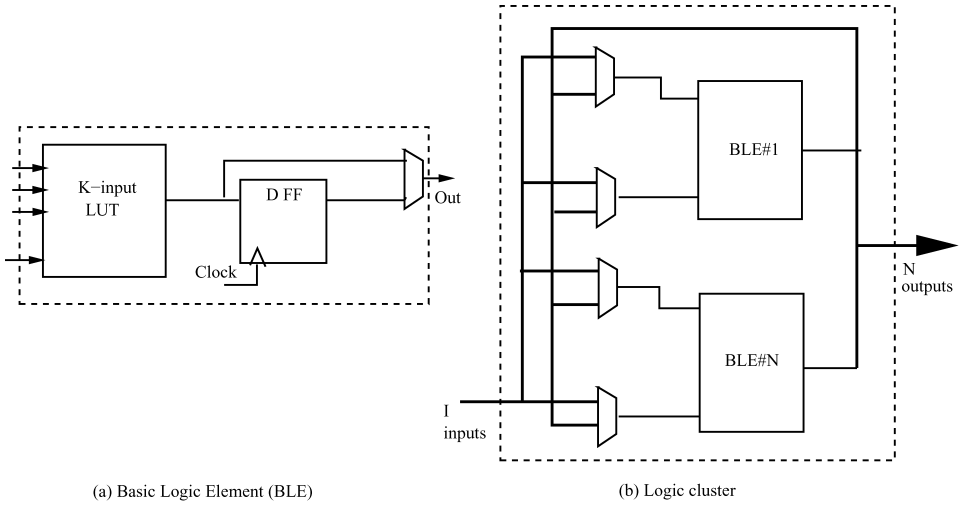


Fig. 1. Architecture of (a) basic logic element and (b) FPGA cluster.

The area of FPGA is determined by area of its clusters and routing between the clusters. Previous evaluations and experiments on FPGAs have shown that intercluster routing has the major contribution in area of FPGAs. However, as will be further discussed in Section 5, having clusters with higher logic capacity will reduce the required intercluster routing area and in turn the overall area of the FPGAs.

It is worth mentioning that state of the art FPGAs may contain other blocks like RAM modules, hard coded parts (e.g., Multipliers) and embedded processors. However, in this work we assume that FPGAs are composed solely of logic clusters as shown in Figure 1.

2.2 Nanoscale Cluster

As mentioned earlier, LUTs and intracluster MUXes consume most of the CMOS clusters' area in FPGAs. The main motivation here is to suggest new structures for LUTs and MUXes based on crossbars of nanowires in an architecture that is functionally compatible with the cluster shown in Figure 1. This provides the opportunity to use existing FPGA synthesis tools in our experiments.

—*Nanoscale LUTs*. Crossbars are widely proposed by researchers to be used as the main component of nanoscale devices. A crossbar is generally made of nanowires or nanotubes arranged in the form of an grid. A molecular layer can be placed between horizontal and vertical wires of this grid. Portions of this molecular layer that are on the crosspoint of horizontal and vertical wires can be programmed as ON or OFF switches through applying appropriate programming voltages to the associated nanowires. The molecules will retain their state until another programming voltage is applied to the nanowires. Therefore, we can have programmable crosspoints (i.e., switches) that can store their connection or disconnection status without requiring a

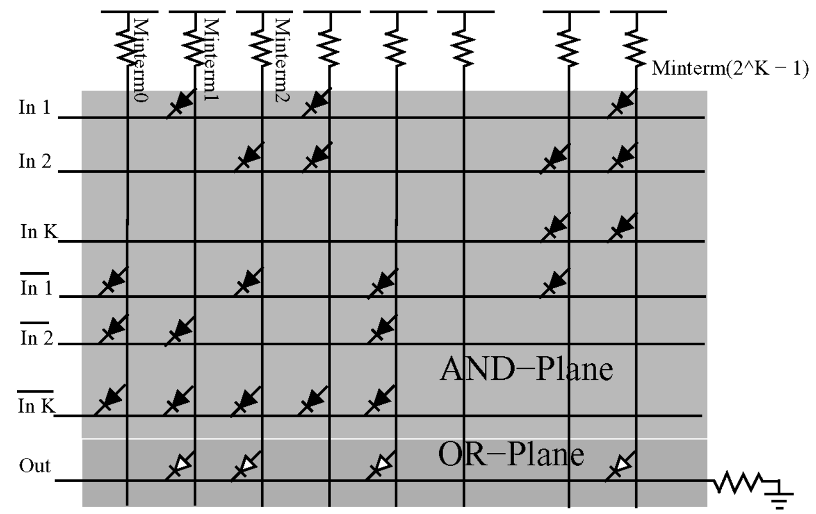


Fig. 2. Function $f = \sum Minterms(1, 2, 4, 2^K - 1)$ implemented on a K-input crossbar LUT.

SRAM cell or other memory elements as in CMOS programmable devices. Such a crossbar can be implemented on top of a substrate containing CMOS circuitry. The CMOS circuitry can apply the required programming voltages to the nanowires. Placing CMOS circuits underneath the nano-crossbars will also result in more area savings. The crossbars can be programmed as PLAs [DeHon 2005], MUXes [Kuekes and Williams 2000], or LUTs. To implement LUTs on crossbars, part of the crossbar is configured as AND-plane and the other part as OR-plane. These parts together provide sum of the minterms for the functions that must be implemented on the crossbar. Figure 2 shows a K-input crossbar-based LUT. Columns are connected to *Vdd* through pull up resistors and make AND-plane of the LUT. The diodes on each column are configured to create one of the minterms, and overall 2^K columns are required for a LUT.

For each column, the binary code of the minterm determines which crosspoints must be configured as diodes. For implementing a minterm on a column, the crosspoints at the inputs (rows) with value “1” in that minterm are programmed as diodes. Since a diode logic is implemented on the crossbar, complements of inputs are also required to be applied to the LUT. Rows of the crossbar are used as inputs and outputs of LUT. Output rows must be connected to *Gnd* through a pull down resistor. These rows will provide sum of the product terms of the functions that are implemented on the LUT. As an example, Figure 2 shows function $f = \sum Minterms(1, 2, 4, 2^K - 1)$.

—*Cluster Architecture*. Figure 3 shows the architecture of the proposed cluster [Rad and Tehranipoor 2006a]. As seen in the figure, different regions of the crossbar are configured to implement several LUTs and intracluster MUXes. The right and top blocks are address decoders and configuration circuits that provide the required access to horizontal and vertical nanowires so that

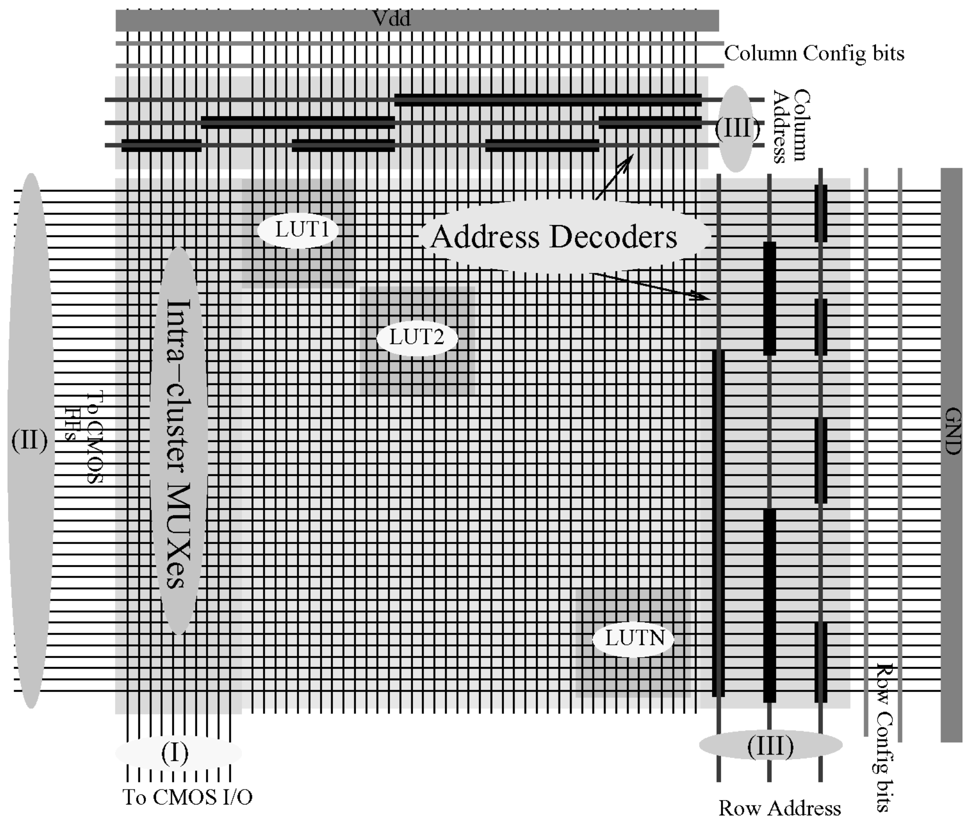


Fig. 3. Architectur of the nanoscale cluster containing address decoders and configuration circuits (top and right components), intra-cluster MUXes (left component) and the LUTs (central component).

programming voltages can be applied to them. The two address decoders and configuration logic provide programmability for each single molecular switch located on crosspoints. Accessing to individual nanowires of a crossbar and providing configurations voltages to them is one of the main challenges that must be addressed in order to have applicable crossbar-based circuits. Several strategies for address decoding and configuring crossbars are proposed in DeHon [2005]; Rad and Tehranipoor [2006b]; Kuekes and Williams [2000]. Any of these interface and decoding methods can be used to provide access and program crosspoints of the nanowires in the architecture proposed in Figure 3. However, decoding methods proposed in DeHon [2005] and Kuekes and Williams [2000] will provide an stochastic access mechanism to the nanowires of the crossbar. Therefore, using these decoding architectures will require pre-processing steps to determine the exact nanowire that can be accessed through each address applied to the decoder.

The technique proposed in Rad and Tehranipoor [2006b] aims to provide a nonstochastic access mechanism that eliminates the need for the pre-processing steps. Application of this technique to dense nanowire crossbars

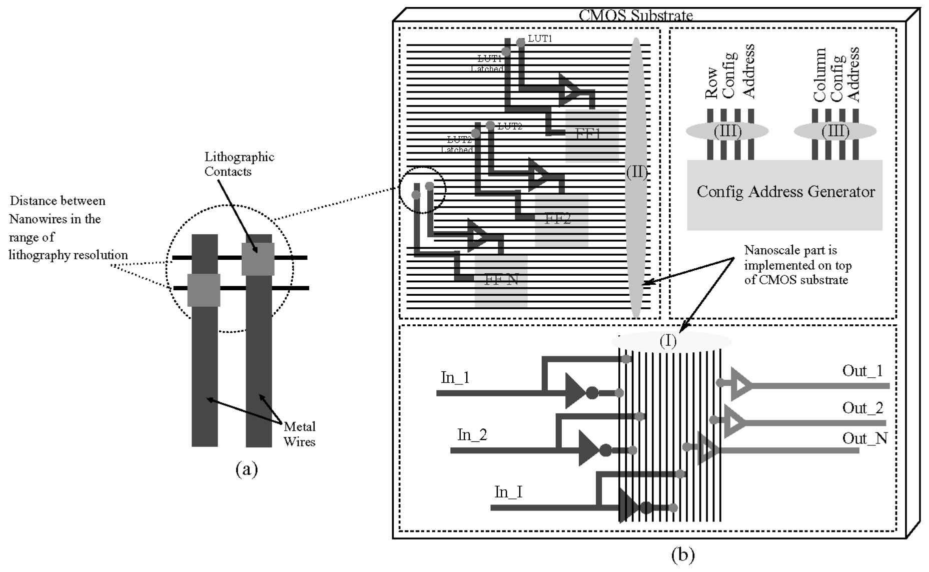


Fig. 4. (a) CMOS support circuitry of the cluster. Nanowires labeled (I) provide I/O to nanoscale cluster in Figure 3. Nanowires labeled (II) come from inputs and outputs of the MUX section of cluster shown in Figure 3, the output of LUTs are connected to the flip-flops and returned back to another nanowire that goes back to the MUX section of Figure 3. CMOS wires labeled (III) will provide configuration signals for rows and columns of the nanoscale cluster. (b) Lithographic contacts can be created between metal wires and nanowires in a sparse array.

relies on advances in future lithography technology to provide accurate placement of the contacts between CMOS wires and nanowires. However, in case of sparser arrays of nanowires lithography methods can be used to create the required contacts (within the precision limits of the employed lithography technique). Therefore, providing nonstochastic access to sparse nanowire arrays is achievable. This is further illustrated in Figure 4(b).

The components on the left side in Figure 3 work as intracluster MUXes. Since the crosspoints can be programmed they can provide a diode connection between the horizontal and vertical wires. Therefore, cluster inputs and outputs of the LUTs can be routed through the MUXes. The central part in the figure is where the LUTs are constructed. This can be done by programming proper diodes on crosspoints to create minterms of inputs as we described in the previous subsection. Since the fabrication technology of nanowires can provide only simple arrays, LUTs must be created in a diagonal style in the central block as shown in the figure. This is because nanowires that compose the columns and rows should directly be connected to the configuration circuits. Inputs and outputs of the cluster are connected to the CMOS support circuitry.

—*CMOS Support*. Figure 4(a) shows the architecture of CMOS support circuitry that is used to provide inversion (bottom), latching (top left) and configuration addresses (top right) for the nanoscale portion of the cluster. This CMOS part can be implemented on the substrate under the nanoscale crossbar to minimize the area. As seen, I inputs, their complements and N outputs of the cluster are connected to the nanowires of the intra-cluster MUXes of Figure 3 (components labeled (I) in Figures 3 and 4). Also the outputs of LUTs of Figure 3 are connected to the flip-flops provided in the CMOS support part (components labeled (II) in the figures).

Although creating very dense crossbars of nanowires based on low cost techniques like nano-imprint [McAlpine et al. 2003] or self-assembly has been achieved, providing effective interface for these crossbars that connect the crossbars to configuration circuitry and other CMOS parts relies on the future advances and new approaches to lithography like Dip Pen Nanolithography [Ginger et al. 2003]. The advances made in these techniques raise the hope for having effective contacts between CMOS and dense nano crossbars. Since the density of nanowires in the crossbars is controllable, sparse crossbars can be created where the distance between each two nanowires lies in the range of lithography precision. In this case providing contacts between CMOS metals and nanowires through lithography can provide the required connections in the CMOS-Nano interface. Figure 4(b) shows the possibility of having lithographic scale contacts between metal wires and nanowires in a sparse array. In this figure we have assumed that the distance between nanowires in the crossbar in the same range as the resolution of the lithography technique.

3. EXPERIMENT SETUPS

In order to investigate the area and delay properties of circuits implemented on the proposed architecture and also to determine the effect of cluster parameters (cluster size (N) and LUT size (K)) on area and performance in the new FPGA, we have implemented MCNC benchmark circuits on the proposed architecture. Since the proposed cluster is functionally equivalent to the traditional CMOS FPGAs and routing structure of the new FPGA is assumed to be the same as CMOS FPGAs, available FPGA tools can be used in these experiments. We have used the CAD tools discussed in Betz et al. [1999] in our experiments. The area and performance estimation procedure using such tools is shown in Figure 5. As seen in the figure, benchmarks are first mapped to K-input LUTs using SIS, then T-vpack is used to pack the LUTs into clusters of size N. Finally, VPR performs placement and routing of the clusters and calculates the area and delay of the FPGA based on the architecture model. The architecture model must be defined for VPR by specifying a number of parameters including the number of I/O signals considered for each cluster, pin configuration for the clusters, the type of switches and buffers that must be considered for switch-boxes between the interconnects, and delays between different parts of clusters and interconnect channels. The procedure also requires the values of cluster size (N), LUT size (K) and the number of cluster inputs (I) for implementing the benchmarks. Experiments are performed for various values of these parameters. The number of inputs to the clusters is set to $I = K \times (N + 1)/2$ in the experiments.

For the sake of comparison, we have also performed the same set of experiments on traditional CMOS FPGAs scaled to $22nm$ technology node. ITRS selected this technology node as probably the last technology node for CMOS

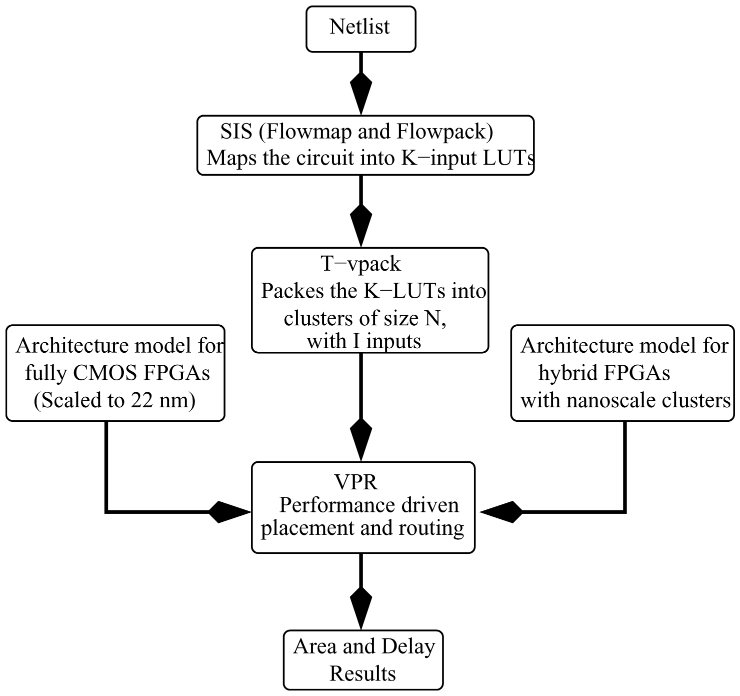

Netlist
SIS (Flowmap and Flowpack) Maps the circuit into K−input LUTs
T−vpack Packes the K−LUTs into clusters of size N, with I inputs
Architecture model for fully CMOS FPGAs (Scaled to 22 nm)
Architecture model for hybrid FPGAs with nanoscale clusters
VPR Performance driven placement and routing
Area and Delay Results

Fig. 5. Area and performance estimation procedure.

technology. Considering the fact that feature sizes in the 22*nm* node are close to the size of nanowires (few nanometers), a fair comparison between area and delay of CMOS FPGAs and hybrid FPGAs would be possible.

To perform the evaluations, VPR needs the value of resistances and capacitances of different parts of the cluster and switches used in the routing channels in addition to the line segments of the routing channels. We have obtained the estimated value of these parameters for 22*nm* CMOS from ITRS [Semiconductor Industry Association 2005]. The value of these parameters for the hybrid FPGA is calculated in two different ways and the experiments are repeated for each case. In the first case we have used analyses and assumptions discussed in DeHon [2005] to evaluate the parasitic parameters. In the second approach the experimental value of resistance of SWCNTs reported in Li et al. [2004] are used to obtain the parameters. Details of these evaluations are presented in the next section.

4. AREA AND DELAY MODEL

In order to perform placement and routing and to obtain area and delay information for the benchmarks, VPR requires an architecture description file. This file contains area and delay parameters of different parts of clusters and routing architecture used in the FPGA. These parameters are calculated for the scaled 22*nm* CMOS FPGAs and device level parasitic information. The area parameters of hybrid FPGA are calculated based on the geometry of cluster

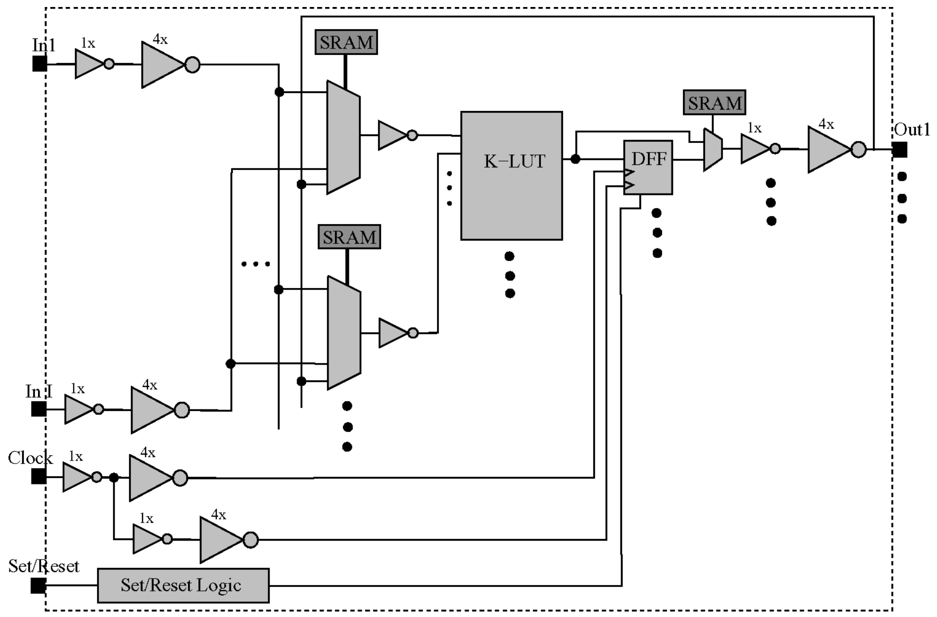


Fig. 6. Components of a CMOS logic cluster.

components. We have chosen two different approaches for evaluation of the delay parameters in the hybrid FPGA. In the first set of evaluations, resistance and capacitance of nanowires are calculated based on basic theoretical estimations. In the second set of estimations we have used experimental values of resistance and capacitance measured for single wall carbon nanotubes to obtain the delay of different components in the hybrid FPGA architecture. An overview of all calculations is given in the following.

4.1 CMOS FPGA Parameters

—*Area.* The area of a logic cluster is calculated based on the number of minimum width transistors. Figure 6 shows a more detailed view of the logic cluster that we use to determine the number of transistors (N_{tr}) required in one cluster. To calculate the number of minimum width transistors in a LUT, we use the architecture shown in Figure 7.

Figure 8 lists all the components shown in Figure 6 in addition to the expressions for number of transistors assuming a cluster with I inputs, N BLEs and K-input LUTs. The cluster size is sum of all transistors in the last column.

The area of a FPGA is estimated based on the total number of minimum width transistors used in the routing switches of inter-cluster interconnects clusters. The number of routing channel transistors depends on width of routing channels used in the FPGA and type of switches used.

Figure 9 shows a FPGA tile that includes a logic cluster and its neighboring interconnects and the switches used to connect routing tracks to each other and to cluster inputs and outputs. The type of switches used in the interconnects (tristate or pass-transistor) is a parameter that can be specified in the architecture

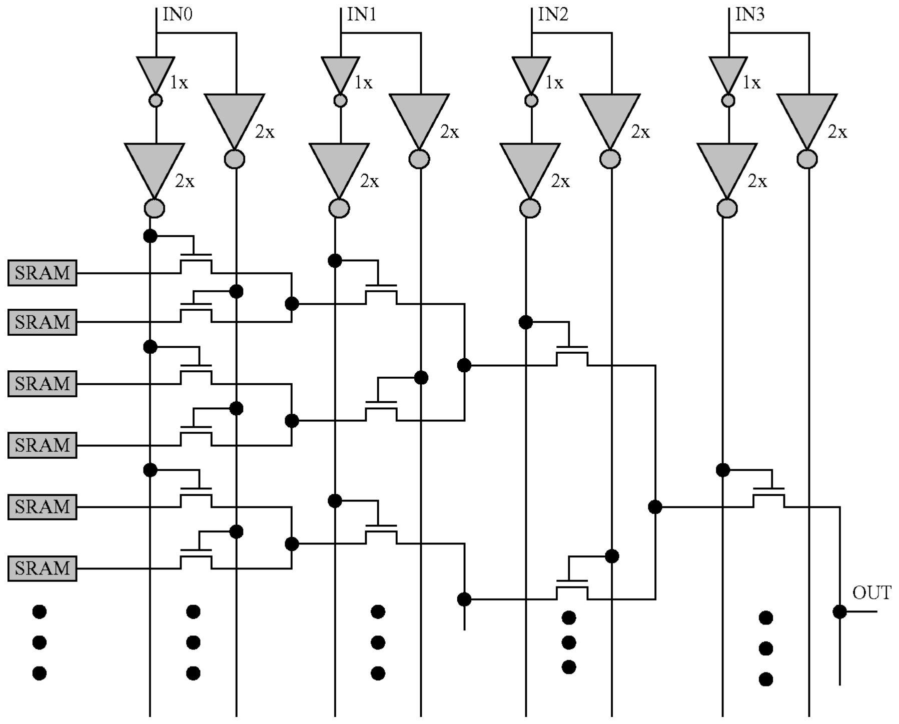


Fig. 7. A CMOS LUT formed by an array of MUXes and a set of SRAMs.

model applied to VPR (see Figure 5). Channel width is another parameter that should be determined according to routing requirements. Once a benchmark is placed and routed, the VPR tool calculates the minimum channel width (W_{min}) required to route the benchmark on the FPGA architecture. Having W_{min} and the type of switches, VPR calculates and reports the number of minimum width transistors required in the routing channels in each tile. Therefore, the number of minimum width transistors used in a tile of the FPGA ($N_{tr}(per\ tile)$) can be calculated by:

$$N_{tr}(per\ tile) = N_{tr}(in\ \ one\ \ cluster) + N_{tr}(inter\text{–}cluster\ \ routing\ \ per\ tile)$$

The total number of transistors in the FPGA is equal to $\sum_{i=1}^{N_{tile}} N_{tr}(cluster\ \ i)$, where N_{tile} is the total number of tiles used by target application.

Since the clusters in hybrid FPGA are made from nanowires, it is not practically possible to calculate the area of these FPGAs based on number of minimum width transistors. Therefore, in order to make fair comparisons between area of the hybrid and CMOS FPGA, we need to obtain the overall area of CMOS FPGA in μm^2. This can be calculated using the number of minimum width transistors, area of a minimum width transistor and the maximum achievable transistor density:

$$Area\ (in\ \mu m^2) = \sum_{i=1}^{N_{tile}} N_{tr}(cluster\ \ i) \times \frac{Area\ of\ a\ Min\ width\ transistor}{Max\ transistor\ density}.$$

Component	Volume	Parts included	# of transistors	Total # of transistors
LUT	N	SRAM	$6 \mathrm{x} 2^{k}$	$(6\mathrm{x}2^{k} + \sum_{i=1}^{k} 2^{i} + 10k)\mathrm{x}N$
		Mux Tree	$\sum_{i=1}^{k} 2^{i}$	
		Input Buffers	10k	
Local Routing MUXes	NxK	MUXes	$\sum_{i=1}^{\log(N+I)+1} 2^{i}$	$N\mathrm{x}K \mathrm{~x} \{ \sum_{i=1}^{\log(N+I)+1} 2^{i} + 6(\log(N+I)+1) \}$
		SRAM	$6(\log(N+I)+1)$	
2:1 MUXes	N	—	8	8xN
DFF	N	—	19	19xN
Set/Reset	1	—	48.5	48.5
Clock Buffers	1	—	18.5	18.5
Buffers on local MUXes	NxK	—	2.35	2.35xNxK

Fig. 8. Various components of a CMOS logic cluster and the number of transistors used in each component.

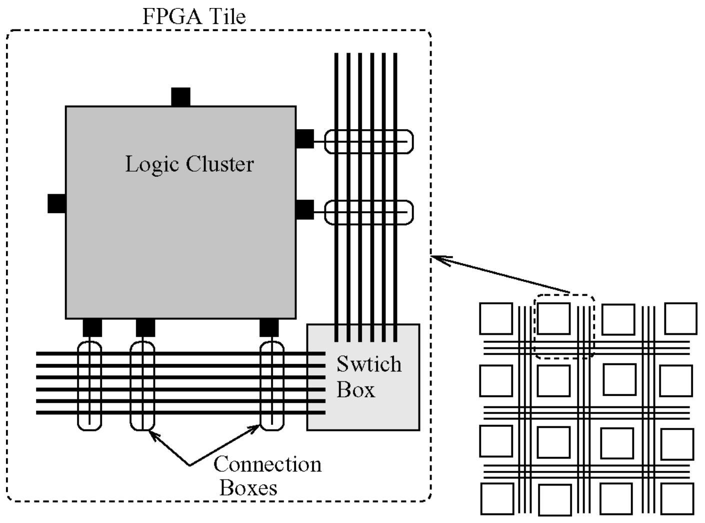


Fig. 9. Architecture of a FPGA tile including logic cluster and switches, MUXes and SRAMs that connect the I/O of the cluster to interconnect tracks.

Maximum transistor density is estimated to be 60% as in Betz et al. [1999]. As an example, the area of a minimum width transistor for TSMC 0.35 μm technology is 6.2275 μm^2. Area of a single transistor can be calculated in a new technology node through scaling the area of the transistor in a known

Parameter	ITRS Predicted Value for 22nm technology
Parasitic Source/Drain series resistance ($\Omega - \mu m$)	79
Ideal NMOS device gate capacitance (F/μm)	3.45E-16
NMOS intrinsic delay (ps)	0.15
Nominal logic gate delay (ps)	3.74
Conductor effective resistivity ($\mu\Omega - Cm$)	6.01
Interconnect RC delay for 1 *mm* Cu metal (ps)	8040

Fig. 10. Delay and parasitic parameters predicted by ITRS for 22*nm* CMOS technology node.

technology:

$$Area\ of\ Tr\ in\ new\ tech = Area\ of\ Tr\ in\ known\ tech \times \left(\frac{Pitch\ of\ new\ tech}{Pitch\ of\ the\ known\ tech}\right)^2.$$

Note that Tr stands for transistor in the above equations. We use this expression to compute the area of a transistor in 22*nm* technology and compute the total area assuming that maximum transistor density remains at 60%.

—*Delay*. Delays of interconnect line segments are calculated based on Elmore model through estimating load capacitances and resistances along a routing metal segment connected to several logic clusters. The value of input and output capacitance of NMOS and PMOS transistors were obtained from foundry data [Betz et al. 1999]. Resistance and capacitance of the metal line segment was calculated based on an estimation of the line segment length and the foundry data for unit length resistance and capacitance of the metal layer used in the interconnects.

While lack of foundry data for 22*nm* technology prohibits accurate SPICE simulation of the logic sections and obtaining precise resistance and capacitance values for NMOS and PMOS transistors and metals, we have chosen to rely on the predictions provided in ITRS [Semiconductor Industry Association 2005] to perform the delay measurements. Figure 10 represents a short list of the ITRS predictions for the parameters used in our delay calculations.

Interconnect delay τ is calculated according to Elmore relation:

$$\tau = \tau_{tr} + 0.7R_{dr}C_L + 0.7(r_{int}C_L + c_{int}R_{dr})L + 0.4r_{int}c_{int}L^2,$$

where τ_{tr} is the intrinsic switching delay of transistors, R_{dr} is the resistance of the driver, C_L is the capacitance of the load, and r_{int} and c_{int} are the intrinsic resistance and capacitance of copper wires, respectively. Length of the interconnect segment, L, must be calculated from the tile dimensions. Dimensions of the tile are obtained through running VPR for the hardest to route MCNC benchmark (pdc) and finding the number of minimum width transistors per tile for routing this circuit. This value, added to the number of minimum width transistors of the logic cluster gives the tile size and it can be used to calculate L. Delays of different parts of the logic cluster are obtained based on the nominal gate delay predicted by ITRS (see Figure 10) and by adding up the number of gates in the path of the logic segment of interest.

Component	# of transistors	Volume (in a cluster)
DFF	19	N + log(Nx(2K+2)) + log(2I+N+Nx2^{k})
Buffers	14	N
Inverters	2	I
Level shifters	4	I+N

Fig. 11. CMOS components of the hybrid logic cluster and the number of transistors in each of component.

4.2 Hybrid FPGA Parameters

—*Area.* Area of the crossbar section of the hybrid FPGA's logic cluster (nanoscale cluster) is calculated from the geometry and minimum pitch of the nanowires. This calculation considers both the area of the crossbar and the area of configuration and interface circuitry. For a cluster with N BLEs, K input LUTs and I inputs (see Figure 3), the cluster width can be computed as the number of vertical nanowires multiplied by the nanowire pitch plus the number of vertical CMOS wires multiplied by the CMOS pitch. The number of vertical nanowires in the MUX section of the cluster is $2I + N$ (I inputs, their complements and N outputs) and there are N LUTs in the cluster each having 2^K product lines. Therefore, the total number of vertical nanowires will be $2I + N + N \times 2^K$. The number of vertical CMOS wires used in decoding and configuration part depends on the CMOS-Nano interface mechanism implemented for the cluster. Assuming that, as proposed in Rad and Tehranipoor [2006b], a regular binary decoder can be used in the CMOS-Nano interface, the number of vertical CMOS wires used in decoding and configuration parts will be $log_2(N \times (2K + 2))$. Altogether, the cluster width can be calculated by:

$$Width = (2I + N + N \times 2^K) \times W_{nano} + log_2(N \times (2K + 2)) \times W_{CMOS},$$

where W_{CMOS} and W_{nano} are wire pitch in CMOS technology and pitch of nanowires used in the crossbars, respectively. By following a similar line of reasoning and counting the number of horizontal nanowires and CMOS wires, the cluster length can be calculated by:

$$Length = N \times (2K + 2) \times W_{nano} + log_2(2I + N + N \times 2^K) \times W_{CMOS}.$$

The area of crossbar section of the hybrid FPGA's nanoscale cluster is then computed as:

$$Area_{crossbar} = Width \times Length.$$

The nanoscale cluster is composed of crossbar section plus its CMOS support circuitry. Area of the CMOS support can be calculated based on the number of minimum width transistors. Figure 11 shows the number of minimum width transistors in each of the components, flip-flops, buffers and inverters, in addition to the total number of such components used in the architecture of the hybrid logic cluster. The total number of transistors is the sum of values shown in the third column. Area of the CMOS section of hybrid FPGA is calculated through the same method described for CMOS FPGA in the previous subsection.

Since CMOS support circuitry is assumed to be implemented on the substrate under the crossbar section, the total area of the hybrid cluster is the *maximum* of the area of CMOS support and crossbar section. Area of the switches used in the routing channels is obtained in the same way as described for CMOS FPGA because the proposed hybrid FPGA uses a CMOS interconnect architecture.

—*Delay*. In order to calculate delay in the hybrid FPGA we take similar steps as previously discussed for the CMOS FPGAs. Delay of the line segments of routing architecture must also be calculated. Since routing architecture of the hybrid FPGA has exactly the same implementation of CMOS FPGA, delay parameters of the routing will not be different. However, the logic cluster has completely different elements (nanowire) and architecture (crossbar) compared to CMOS clusters. Hence, delay estimation procedure must be repeated for the logic cluster. Calculation of delay between two different ends of a path in the cluster can be done through measuring resistances and capacitances of the nanowires involved between the two ends of the path. For example, delay from the input of a cluster to input of a BLE, ($T\_cluster\_input\_to\_BLE\_output$), in Figure 6 can be calculated as:

$$\begin{aligned} T\_cluster\_input\_to\_BLE\_output \; &= (R_C + R_{col}) \times C_{col} + N \times (R_{on} + R_{row}) \\ &\quad \times C_{row} + T_{CMOS\_inv}, \end{aligned}$$

where R_{row}, R_{col} and C_{row}, C_{col} are the resistances and capacitances of a nanowire row and a nanowire column in the crossbar, R_C is the nanowire-metal contact resistance, R_{on} is the resistance of the molecular switch in ON state, and T_{CMOS_inv} is delay of a CMOS inverter. Delays of other parts of the clusters of the hybrid FPGA are calculated in similar way using capacitance and resistance values of nanowires, contact resistance and resistance of the molecular switches.

Many reports have been published on experimental evaluation of the resistance of different types of nanowires and molecular switches. Depending on the type of material used in the nanowires and their diameter, the conductivity ranges from semiconducting to highly conductive metals. In our delay calculations we have used two different sets of evaluations. The first set of evaluations is made for NiSi nanowires and is based on the assumption made in DeHon [2005] that, as a first order approximation, classical resistance relations can be used for nanowires. Therefore, resistance of the nanowire can be calculated as:

$$R = \frac{4\rho L}{\pi d^2}$$

$$d = \frac{W_{nano}}{2},$$

Resistance	NiSi	CNT
Nanowire	Classic relation: $R = \frac{4\rho L}{\pi d^2}$	$\simeq 7k\Omega \times \frac{L}{L0}$ (L0 $\simeq 1\mu m$) [13]
Molecular switch in ON state	Pessimistic estimate: 100kΩ	≃20kΩ [2]
Nanowire-metal contact	Pessimistic estimate: 100kΩ	≃15kΩ [13]

Fig. 12. Resistance values of nanowires, molecular switches, and nanowire-metal contacts in the two different estimations.

where L is the length of nanowire and ρ is the resistivity of the bulk material ($10^{-5}\ \Omega cm$) for NiSi). In these evaluations we have used pessimistic estimate of 100kΩ for the resistance of molecular switch (R_{on}) and metal nanowire contact (R_C).

The second set of evaluations is based on the experimentally measured value of resistance for carbon nanotubes (CNTs) reported in Li et al. [2004] and resistance of ON state molecular switches as reported in Chen et al. [2003] and nanotube-metal contact resistance reported in Li et al. [2004]. Figure 12 shows the value of resistances used in these two evaluations, NiSi and CNT.

Capacitance of the nanowires can be calculated according to basic electrostatic relations or using a field solver. In the first set of evaluations, we have used relations reported in DeHon [2005] for nanowires mutual capacitance and capacitance between a nanowire crossing another wire (C_{junc}), which is computed by:

$$C_{junc}(d) = \epsilon \times \frac{2\pi d}{ln\left(1 + \frac{2t_{junc}}{d} \times \left(1 + \sqrt{1 + \frac{d}{t_{junc}}}\right)\right)},$$

where $d = \frac{W_{nano}}{2}$ for calculating the capacitance of crosspoint of two nanowires and $d = \frac{W_{CMOS}}{2}$ for calculating the capacitance of crosspoint of a nanowire crossing a CMOS wire. t_{junc} is the thickness of insulator between the conductors and ϵ is the permittivity of the insulator. The mutual capacitance for unit length of two neighboring nanowires ($C_{unit\_parallel\_nw}$) is calculated by:

$$C_{unit\_parallel\_nw} = \epsilon \times \frac{\pi}{ln\left(1 + \frac{2t_{sh}}{d} \times \left(1 + \sqrt{1 + \frac{d}{t_{sh}}}\right)\right)},$$

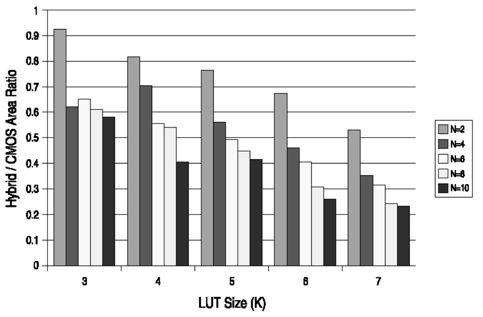
where, t_{sh} is the thickness of the shell around the conductors ($2t_{sh}$ is the distance between the conductors). Based on the above relations, one can compute capacitance of a row nanowire in a hybrid FPGA cluster:

$$C_{row} = C_{nano\_overlap} + 2C_{mutual} + C_{micro\_overlap} \quad (1)$$

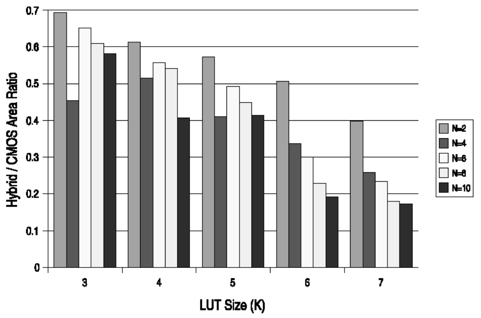
$$C_{nano\_overlap} = N_{crossed\_nws} \times C_{junc}(W_{nano}/2) = (2I + N + N \times 2^K) \times C_{junc}(W_{nano}) \quad (2)$$

$$C_{micro\_overlap} = N_{crossed\_mws} \times C_{junc}(W_{CMOS}/2) = (1 + log_2(N \times (2K + 2))) \times C_{junc}(W_{CMOS}/2) \quad (3)$$

$$C_{mutual} = Width \times C_{unit\_parallel\_nw}, \quad (4)$$



(a)



(b)

Fig. 13. Ratio of the area required for implementing MCNC benchmarks on hybrid FPGAs to the area required for implementing the benchmarks on CMOS FPGA, (a) for NiSi nanowires and (b) for SWCNTs.

where *nws* and *mws* denote nanowires and microwires, respectively. The capacitance of column nanowires of the cluster can be calculated in the same way by substituting:

$$N_{crossed\_nws} = N \times (2K + 2)$$
$$N_{crossed\_mws} = 1 + log_2(2I + N + N \times 2^K)$$

and using *Length* instead of *Width* in Equation (3). Note that *Width* and *Length* were calculated in the previous subsection.

In the second set of simulations we calculated the value of capacitance ($C_{unit\_parallel\_nw}$, $C_{nano\_overlap}$ and $C_{micro\_overlap}$) for nanotubes using a field solver [Nabors and White 1991]. These values are replaced in Equations (1–4) above to obtain row and column capacitance for the nanotubes in the second evaluation.

5. SIMULATION RESULTS

Figure 13 shows the ratio of average area required for implementing all MCNC benchmarks on hybrid FPGAs to the average area required for implementing the same benchmarks on CMOS FPGAs. Figures 13(a) and (b) show the results when NiSi nanowires and single wall carbon nanotubes (SWCNTs) are used in the hybrid FPGA, respectively. As seen, area of the circuits will considerably be smaller when implemented on the hybrid FPGA. Using SWCNTs provides more area saving compared to NiSi nanowires. This is because the diameter of SWCNTs ($\simeq 1nm$) is smaller than minimum width of implemented NiSi nanowires ($\simeq 5nm$).

Increasing the number of LUT inputs (K) results in a drastic area increase for CMOS FPGAs especially when $K > 5$. When K increases, CMOS LUTs and the intracluster MUXes will exponentially consume larger number of transistors and therefore the cluster will occupy more area. At the same time, for FPGAs with larger clusters, intercluster routing area will decrease because more logic can be mapped into each cluster and a lower number of routing tracks between clusters will be required. Since the rate of increase in cluster area is exponential with K increase, overall the area of the CMOS FPGAs increases even though the routing area decreases.

	NiSi-nanowire crossbars				SWCNT crossbars			
	N=2	N=4	N=6	N=8	N=2	N=4	N=6	N=8
K=3	8%	38%	35%	39%	31%	45%	39%	42%
K=4	18%	29%	44%	46%	39%	49%	46%	48%
K=5	24%	44%	51%	55%	43%	59%	54%	57%
K=6	32%	54%	59%	69%	49%	66%	70%	77%
K=7	47%	65%	68%	76%	60%	74%	77%	82%

Fig. 14. Percentage of area reduction for implementing benchmarks on hybrid FPGAs compared to 22 *nm* CMOS FPGA.

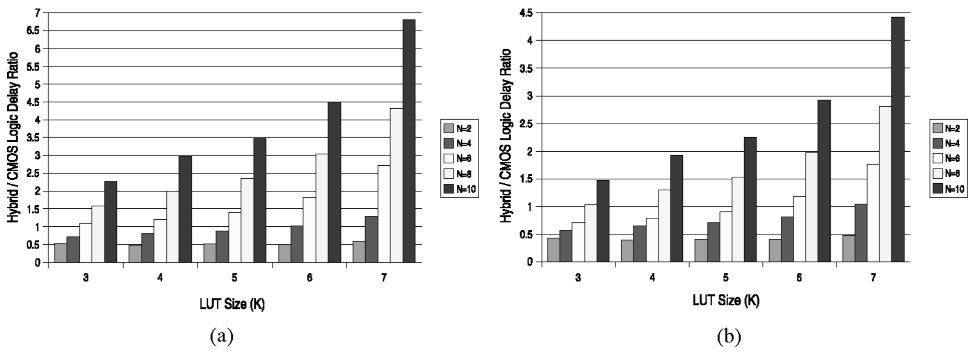
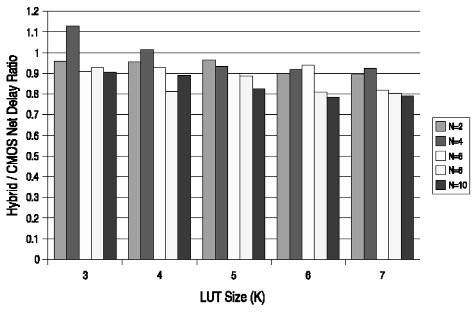


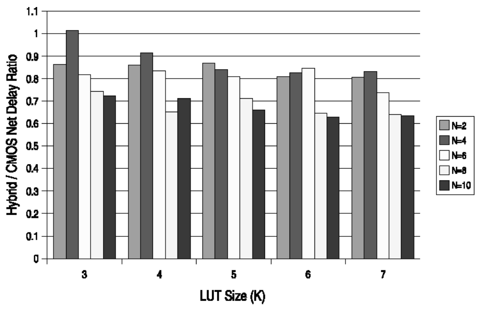
Fig. 15. Ratio of average logic delay for implementing MCNC benchmarks on hybrid FPGAs to that of CMOS FPGAs, (a) based on theoretical estimations made for crossbars made with NiSi nanowires and (b) based on experimental parasitics values for SWCNTs and molecular switches.

For hybrid FPGAs, when K increases, the increase in area of the LUT and the intra-cluster MUXes will not be significant because these parts are implemented in nanowire crossbars. In these FPGAs, the increase in intercluster routing area when K increases is almost equal to that of CMOS counterparts. Hence, the net result is that for the hybrid FPGA, the rate of area increase with enlarging clusters (increasing N and K) is much slower than that of CMOS FPGAs. As a result, the area of hybrid FPGAs will be smaller than CMOS FPGAs especially for larger cluster sizes. Figure 14 shows that area reduction of up to 82% can be achieved from implementing the benchmarks on hybrid FPGAs.

Figures 15, 16, and 17 show the ratio of logic delay, net delay, and average critical path delay for implementing benchmarks on the hybrid FPGA to that of CMOS FPGA for different values of K and N. In these figures, part (a) shows the results obtained from the theoretical estimations made for crossbars of NiSi nanowires and part (b) depicts the results for SWCNTs. In CMOS FPGAs, increasing cluster size (N) will result in increase in cluster delay because of the increased number of wires that each of the intracluster MUXes must drive. In CMOS implementation, cluster delay has an almost linear rate of growth with N. On the other hand, for the hybrid implementation, all intracluster MUXes are implemented as a single crossbar (see Figure 3). Therefore, increasing N will increase the number of inputs and outputs of the crossbar. Also implementation of a K input LUT requires 2^K vertical lines (product lines) and increasing

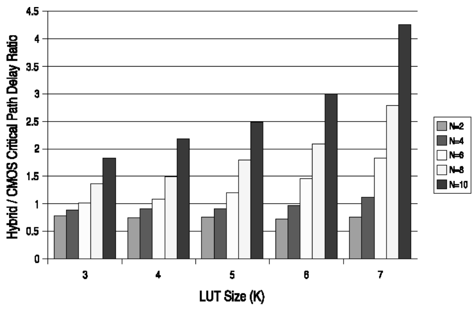


(a)

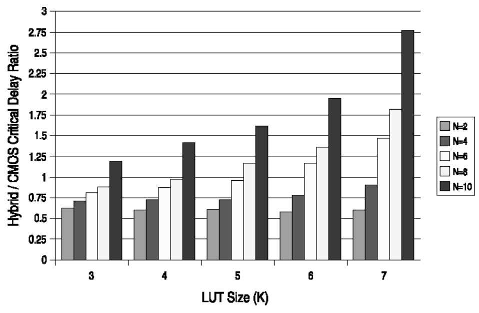


(b)

Fig. 16. Ratio of average routing delay for implementing MCNC benchmarks on hybrid FPGAs to that of CMOS FPGAs, (a) based on theoretical estimations made for crossbars made with NiSi nanowires and (b) based on experimental parasitics values for SWCNTs and molecular switches.



(a)



(b)

Fig. 17. Ratio of average critical path delay for implementing MCNC benchmarks on hybrid FPGAs to that of CMOS FPGAs, (a) based on theoretical estimations made for crossbars made with NiSi nanowires and (b) based on experimental parasitics values for SWCNTs and molecular switches.

K will exponentially increase the number of vertical lines in the cluster. Hence increasing N and K will dramatically increase the logic cluster delay. This is visible from Figure 15, where delay of hybrid logic cluster is seen to be very sensitive to N and K and becomes considerably larger than that of CMOS FPGAs for larger values of these parameters.

Since routing structure of hybrid FPGA is the same as that of CMOS FPGA, routing delay of these two FPGAs is close and will follow the same pattern for different values of N and K. However, routing delay of the hybrid FPGA is smaller than that of CMOS FPGA in almost all variations of N and K (Figure 16). This can be related to the reduced values of input capacitances for clusters in the hybrid case because of application of low capacitance nanowires in their structure.

As seen in Figure 17, when ($N \leq 4$) and ($K \leq 7$) comparable delays are obtained for hybrid FPGAs. Considering the results shown in Figure 13, it is clear that hybrid FPGAs implemented with N and K parameters in this range provide significant area reductions while they can maintain the same performance or even improve the delay.

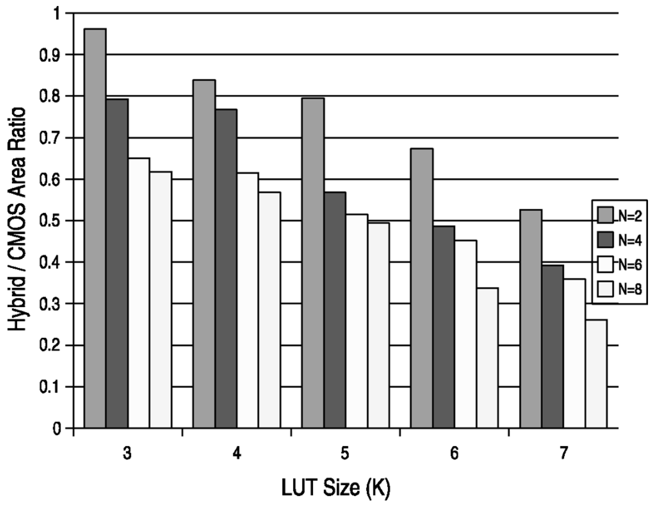


Fig. 18. Ratio of the area required for implementing larger benchmark circuits on hybrid FPGAs to the area required for implementing the benchmarks on CMOS FPGA.

We have also performed another set of experiments to show the scalability of the results for larger applications. In these experiments we used new benchmarks obtained from combining several larger MCNC benchmark circuits. Implementing MCNC benchmarks results in FPGA array dimensions of up to $30 \times 30 = 900$ clusters in the FPGA array. In the new combined benchmark set, FPGA array dimensions reaches to 55 (3025 clusters in the array). Considering that the clusters are made of N LUTs of size K, each cluster is logically equivalent to few gates (depending on N and K). Therefore, average size of the new benchmark circuits is in the range of few tens of thousands of gates to 100K gates. Even though this average size is still smaller than today's applications implemented on FPGAs, the results obtained in the new set of experiments are consistent with the previous ones. This indicates that similar results for hybrid FPGAs can be achieved for real-life applications as well.

Figure 18 presents the area ratio for implementing the new benchmarks on hybrid (NiSi based) FPGAs and CMOS FPGAs. Comparing Figure 18 with Figure 13 demonstrates that almost the same area reductions rates can be obtained for larger benchmarks. The same holds true for performance of the hybrid FPGA compared to CMOS FPGA (results obtained for the larger benchmarks indicate the same properties as those obtained from original MCNC benchmarks).

As Figure 19 demonstrates, the percentage of routing area per tile area will not have significant changes as the benchmark size (and hence the array dimension) grows. This indicates that increasing the circuit size will not change the relation between routing area and the total area. On the other hand, as figure shows, changing cluster parameters N and K has a significant effect on routing per tile area ratio and increasing N and K decreases this ratio. So, in FPGAs with higher N and K (larger clusters) the routing area to the total area will be reduced. Hence, in the hybrid FPGA where routing is made in CMOS scale and clusters are made with nanoscale crossbars, choice of larger values

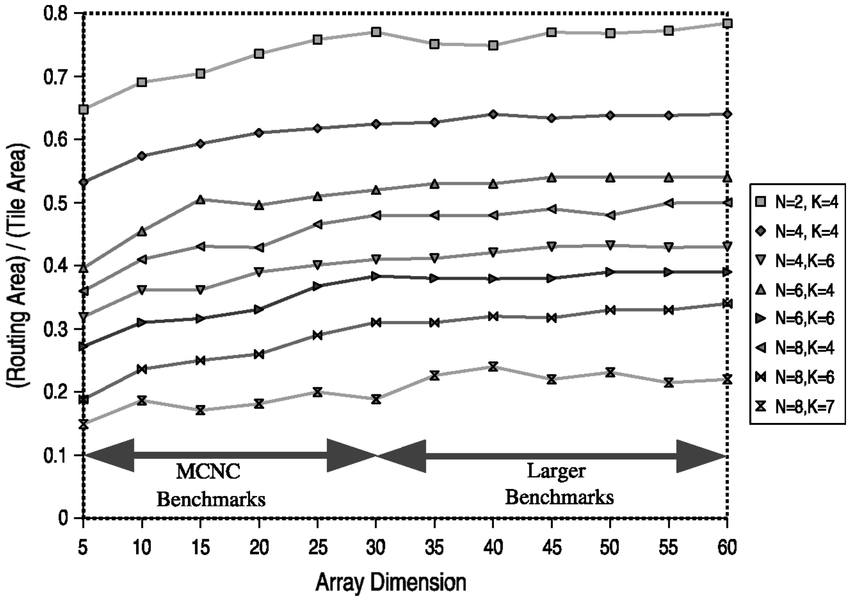


Fig. 19. Percentage of routing area per tile area for different benchmarks.

of N and K will reduce the share of routing in the overall area and therefore results in reduction of the area. This however is limited because of the adverse effect of increasing N and K on the FPGA performance as discussed in previous subsection.

Figure 20 combines the area ratio results obtained from MCNC benchmarks with those obtained from new combined benchmarks. As seen in the figure, area ratio of hybrid FPGA to CMOS FPGA has little variations as the dimensions of the FPGA array becomes larger. Therefore, it can be expected that the real life applications can have the same area advantages once implemented on the hybrid FPGAs.

6. CONCLUSION

In this article, a new hybrid FPGA was proposed comprising of nanoscale clusters and CMOS routing structure. Architectures of the nanoscale portion and the required CMOS support part of the cluster were presented. Models for area and delay calculations were also presented. These models were used to compute area and delay parameters for both CMOS FPGAs scaled to 22nm and for the hybrid FPGA. Using such delay parameters, several experiments were performed to investigate the area and delay specifications of hybrid FPGAs for different values of cluster parameters (N and K). In these experiments, MCNC benchmark circuits were implemented both on the proposed hybrid FPGA and CMOS FPGA. Hybrid FPGAs showed area reduction of up to 82% in comparison with CMOS FPGAs. As the LUT size (K) and the number of LUTs in the cluster (N) increases, the area reduction ratio increases. However, at the same time, the delay of the hybrid FPGA will increase compared to CMOS FPGA.

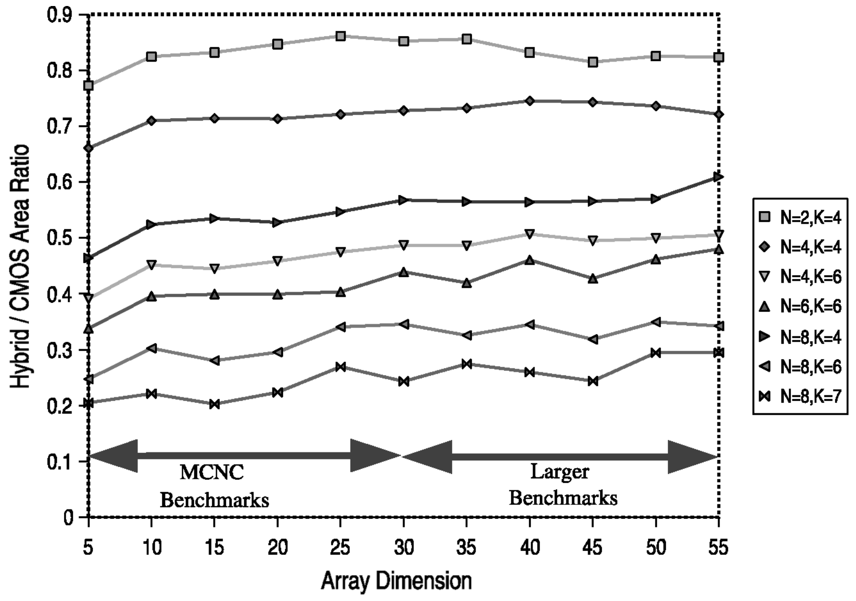


Fig. 20. Ratio of area for implementing different benchmarks on hybrid FPGA to that of CMOS FPGA.

Hybrid FPGAs with $N \leq 4$ and $K \leq 7$ are shown to have considerably reduced area and performance.

ACKNOWLEDGMENTS

The authors would like to acknowledge Dr. James Plusquellic for providing comments and discussions.

REFERENCES

AHMED, E. AND ROSE, J. 2004. The effect of LUT and cluster size on deep-submicron FPGA performance and density. *IEEE Trans. VLS Integr. Syst. 12*, 3, 3–12.

BETZ, V., ROSE, J., AND MARQUARDT, A. 1999. *Architecture and CAD for Deep Submicron FPGAs*. Norwell, MA.

CHEN, Y., OHLBERG, D. A. A., LI, X., STEWART, D. R., WILLIAMS, R. S., JEPPESEN, J. O., NIELSEN, K. A., STODDART, J. F., OLYNICK, D. L., AND ANDERSON, E. 2003. Nanoscale molecular switch devices fabricated by imprint lithography. *Appl. Phys. Lett. 82*, 10, 1610–1612.

CHOU, S. Y., KRAUSS, P. R., AND RENSTROM, P. J. 1996. Imprint lithography with 25-nanometer resolution. *Science 272*, 85–87.

CUI, Y., LAUHON, L. J., GUDIKSEN, M. S., WANG, J., AND LIEBER, C. M. 2001. Diameter-controlled synthesis of single crystal silicon nanowires. *Appl. Phys. Lett. 78*, 15, 2214–2216.

CUI, Y., ZHONG, Z., WANG, D., WANG, W. U., AND LIEBER, C. M. 2003. High performance silicon nanowire field effect transistors. *Nano. Lett. 3*, 2, 149–152.

DEHON, A. 2005. Nanowire-based programmable architectures. *ACM J. Emerg. Technol. Comput. Syst. 1*, 2, 109–162.

DEKKER, C. 1999. Carbon nanotubes as molecular quantum wires. *Phys. Today* 22–28.

GAYASEN, A., VIJAYKRISHNAN, N., AND IRWIN, M. J. 2005. Exploring technology alternatives for nanoscale FPGA interconnects. In *Proceedings of the Design Automation Conference*. 921–926.

GINGER, D. S., ZHANG, H., AND MIRKIN, C. A. 2003. The evolution of dip-pen nanolithography. *Angewandte Chemie Int. Edi.* 43, 1, 30–45.

Goldstein, S. C. and Budiu, M. 2001. NanoFabric: spatial computing using molecular electronics. In *Proceedings of the International Symposium on Computer Architecture*. 178–189.

Huang, Y., Duan, X., Cui, Y., Lauhon, L., Kim, K., and Lieber, C. M. 2001. Logic gates and computation from assembled nanowire building blocks. *Science 294*, 1313–1317.

Huang, Y., Duan, X., Wei, Q., and Lieber, C. M. 2001. Directed assembly of one-dimensional nanostructures into functional networks. *Science 291*, 630–633.

Kuekes, P. J. and Williams, R. S. 2000. Demultiplexer for a molecular wire crossbar network, U.S. Patent No. 6256767.

Li, S., Yu, Z., and Burke, P. J. 2004. Electrical properties of 0.4 cm long single walled carbon nanotubes. *Nano Lett. 4*, 2003–2007.

McAlpine, M. C., Friedman, R. S., and Lieber, C. M. 2003. Nanoimprint lithography for hybrid plastic electronics. *Nano Lett. 3*, 443–445.

Morales, A. M. and Lieber, C. M. 1998. A laser ablation method for synthesis of crystalline semiconductor nanowires. *Science 279*, 208–211.

Nabors, K. and White, J. 1991. FastCap: A multipole accelerated 3d, capacitance extraction program. *IEEE Trans. CAD 10*, 11, 1447–1459.

Rad, R. M. and Tehranipoor, M. 2006a. A hybrid FPGA using nanoscale cluster and CMOS scale routing. In *Proceedings of the Design Automation Conference*.

Rad, R. M. and Tehranipoor, M. 2006b. Fine-grained island style architecture for molecular electronic devices. In *Proceedings of the International Symposium on Field Programmable Gate Arrays*.

Semiconductor Industry Association. 2005. International technology roadmap for semiconductors (ITRS). http://public.itrs.net/.

Snider, G., Kuekes, P., and Williams, R. S. 2004. CMOS-like logic in defective, nanoscale crossbars. *Nanotechn. Inst. Phys. 15*, 881–891.

Strukov, D. B. and Likarev, K. K. 2005. CMOL FPGA: A reconfigurable architecture for hybrid digital circuits with two-terminal nano devices. *Nanotechn. Inst. Phys. 16*, 888–900.

Whang, D., Jin, S., and Lieber, C. M. 2003. Nanolithography using hierarchically assembled nanowire masks. *Nano Lett. 3*, 7, 951–954.

Received November 2006; revised March 2007; accepted April 2007