

---

## Molecular electronics and reconfigurable logic

---

I. O'Connor<sup>1\*</sup>, J. Liu<sup>1</sup>, D. Navarro<sup>1</sup>, R. Daviot<sup>2</sup>,  
N. Abouchi<sup>2</sup>, P.E. Gaillardon<sup>3</sup>, F. Clermidy<sup>3</sup>

<sup>1</sup>Lyon Institute of Nanotechnology – Ecole Centrale de Lyon  
36, avenue Guy de Collongue  
F-69134 Ecully cedex, France

<sup>2</sup>Lyon Institute of Nanotechnology – CPE Lyon  
43, boulevard du 11 novembre 1918  
F-69100 Villeurbanne, France

<sup>3</sup>CEA – LETI – MINATEC  
17, rue des Martyrs  
F-38054 Grenoble, France

\*Corresponding author  
E-mail: ian.oconnor@ec-lyon.fr

**Abstract:** This paper describes a reprogrammable architecture based on regular matrices of fine-grain dynamically reconfigurable cells based on double-gate carbon nanotube field effect transistors (DG-CNTFET) exhibiting ambivalence (p-type or n-type behaviour depending on the back-gate voltage). Using available models, reconfigurable cells have been simulated and performance metrics quantified at 4GHz operation. A function mapping method suitable for this matrix structure has been devised, and various interconnect topologies have been analyzed, showing average mapping success rates to 4x4 cell matrices of above 80% for 8-node function graphs. This gives insight into the way data functionality could be coded into the architecture based on such reconfigurable cells for on-the-fly and partial reprogrammability.

**Keywords:** carbon nanotubes, double-gate, reconfigurable logic, function mapping

**Biographical notes:** Ian O'Connor (IEEE S'95-M'98-SM'07, IEE S'87-M'98) was born in Cambridge, U.K., in 1969. He received the Ph.D. degree in Electronics from the University of Lille, France, in 1997; and the professoral dissertation (Habilitation à Diriger des Recherches) from Ecole Centrale de Lyon, France, in 2005. He joined Ecole Centrale de Lyon in 1998, where he is now Professor in the Department of Electronic, Electrical and Control Engineering. He is currently head of the Heterogeneous Microelectronics Design group at the Lyon Institute of Nanotechnology, of which he is also one of the vice-directors. Since 2008 he also holds an Adjunct Professor position at Ecole Polytechnique de Montreal, Canada. His research interests include design methods and tools for physically heterogeneous systems on chip, and their application to novel system architectures based on non-conventional devices. He has authored or co-authored around 100 book chapters, journal publications and conference papers and has been workpackage leader or scientific coordinator for several national and european projects. He also serves as an expert with the french Observatory for Micro and Nano Technologies (OMNT).

---

## 1 Introduction

It is widely recognized that transistor scaling, as a vector for the pursuit of performance levels predicted by Moore's Law and required by future applications, will not last through the next decade. Alternatives must be found, be they at the architectural level (e.g. exploring multiple core architectures) or at the device level (heterogeneous or nanoelectronic devices). In this context, the emergence of new research devices offers the opportunity to provide novel logic building blocks and to elaborate non-conventional techniques for digital design. Ultimately it will be possible to reconsider the paradigms of computing architectures to achieve orders of magnitude improvements in the conventional figure of merit (MIPS / volume\*power).

In general terms, the design of nanodevice-based circuits is a highly strategic objective, at the interface between two scientific communities: that of nanoscience and nanotechnology on one hand, and that of data processing and embedded systems on the other. While much work is underway at the technology and device level to develop radically different computing devices, there are also critical challenges from the design point of view. One is to be able to understand how such devices can best be used in architectures and indeed if they can be expected to deliver significant benefits at this level. Another major challenge is to extend existing design and simulation approaches to take into account the nanoelectronic approach. Close collaboration between designers and technologists is key to the strengthening of mutual design approaches necessary for the development of nanoelectronic systems and the generation of truly original designs exploiting the specific properties of nanodevices. The outcome of such collaboration is a clearer understanding of choices among the broad spectrum of potential devices and possible technologies capable of challenging conventional approaches in future nanoscale applications.

Indeed, it is expected that the necessary structuring of the projected tens of billions of elementary, unreliable, nanometric devices to achieve the computing capacities necessary for future software applications will lead to the emergence of reconfigurable platforms as the principal computing fabric before the end of the next decade. The reconfigurable approach allows volume manufacturing and reduces the impact of the evolution of mask costs, projected to move above the \$10M mark in 2010. It can also efficiently cover a broad range of applications while exceeding performance levels of programmable systems, and couples naturally to fault-tolerant design techniques for robust architectures.

The future organization of reconfigurable cells in a system is uncertain, but integration density and switchbox overhead concerns are a growing issue. These point to the rising probability of fixed interconnect topologies between individual cells organized into clusters, and the use of switchboxes or network approaches only between clusters of cells. In parallel, the unreliability of individual devices will lead to a loss of accessible functions in certain cells. This can be circumvented by reformulating cluster configurations based on the incomplete set of identified operators. In this context, the development of methods capable of mapping complex functions onto clusters of reconfigurable cells with incomplete sets of operators is a key milestone to exploiting the full potential of future reconfigurable systems.

In addition, recent technological breakthroughs have led to the proposal of area- and power-efficient reconfigurable cells based on emerging devices such as double-gate carbon nanotube transistors (CNTFET) with incomplete operator sets [1]. CNTFETs have attracted much attention in recent years, and benchmark figures against state-of-the-art planar and non-planar silicon logic transistors are favourable. They have shown in particular that the high mobility, achievable current density, theoretical transition frequency and  $I_{on}/I_{off}$  ratio place CNTFETs among the most promising nanodevices in line to succeed the MOS transistor from the standpoint of their integration into future nanoelectronic systems on chip [2]. Some work has been carried out to explore the use of the unique properties of multiple diameter and ambipolar CNTFETs with respect to CMOS for new computing paradigms ([3], [4]). The emergence of double gate devices, with four accessible terminals, also opens the way to solutions specifically exploiting the additional terminal for reconfigurability purposes. In the case of the double gate CNTFET [5], completely new prospects for reconfigurability are possible due to its ambivalent (n- and p-type) behaviour. Using this property, logic cells can be built that offer fine-grain reconfigurability not available with MOSFET technology, at comparable or better speed and power figures, and improving over current reconfigurable systems in terms of the number of devices used to realize a single function.

It is therefore useful to combine such novel types of nanodevice-based reconfigurable cell with the exploration of new function mapping methods in anticipation of the deployment of incomplete-operator cluster-based systems. A primary objective will be to analyze the limits of such architectures when mapping a complex software application onto it.

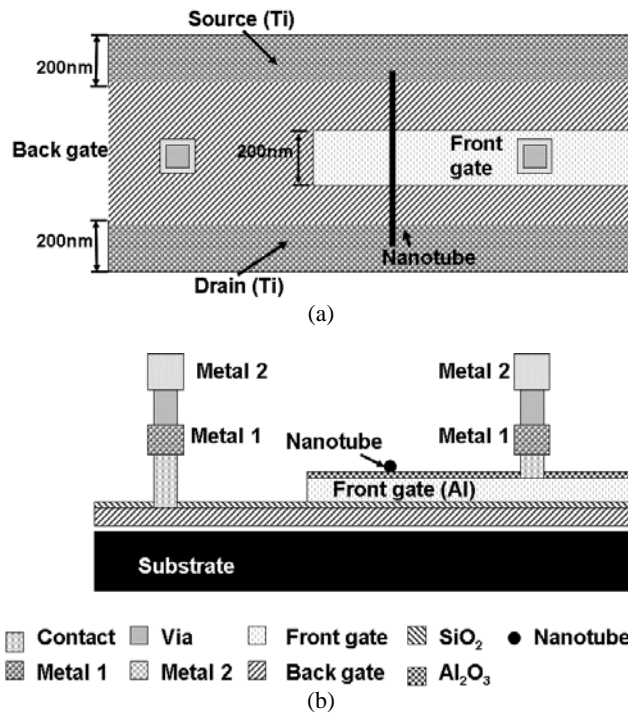
This paper begins by describing the structure and properties of a DG-CNTFET based dynamically reconfigurable logic cell to be used in a functional matrix in section 2. We propose in section 3 a method to map function graphs to matrices of such reconfigurable cells for on-the fly and partial reprogrammability. Finally in section 4, we apply the method to various intra-matrix topologies and evaluate the most suitable one in terms of the success rate of function mapping to topologies.

## 2 Dynamically reconfigurable logic gates based on double-gate CNTFET

### 2.1 Double-Gate CNTFET and electronic behaviour

For high-performance FETs, short gate lengths and high channel mobility are required. Since nanotubes typically exhibit very small diameters (allowing excellent gate control) without suffering from mobility degradation, they are promising candidates to overcome the limitations of nanometric silicon devices. Figure 1 shows the structure of a novel DG-CNTFET [5], fabricated with an aluminium front gate placed under the nanotube between the contacts of the source and the drain and controlling the electrostatics and switching of the nanotube bulk channel in region B. The Schottky barriers (SB) at the nanotube/metal contacts are controlled by the silicon back gate (substrate), which also prevents the electrostatics in region A from being influenced by the front gate. The SBs at the contacts are not affected by the front gate voltages.

**Figure 1** Double-Gate CNTFET structure [2] (a) top view (b) cross-sectional view



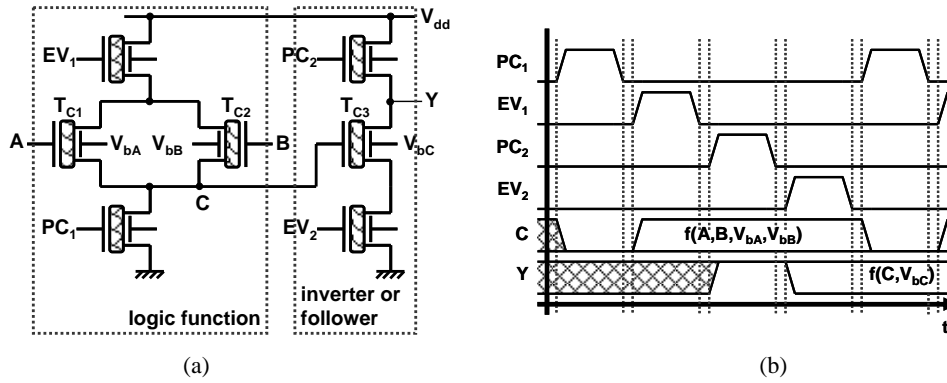
The behaviour of this DG-CNTFET device is strongly dependent on the potential of the silicon back gate, which we call  $V_{gs-bg}$ :

- when  $V_{gs-bg}$  is sufficiently negative (some hundreds of mV), the device functions like a p-type FET with a negative threshold voltage.
- when  $V_{gs-bg}$  is sufficiently positive (some hundreds of mV), the device functions like an n-type FET with a positive threshold voltage;
- when  $V_{gs-bg}$  is floating, the sub-bands with the contacts are not affected by the bias of the front gate, and the device is in the off state with a very weak current ( $I_{off} < 100fA$ ).

## 2.2 Dynamically reconfigurable logic cells

The impact of the back gate voltage polarity on the transistor channel transport characteristics opens up new opportunities for using CNTFETs in logic circuits. We have built a reconfigurable logic block which can be configured to any one of fourteen basic binary operation modes. This functionality is impossible to achieve in CMOS technology without resorting to far more complex circuitry (and therefore silicon real estate and system power) than the cell structure described here. The polarity (n-type or p-type) of each transistor is controlled by the back-gate bias voltage values. The dynamically reconfigurable logic cell (DRLC\_7T) is shown in Figure 2(a); while Table I describes the 3-input configuration, corresponding basic binary logic functions and power figures for operation at 4GHz and 250MHz extracted from transient simulations using a Verilog-A model adapted from [1] and parasitic capacitances extracted from layout estimations.

**Figure 2** Dynamically reconfigurable logic cell (DRLC\_7T) (a) transistor-level schematic (b) clock signal scheme



DRLC\_7T is made up of 7 CNTFETs arranged in two logic stages: the first stage performs an elementary logical operation and the second stage works either in follower or inverter mode.

- A and B are boolean data inputs (voltages at A and B vary between 0V and 1V);
- $V_{bA}$ ,  $V_{bB}$ ,  $V_{bC}$  are control inputs which configure the circuit according to Table I (control bias voltages may take one of three values at -1V, 0V and 1V);
- $PC_1$ ,  $PC_2$  (pre-charge) and  $EV_1$ ,  $EV_2$  (evaluation) are four non-overlapping clocking inputs with pre-charge and evaluation periods as in classical CMOS dynamic logic gates;
- Y is the circuit output.

We can see that  $V_c$  is evaluated between  $EV_1$  (evaluation of the first logical stage) and the next  $PC_1$  (pre-charge of the first logical stage) according to the value of inputs A and B; and Y is evaluated and maintained between  $EV_2$  (evaluation of the second logical stage) and the next  $PC_2$  (pre-charge of the second logical stage). This clocking scheme is illustrated in Figure 2(b).

An example illustrates how this logic gate works. When  $V_{bA}=V_{bB}=V_{bC}=1V$ , CNTFETs  $T_{c1}$ ,  $T_{c2}$  and  $T_{c3}$  (shown in Figure 2) are all configured as n-type FETs, as indicated in the previous section. When  $PC_1$  is enabled, the first stage is pre-charged, and the voltage of the internal node C ( $V_c$ ) is discharged to 0V. If for example either of the data inputs A or B=logic "1", then when  $EV_1$  is enabled, the first layer evaluates its output such that the internal node C is set to logic "1". Then  $PC_2$  is enabled (pre-charge of the second stage), and the output Y is charged to logic "1"; and when  $EV_2$  is enabled, the output is evaluated and Y is evaluated to logic "0". In fact in this configuration, the only situation where C is not set to logic "1" and Y therefore evaluates to logic "1" (since  $T_{c3}$  is off) is when both A and B=logic "0". This shows that for  $V_{bA}=V_{bB}=V_{bC}=1V$ , DRLC\_7T is configured as a NOR operator, as specified in Table 1. Simulation results of DRLC\_7T in this configuration are shown in the left half of Figure 3 (up to 8ns) at 500MHz operation. After this point,  $V_{bA}$ ,  $V_{bB}$  and  $V_{bC}$  change to -1V, such that CNTFETs  $T_{c1}$ ,  $T_{c2}$  and  $T_{c3}$  are all configured as p-type FETs. When  $PC_1$  is enabled, the first stage is

pre-charged, and the voltage of the internal node C ( $V_c$ ) is discharged to 0V. If for example either of the data inputs A or B=logic "0", then when  $EV_1$  is enabled, the first layer evaluates its output such that the internal node C is set to logic "1". Then  $PC_2$  is enabled (pre-charge of the second stage), and the output Y is charged to logic "1"; and when  $EV_2$  is enabled, the output is evaluated and Y is evaluated at logic "1". The only situation here where C is not set to logic "1" and Y therefore evaluates to logic "0" (since  $Tc_3$  is on) is when both A and B=logic "1". This shows that for  $V_{bA}=V_{bB}=V_{bC}=-1V$ , DRLC\_7T is configured as a NAND operator.

**Table 1** 3-input configurations for reconfigurable cell with 3 logic levels (+V, 0, -V) and corresponding 14 basic binary logic functions

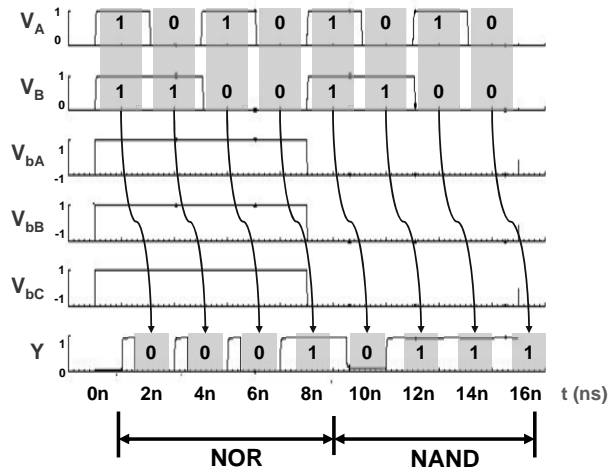
$V_{bA}$	$V_{bB}$	$V_{bC}$	Y	Consumption @4GHz(nW)	Consumption @250MHz(nW)
+V	+V	+V	!(A+B)	1.87	1.076
+V	+V	-V	A+B	1.85	0.99
+V	0	+V	!A	1.83	0.84
+V	0	-V	A	1.81	0.82
-V	-V	+V	A.B	1.84	0.9
-V	-V	-V	!(A.B)	1.82	0.814
+V	-V	+V	(!A).B	1.86	1.05
+V	-V	-V	A+(!B)	1.84	0.96
0	+V	+V	!B	1.82	0.8
0	+V	-V	B	1.79	0.79
0	0	0	1	1.12	0.04
0	0	-V	0	1.82	0.2
-V	+V	+V	A.(!B)	1.84	1.03
-V	+V	-V	B+(!A)	1.82	0.95

It is thus clear that this gate can realize several functions and can be dynamically reconfigured during the calculation.

Table I also gives the power consumption when DRLC\_7T (working with 2-logic-stage control bias voltage) operates at 250MHz and 4GHz.

It should be noted that as this is prospective work, no technology as yet exists to build this circuit, although a single DG-CNTFET has been fabricated and characterized [5]. Further, significant technological advances have been made recently to achieve 95-98% horizontally aligned semiconducting CNTs [6] and, separately, hybrid integration with CMOS [7]. This enables us to envisage systems using "substrates" of many aligned semiconducting CNTs with conventional metallization and lithography techniques creating interconnections. In terms of device design, much work has focused on improving drive current (and therefore maximum frequency and insensitivity to noise) and reliability by using an array of parallel single-walled carbon nanotubes as multiple channels in a single transistor with good directional and spatial control [8]. This device can pass currents of up to 1.5mA and has achieved a record current gain cutoff frequency of 8GHz. These performances are still dominated by parasitics but recent advances project that this device should reach a current gain cutoff frequency of 31GHz. Double-gate transistors using the same principle for the channel should not pose any technological obstacle.

**Figure 3** Simulation results of dynamic reconfiguration of reconfigurable cell DRLC\_7T from NOR operator to NAND operator

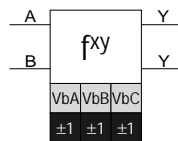


Here we have given only one example of the family of dynamically reconfigurable logic cells. By changing the number of the transistors we have developed 3 other logic cells which can realize different logic function sets [1].

### 3. Interconnect topology and function mapping method

Such fine-grain reconfigurable gates open the way towards structures which can be configured dynamically, for on-the-fly and partial system reprogrammability. In this section, the elementary building blocks and the way the data is coded are open to question to achieve increased efficiency at application level. The symbol used for each reconfigurable cell is shown in Figure 4.

**Figure 4** Symbol for reconfigurable cell DRLC\_7T

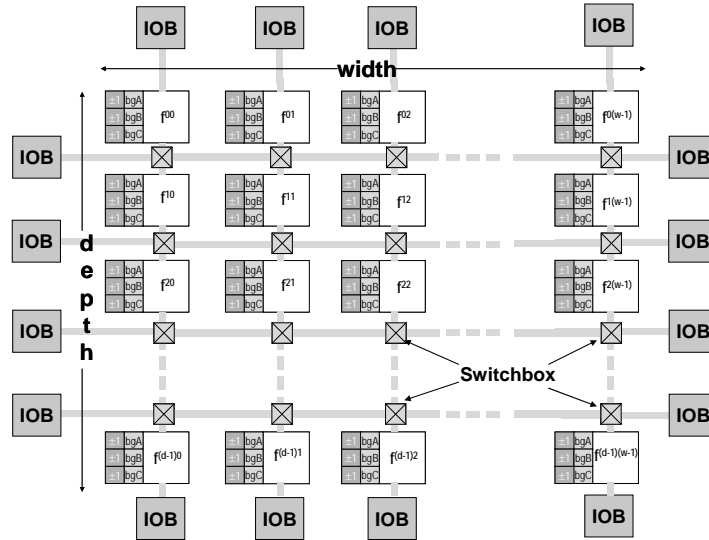


The function and  $\{x,y\}$  coordinates of the cell within the matrix is given by  $f^{xy}$ ; back gate voltages of  $\pm 1V$  are represented by  $V_{bA}$ ,  $V_{bB}$ ,  $V_{bC}$ ; inputs A and B are shown, as is the output Y (duplicated); precharge and evaluation connections are not shown to avoid making the figure overly cumbersome.

#### 3.1 Fixed-matrix interconnect topology strategy

In a conventional architecture, each calculation cell would be connected to the switch box directly, as shown in Figure 5.

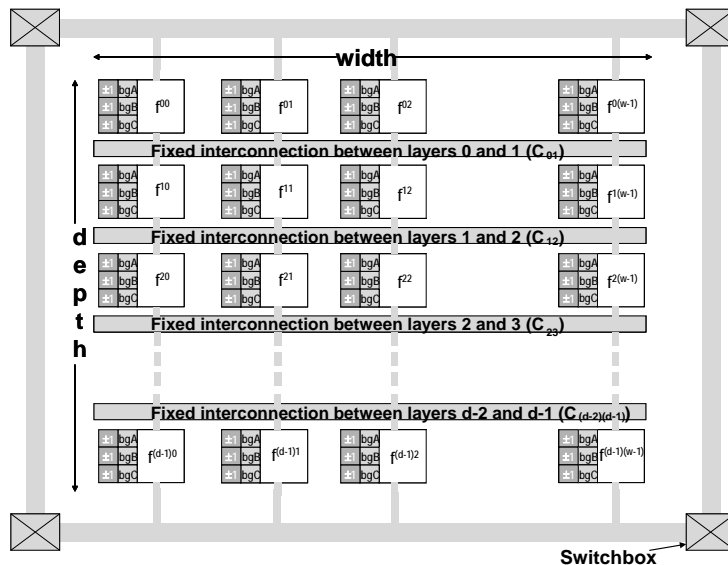
**Figure 5** Fine-grain connection topology between reconfigurable cells DRLC\_7T and data switch boxes



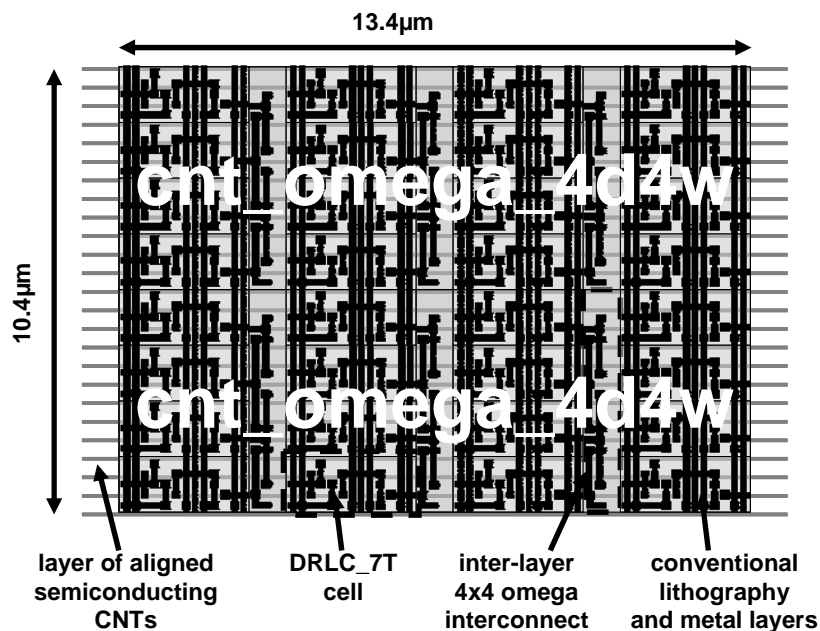
However, this would lead to a loss of efficiency because the size of the reconfigurable cell is quite small with respect to the size of the switch box, and would thus lead to large overheads. Thus, building reconfigurable cell matrices with fixed intra-matrix interconnect, and subsequently interconnecting the matrices to the switch boxes seems to be a solution. Such a matrix with an association of physically identical cells and an example layout are presented in Figure 6 and Figure 7 respectively.

To reach on-the fly and partial reprogrammability, a new method is necessary to map a specific function dynamically to one pre-defined matrix (as part of the whole architecture). In the following section, we will give an example to show how this method works. We consider a matrix of 4 cell depth and width ( $4d4w$ ) using a banyan interconnect topology. Individual cross-connectivity matrices  $X_{mn}$  ( $X_{01}$ ,  $X_{12}$  and  $X_{23}$ ) between logic cell stages of depth  $m$  to  $n$  are shown in Figure 8.

**Figure 6** Matrix of reconfigurable gates DLRC\_7T with fixed interconnect topology



**Figure 7** Layout of 2 4x4 (4d4w) reconfigurable cell DLRC\_7T matrices with fixed intra-matrix interconnect



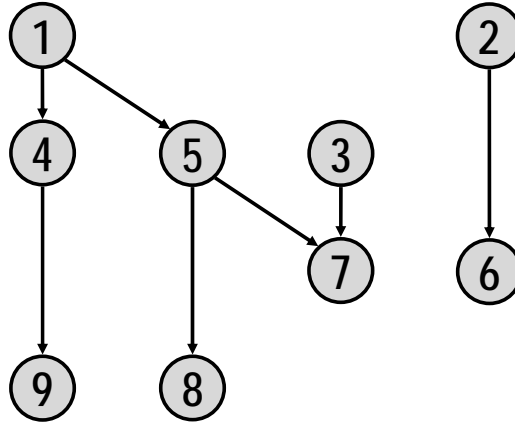
**Figure 8** Connectivity matrices for logic layers 0 to 1, 1 to 2 and 2 to 3 in 4d4w matrix using banyan topology

$$X_{01} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \quad X_{12} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad X_{23} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

### 3.2 Mapping method

The function to map is presented as a graph (an example is shown in Figure 9), generated by a random graph generator. In the adjacency matrix (shown in Figure 10), (i, j) refers to the intersection of the row i and column j. A 1 at the position (i, j) means that the point i is connected to the point j. This matrix is essential to subsequent processing steps.

**Figure 9** Example function graph to map onto the 4d4w interconnect topology



**Figure 10** Adjacency matrix of the original function graph

$$\begin{bmatrix} p1 \\ p2 \\ p3 \\ p4 \\ p5 \\ p6 \\ p7 \\ p8 \\ p9 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p1 \\ p2 \\ p3 \\ p4 \\ p5 \\ p6 \\ p7 \\ p8 \\ p9 \end{bmatrix}$$

In the first step, we identify logic layers, and divide the adjacency matrix into 16 small matrices according to the different layers (shown in Figure 11). Between the 16 matrices,  $C_{01}$ ,  $C_{02}$ ,  $C_{03}$ ,  $C_{12}$ ,  $C_{13}$ ,  $C_{23}$  are 6 matrices containing 1's to be considered.  $C_{m,n}$  is the adjacency matrix between the points in the logic layer m and those in the logic layer n. We therefore pay particular attention to  $C_{02}$ ,  $C_{03}$ ,  $C_{13}$  (where  $n \neq m+1$ ) because the presence of any non-zero element in these three matrices signifies a connection which

jumps at least one logic layer. Such a direct connection cannot be realized in the topology since the interconnect topology is physically fixed. In our example, the connections between points (2, 6), (5, 8) and (4, 9) are the three connections to be adjusted. The solution is to add synchronization elements, then repeat the process until no more connections jump logic layers.

**Figure 11** Adjacency matrix of the graph after layer identification

		$C_{01}$	$C_{02}$	$C_{03}$		
		↓	↓	↓		
$p1$	0	0	0	1	1	$p1$
$p2$	0	0	0	0	0	$p2$
$p3$	0	0	0	0	0	$p3$
$p4$	0	0	0	0	0	$p4$
$p5$	0	0	0	0	1	$p5$
$p6$	0	0	0	0	0	$p6$
$p7$	0	0	0	0	0	$p7$
$p8$	0	0	0	0	0	$p8$
$p9$	0	0	0	0	0	$p9$

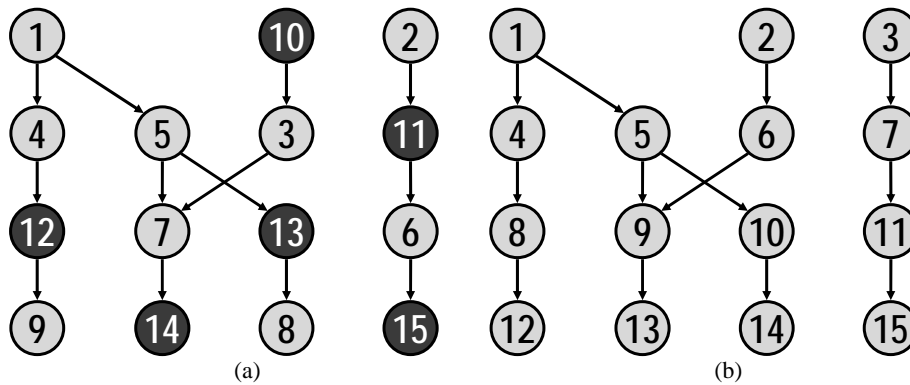
$C_{12}$

$C_{13}$

$C_{23}$

In the second step, we add an intermediate point between the extremities of a long connection and divide it into two short connections. In our example, points 11, 12, 13 are added into the graph, so the connection between the points 2 and 6 becomes 2 connections (between points 2 and 11, 11 and 6), the connection between the points 4 and 9 becomes 2 connections (between points 4 and 12, 12 and 9), and the connection between the points 5 and 8 becomes 2 connections (between points 5 and 13, 13 and 8). The function graph after synchronization is shown in Figure 12(a) and its adjacency matrix is shown in Figure 13.

**Figure 12** Function graph (a) after synchronization (b) after renumbering



**Figure 13** Adjacency matrix of the function graph after synchronization

$$\begin{array}{c}
 \begin{array}{l}
 p1 \\
 p10 \\
 p2 \\
 p4 \\
 p5 \\
 p3 \\
 p11 \\
 p12 \\
 p7 \\
 p13 \\
 p6 \\
 p9 \\
 p14 \\
 p8 \\
 p15
 \end{array}
 \end{array}
 =
 \begin{array}{c}
 \begin{array}{cccc|cccc|cccc}
 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
 \end{array}
 \end{array}
 \begin{array}{l}
 p1 \\
 p10 \\
 p2 \\
 p4 \\
 p5 \\
 p3 \\
 p11 \\
 p12 \\
 p7 \\
 p13 \\
 p6 \\
 p9 \\
 p14 \\
 p8 \\
 p15
 \end{array}$$

In the third step, we first reassign the number of each point (shown in Figure 12(b)) and assign each point to the cells in the matrix, defined by the topology. The process starts from the first layer.

- the function graph is firstly analyzed in sequence, meaning that for each node in the structure, children branches are identified and recursively explored. This step results in the order of the points to assign. In our example, this is:

$$p1-p4-p8-p12-p5-p9-p6-p2-p13-p10-p14-p3-p7-p11-p15$$

- each layer's connections is then compared to the relevant inter-layer connectivity matrix and allows the assignment of functions to cells. In the example, the first point  $p_1$  is assigned to the cell  $f^{00}$ . According to the path defined in the previous step,  $p_4$  is the next point to assign to a cell in the matrix. Since  $f^{00}$  is physically connected to  $f^{10}$  and  $f^{12}$ , the cell with lower y-index (here  $f^{10}$ ) is arbitrarily chosen for  $p_4$  assignment, and the other possibility is memorized. Branching (i.e. the exploration of the immediately preceding alternative) is used when the arbitrary choice leads to a dead-end, and the process is repeated until all functions are assigned to cells. In our example, the final programmed matrix is shown in Figure 14.

**Figure 14** Complete 4d4w matrix and function assignment after mapping

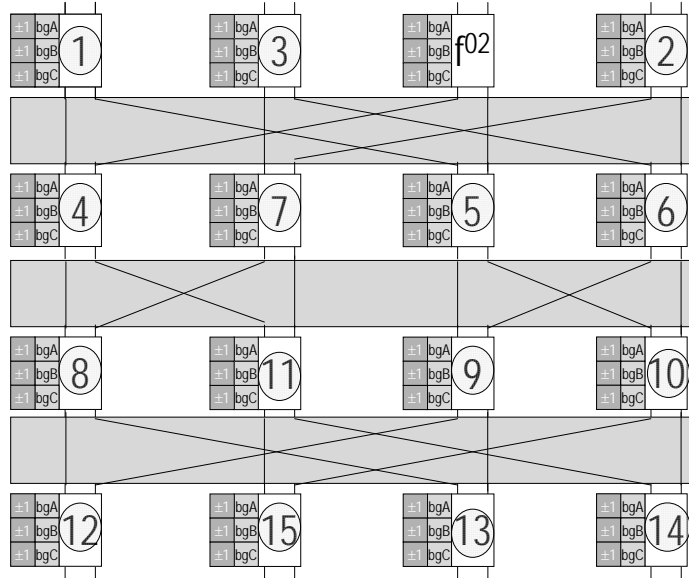
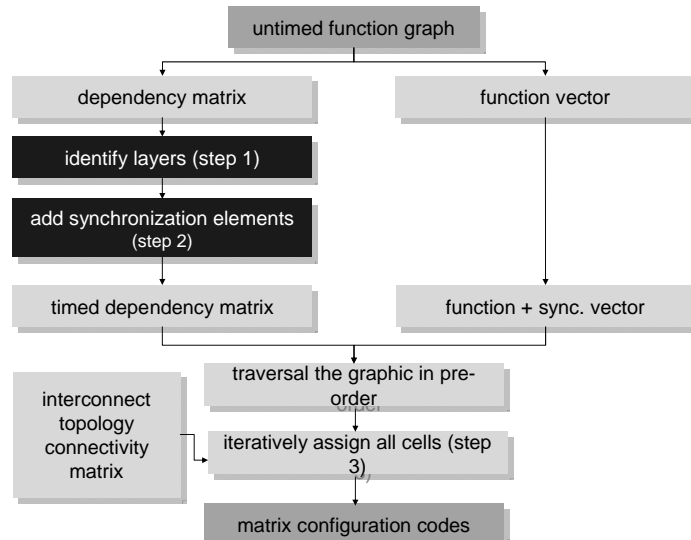


Figure 15 summarizes the complete function mapping method. While this approach enables the mapping of simple functions to the fixed-interconnect matrix based on the reconfigurable cells, function partitioning and merging methods are required to map more complex functions over several matrices.

**Figure 15** Steps in function mapping method

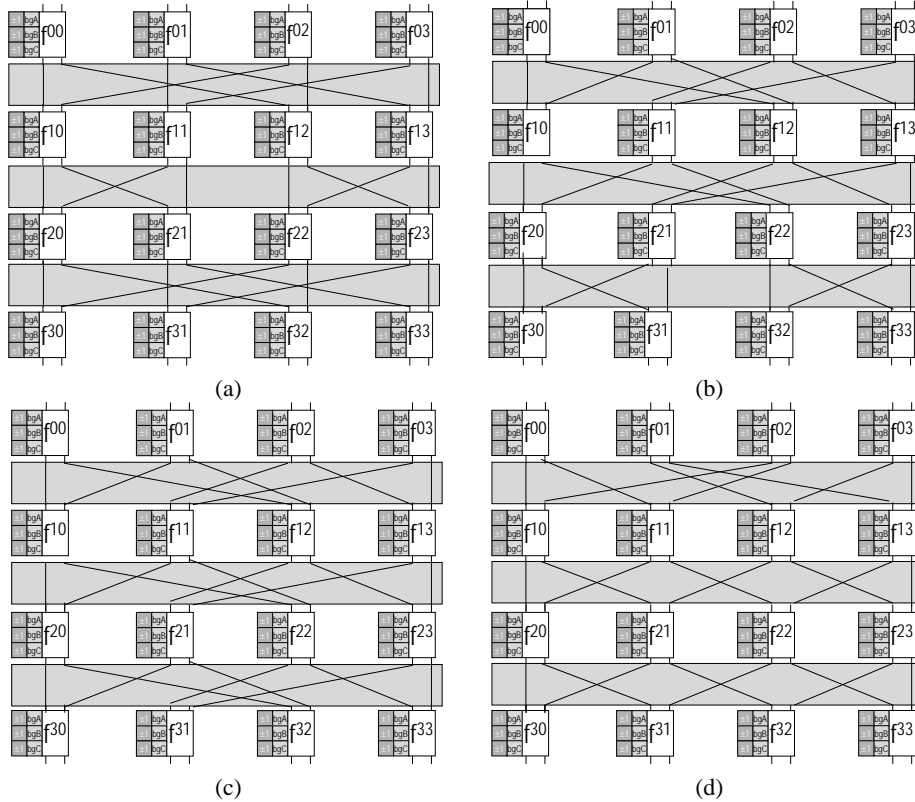


This method thus enables the dynamic configuration of matrices in the reconfigurable computing architecture, based on double-gate CNTFET reconfigurable logic cells.

#### 4. Application to interconnect topologies and results

In the previous section, we gave an example of a matrix with banyan interconnect topology to study the function mapping method. Some other intra-matrix interconnect topologies such as baseline (shown in Figure 16(b)), flip (shown in Figure 16(c)), omega (shown in Figure 16(d)) also exist. The aim of this part of our work was to evaluate and compare the rate of success of mapping function graphs to a 4d4w matrix using various intra-matrix interconnect topologies.

**Figure 16** 4d4w interconnect topologies (a) banyan (b) baseline (c) flip (d) modified omega



We have carried out detailed analyses to compare the efficiency of the different intra-matrix interconnect topologies. We use a random graph generator to generate sets of function graphs containing 6-16 points. Each set, corresponding to a given number of points in the function graph, contains 1000 samples. Using the previously described mapping method, each function is programmed onto the 4d4w matrix using the various intra-matrix interconnect topologies, and the success rate is calculated.

**Figure 17** Comparison of programmability success rates for banyan, modified-omega, flip and baseline interconnect topologies within 4d4w matrices

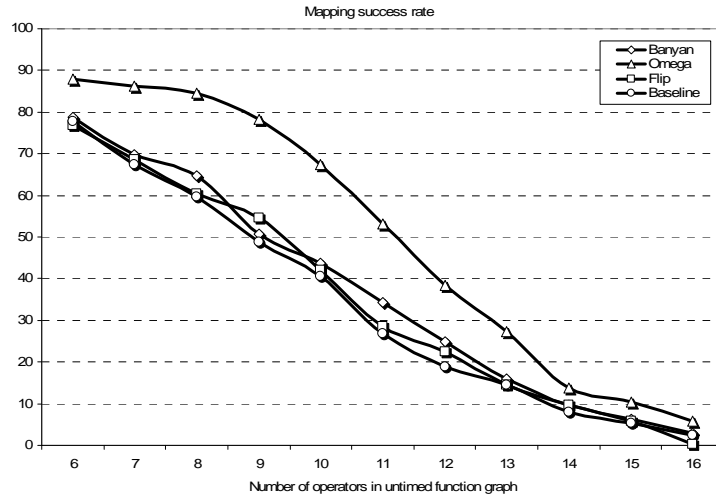


Figure 17 shows the comparison of programmability for 4d4w banyan, omega, flip and baseline topologies. For banyan, flip and baseline interconnect topologies, the success rate is about 80% when the function graphs have 6 points. At 12 points, the success rate is about 25%. There is no great difference between these two topologies. However for the modified-omega interconnect topology, the success rate is about 90% for 6-point function graphs and about 40% for 12-point graphs. This clearly shows that the modified-omega interconnect topology is more suitable for this type of matrix. This is because this topology is less symmetric than the other topologies and spreads calculations over cells occupying less width, which seems to correspond better to typical function graphs. In fact in the matrix, there are pairs of cells which have the same inputs. For two cells which have the same inputs, the sum of the number of function they can achieve is 14. For two cells for which the inputs are different, the sum of the number of functions they can achieve is  $14 + 14 = 28$ . In the banyan topology, there are 6 pairs of cells which have the same inputs, while in the omega topology, there are only 2. That is why the omega topology has the potential to realize more functions than other topologies. However, this topology cannot be chosen systematically when we change the size of the matrix, especially the depth. Future work will involve analysis of the interconnect topology for matrices of various sizes, and evaluating the matrix size best able to achieve efficient programming.

## 5. Conclusion

This paper has firstly introduced the opportunities opened up by the electrical behaviour of the double-gate CNTFET focusing primarily on a specific property, namely the enabling of p-type or n-type device behaviour to be achieved in the same CNTFET according to the voltage applied to the back-gate. We have developed a family of dynamically reconfigurable logic cell, based on these devices and configured by the set of

back gate bias voltages. These fine-grain reconfigurable cells have been considered as a universal reconfigurable cell enabling the synthesis of any Boolean function. We have proposed a new method to map specific functions to the matrices of reconfigurable cells. Finally, we have studied different intra-matrix topologies and shown that the mapping success rate is about 90% for 6-point function graphs and about 40% for 12-point graphs when using the omega interconnect topology. This new function mapping method is a key step towards using fine-grain reconfigurable cells for the on-the fly and partial reprogrammability. Future work will focus on random interconnect topologies representing bottom-up fabrication processes, and on fault-tolerance at the matrix level.

### Acknowledgements

This work is supported by the French National Agency (ANR) through the Embedded Systems and Large Infrastructures Program (Project NANOGRAIN no. ANR-08-SEGI-012-01).

### References

- 1 O'Connor, I. et al. (2007), 'CNTFET Modeling and Reconfigurable Logic-Circuit design', *IEEE Trans. Circuits and Systems*, Vol. 54, No. 11, pp.2365-2379
- 2 Chau, R. et al., (2005) 'Benchmarking Nanotechnology for High-Performance and Low-Power Logic Transistor Applications', *IEEE Trans. Nanotechnology*, Vol. 4, No. 2. 153.
- 3 Raychowdhury, A., Roy, K., (2005) 'Carbon-Nanotube-Based Voltage-Mode Multiple-Valued Logic Design', *IEEE Trans. Nanotechnology*, Vol. 4, No. 2, pp. 168-179.
- 4 Sordan, R., Balasubramanian, K., Burghard, M., Kern, K., (2006) 'Exclusive-OR gate with a single carbon nanotube', *Appl. Phys. Lett.*, Vol. 88, 053119.
- 5 Lin, Y., Appenzeller, J., Knoch, J., Avouris, P., (2005) 'High-Performance Carbon Nanotube Field-Effect Transistor With Tunable Polarities', *IEEE Trans. Nanotechnology*, Vol. 4, No. 5, September 2005
- 6 Ding, L., Tselev, A., Wang J., et al., (2009) 'Selective Growth of Well-Aligned Semiconducting Single-Walled Carbon Nanotubes', *Nano Lett.*, Vol. 9, No. 2, p. 800.
- 7 Akinwande, D., Yasuda, S., Paul, B., Fujita, S., Close, G., Wong, H.S.P., (2008) 'Monolithic integration of CMOS VLSI and carbon nanotubes for hybrid nanotechnology applications', *IEEE Trans. Nanotechnology*, Vol. 7, No. 5, p. 636.
- 8 Beyhous, J.-M., Happy, H., Dambrine, G., Derycke, V., Goffman, M., Bourgoin, J.-P., (2006) 'An 8-GHz  $f_t$  Carbon Nanotube Field-effect transistor for Gigahertz Range Applications', *IEEE Electron Device Letters*, Vol. 27, No. 8, pp. 681-683.