Crosstalk-Computing based Gate-Level Reconfigurable Circuits

Naveen Kumar Macha, Bhavana Tejaswini Repalle, Md Arif Iqbal, Mostafizur Rahman

***Abstract*— The functionality of Polymorphic circuits can be altered using a control variable. Owing to the multi-functional embodiment in polymorphic-circuits, they find a myriad of useful applications such as reconfigurable system design, resource sharing, hardware security, and fault-tolerant circuit design, etc. The polymorphic circuit approaches available in literature so far are either based on custom nonlinear circuit designs or based on special emerging devices such as ambipolar FET, configurable magnetic devices, etc. While some of these approaches are inefficient in performance, the other approaches involve exotic devices. We have proposed a novel polymorphic circuit design approach based on Crosstalk-Computing, where deterministic signal interference between nano-metal lines is leveraged for logic computation and reconfiguration purposes. In this paper, we elaborate upon the polymorphic circuit design in Crosstalk-Computing, present a comprehensive list of polymorphic logic gates designed, and characterize and benchmark our circuits with respect to CMOS circuit implementations. The ability to design a wide range of polymorphic logic circuits (basic and complex logics) that are compact in design and minimal in transistor count is unique to Crosstalk Computing, which leads to benefits in the circuit Power, Performance and Area (PPA). Our circuit designs, simulation, and PPA characterization results show that the polymorphic crosstalk circuits provide 3x improvement in transistor count, 2x improvement in switching energy, and 1.5x improvement in speed for polymorphic logic circuits. In a best-case, the transistor count reduction is 6x. We present a cascaded circuit example of a polymorphic Multiplier-Sorter-Adder circuit and benchmark it with respect to the CMOS implementation.**

***Index Terms*— Crosstalk-Computing, Crosstalk Circuits, Reconfigurable Circuits, Polymorphic Circuits.**

I. Introduction

POLYMORPHIC logic circuits are rich in their functional behavior, where a control variable can deterministically morph the circuit's behavior between multiple functions [1]. For example, an AND gate can change as an OR gate and vice-versa. Thanks to their ability to transform intrinsically, polymorphic circuits find their use in a myriad of applications [2-8] such as reconfigurable circuits/systems design[2][3], resource sharing[2][3], multifunctional adaptive systems [4], hardware security[5], fault tolerance [6], and self-test circuits[7][8][9]. Besides, as scaling down of feature size in Integrated Circuits (ICs) is approaching the physical limits, the miniaturization trend of ICs (Moore's Law) is relaxing. Therefore, developing alternate techniques that try to push the horizons of Moore's Law can be of tremendous potentials. Polymorphic circuits can be one such technique that tries to sustain Moore's Law because they increase the circuit functionality in a given footprint by reusing the circuits to execute different functions. However, the tradeoffs in achieving such polymorphic circuits make them go or no-go for applications.

To be considered as a truly polymorphic circuit, along with the innate multifunctional nature, the control between different functions should be enabled by inherent device characteristics and/or external environmental influences [1]. The Polymorphic circuits evolved using genetic algorithms [13] were extensively researched. These evolved circuits can morph their functional behavior based on different environmental control variables such as temperature, supply-voltage, control-signal, light, radiation, etc. They find interesting applications in sensor circuits that morph and adapt their behavior in different environments [14], especially in extreme conditions of temperature, radiation, microwaves, etc. (for example in space electronics) [15][16]. The disadvantages [17] they face are technology dependency, scalability and inefficiency in speed and power, because of which the evolved circuits do not find applications in mainstream digital circuits/systems.

Another approach actively pursued is chaos computing [18], in which non-linear dynamics in transistors and circuits are captured to implement multifunctional circuits. But these circuits are custom nonlinear/mixed-signal circuit designs for digital circuits. More recently, polymorphic circuits are also designed using emerging tunable polarity transistors [19-23], which can be configured either as p-type or n-type based on a control signal. These morphable transistors foster various fine-grained polymorphic circuit schemes. However, these novel devices require complex device engineering compared to mainstream CMOS devices, and the circuit schemes necessitate additional circuitry to switch the power rails when the transistors change as p-type/n-type [20]. The other alternate approaches using emerging spintronic devices were also proposed [5], but they rely on complex information encoding schemes through spin-polarized currents and bipolar voltages, etc. Consequently, they are a significant departure from existing computational device and circuit paradigms.

We have proposed a novel computation concept called Crosstalk-Computing (CT-Computing) [24] and implemented a wide range of compact and efficient polymorphic circuits in this approach [25]. In the CT-Computing technique, the signal interference between adjacent metal lines is astutely engineered to a logic principle. In contrast to the traditional Switch based logic computations, a deterministic superposition of crosstalk coupled input signals produces the logic operation in CT-Computing. The counterpart of connecting switches/transistors in different patterns to achieve different logic is tuning the crosstalk coupling capacitances in CT-Computing. For polymorphism, an additional control signal is used, which

biases the circuit to alter its functional behavior [23]. We have also explored the application potentials of CT-Polymorphic circuits in [26] and [27].

In this paper, we discuss the CT-Computing concept in detail and present a comprehensive list of crosstalk polymorphic circuit designs and their simulation responses. The polymorphic gates shown are AND-OR, AND-AO21(AND-OR-21, i.e. (AB)+C), AND-OA21 (OR-AND-21, i.e. (A+B)C), AND-CARRY(AB+BC+CA), OR-AO21, OR-OA21, OR-CARRY, AO21-OA21, CARRY-AO21, CARRY-OA21, and Inverter-Buffer (Inv-Buf). We have also presented a cascaded polymorphic circuit example of a 2-bit Multiplier-Sorter-Adder designed using the above Crosstalk-Polymorphic (CT-Polymorphic) logic gates. Finally, we characterized the performance metrics of Crosstalk-Circuits (CT-Circuits) and benchmarked them with respect to CMOS implementations.

The rest of the paper is organized as follows. Section II discusses the CT-Computing concept, provides the intuition for logic implementation and presents the circuit techniques to implement basic and complex logic circuits. Section III discusses polymorphism in CT-Computing and shows a wide range of CT-Polymorphic gates that can morph between different operations. In Section IV we present the comparison of CT-Polymorphic circuits with other approaches in the literature. In Section V we present the benchmarking results. Finally, Section VI concludes the paper.

II. Crosstalk Computing

The Fig.1(i) shows an overview of CT-Computing Fabric, which majorly comprises of four components, Crosstalk Layer (CTL), Active Devices, Interconnects, and Vias. The CT layer which computes the logic is a metal layer/layers comprised of capacitively coupled metal lines called Aggressors (*Ag*) and Victim (*Vi*). Interconnects and Vias serve their regular purpose, along with their contribution to coupling capacitance in CTL. The active devices depicted are FinFETs on SOI substrate. The purpose of transistors is to accurately control and reconstruct the signals, which would be discussed in the following sections. The aggressors serve as inputs, and the victim serves as the output. The Fig.1(ii) illustrates the aggressor-victim scenario of crosstalk-logic. It shows the capacitive interference of the signals for logic computation— the transition of the signals on two aggressor metal lines (*Ag1* and *Ag2*) induce a resultant summation charge/voltage on the victim metal line (*Vi*) through capacitive couplings C_C. Since this phenomenon follows the charge conservation principle, the victim net voltage is deterministic and possesses the information about signals on two aggressor nets; its magnitude depends upon the coupling strength between the aggressors and the victim net. The coupling capacitance is directly proportional to the relative permittivity of the dielectric and lateral area of metal lines (which is length multiplied the vertical thickness of metal lines) and inversely proportional to the distance of separation of metal lines. Tuning the coupling capacitance values using the variables mentioned above provides the engineering freedom to tailor the induced summation signal to the specific logic implementation.

Fig.1(iii) depicts the notion of implementing logic gates (AND and OR) using crosstalk signal interference. Input signal transitions induce a voltage proportional to coupling capacitances. As shown in Fig.1(iii), for AND gate, the inputs coupling, C_A, can be chosen such that the magnitude of the voltage induced on output net (*Vi*) is greater than a selected threshold voltage V_T only when both inputs transition from 0 to 1 (i.e., for input combination 11). The threshold voltage V_T differentiates logic 0 and 1. When only one of the input transitions (input combinations 01 and 10), the voltage induced on the victim net is below the V_T; hence, the output can be considered as logic 0. Thus, as shown in Fig.1(iii), AND gate functionality can be realized using the crosstalk signal interference mechanism. Similarly, OR gate functionality can be realized just by increasing the coupling capacitance, which can be done by appropriately tuning the physical dimensions or choosing high-k dielectric material or both. The intuition for OR gate implementation is also shown in Fig.1(iii). Compared to AND gate, for OR gate, the coupling capacitance C_O is increased ($C_O > C_A$) such that the transition of either of the input signal from 0 to 1 is now sufficient to induce a voltage above the logic threshold (V_T). Therefore, input combinations, 01, 10, and 11 evaluates to logic output 1, as an OR gate. Practical realization of Crosstalk-Logic (CT-logic) circuits and their reliable and robust operation in cascaded circuits requires additional circuit techniques to be augmented to the intuitive idea described above, which is presented next.

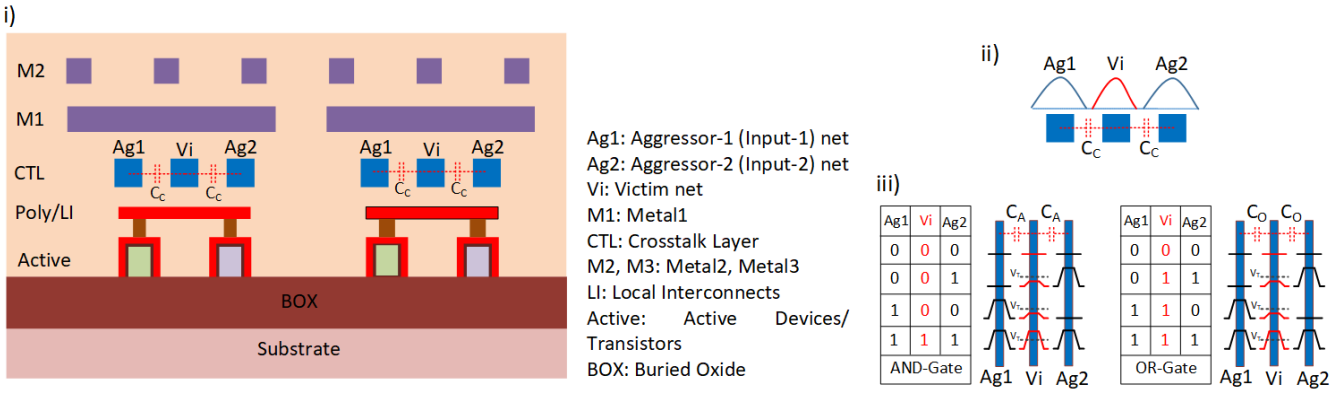

Fig.1: (i) Abstract view of Crosstalk-Computing fabric, (ii) Crosstalk-Computing Mechanism, (iii) Implementing Logic Gates based on CT-Computing

A. Basic Logic Gates

Although properly engineered and coupled nano-metal lines are sufficient to emulate the logic behavior [24] in CT-Computing, the output net (*Vi*) which collects the crosstalk charge needs to satisfy three conditions to achieve deterministic functionality in all sorts of real circuit environments. First, the *Vi* net needs to start from a known initial state. Second, it should remain floating during logic evaluation to collect the crosstalk charge. Third, the output node should be able to drive the fanout gates in real circuits and maintain the signal integrity of binary voltage levels. As shown in Fig.2, the first two conditions are met by connecting a discharge transistor to the *Vi* net, and the third condition is met by adding an inverter to the *Vi* net. Fig.2(i) shows the 2-input AND gate in which input aggressor nets (*A* and *B* acting as *Ag1* and *Ag2*) are coupled to *Vi* net through coupling capacitances of value C_C (C_C values are given in Table.1). The *Dis* signal drives the discharge transistor. The

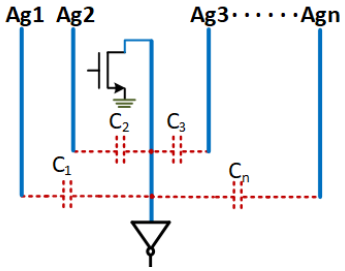

Fig.3 Capacitive Network in a Generic Crosstalk Gate

CT-logic gates operate in two alternate states, Discharge state (DS) and Logic Evaluation state (LE). During DS (enabled by *Dis* signal), the floating victim node is shorted to ground through the discharge transistor and thus starts with a known initial condition, i.e., 0. The alternate DS states ensure the correct logic operation during every LE state by clearing off the charge from the previous logic operation. During LE state (*Vi* net is floating) the rise transitions on aggressor nets induce a proportionate linear summation voltage on *Vi* net which is connected here to a CMOS inverter. The inverter acts as a regenerative threshold function. That is if the voltage computed on *Vi* net is above the inverter's threshold-voltage/trip-point (V_{INV}), it outputs the logic level 0, and vice-versa; It regenerates the signals and restores them to full swing. Also, V_{INV} is tunable by changing PMOS to NMOS width ratios if required. The CT-logic gates presented in thi

Gate	C_C (fF)	Aggressor Weights		Margin Fuction	Width Ratio (PMOS:NMOS)
		w1	w2		
AND2	0.8	1	1	$CT_M(2C_C)$	1 : 1
OR2	4	1	1	$CT_M(C_C)$	1 : 3

s paper are designed using the 16nm Predictive Technology Modeling (PTM) transistors and simulated on SPICE. The simulation response of the designed AND gate is shown in Fig.2(iii), where the first panel shows the discharge signal (*Dis*), the second panel shows two input signals (*A* and *B*) with *00*, *01, 10, and 11* combinations given through successive LE stages (i.e., when *Dis=0*), and the third panel shows the output response of the AND gate. For all the circuits, the *FI* node gives inverting logic output (NAND, NOR, etc.), and the *F* node gives a noninverting logic output (AND etc.). Similarly, OR gate implementation is shown in Fig.2(ii) and simulation response is shown in panel-4 of Fig.2(iii) (the input signals in panel-1 and panel-2 are shown common for both the circuits to limit the space). The difference between AND and OR gates is that the coupling strength (C_C), as given in the Table. I, is greater for OR gate than AND; i.e., 0.8fF and 4fF for AND and OR, respectively. C_C is the quantized capacitance specific to each gate. The input aggressors would receive the coupling strengths in integer multiples of C_C.

The operation of CT-logic gates would be represented functionally using a crosstalk-margin function, $CT_M(C_C)$, which specifies that the inverter of the CT-logic gate flips its state only when victim node sees the input transitions through the total coupling greater than or equal to C_C. For example, AND gate CT-margin function is $CT_M(2C_C)$. It states that the inverter flips its state only when the victim node sees the input transitions through total coupling greater than or equal to $2C_C$, which happens only when both inputs are high. Similarly, for the OR gate (Fig.3(i)), the CT-margin function is $CT_M(C_C)$; which means the transition of any one of the aggressors is sufficient to flip the inverter, thus execute the OR behavior.

The CT-Circuit design and working mechanism for all logic types will be explained using the CT-margin function in this paper. Therefore, to further elucidate the relationship between CT-margin function and working mechanism of CT-logic gates, consider a generic crosstalk capacitive network with 'n' number of input aggressors as shown in Fig.3. The voltage induced on victim net can be calculated, by applying KVL, as follows

$$V_{Vi} = \left(\frac{C_1}{C_T}V_1 + \frac{C_2}{C_T}V_2 \ldots + \frac{C_n}{C_T}V_n\right) \ldots \text{(I)}$$

Where, $C_1, C_2 \ldots C_n$, are capacitances from respective aggressors to the *Vi* net. C_T is the total capacitance on *Vi* net,

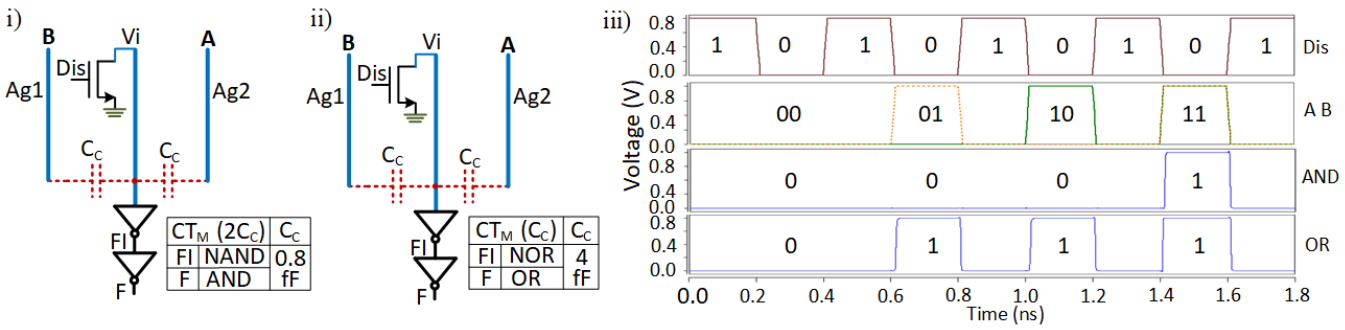

Fig.2 Crosstalk Basic Gates: (i) AND Gate Circuit Schematic, (ii) OR Circuit Schematic, (iii) Simulation response of AND and OR gates.

which is,

$$C_T = C_1 + C_2 \ldots + C_{INV} + C_{ds};$$

C_{INV} = Inverter Gate Capacitance,

C_{ds} = Discharge transistor drain to source capacitance

The final voltage levels on input aggressors, which are given by $V_1, V_2 \ldots V_n$ in equation (I), can be formulated as voltage sources, given by,

$$V_i = L_i V_{DD};$$

Where, L_i represents the logic level, i.e.,

$$L_i = \begin{cases} 0 & if\ logic\ 0 \\ 1 & if\ logic\ 1 \end{cases}$$

The capacitances given to input aggressors are in integer multiples of a constant C_C specific to each gate. Therefore, $C_i = w_i * C_C$; where, w_i is the integer multiplying factor representing the weighted strength of each aggressor. The equation (I) now modifies to

$$V_{Vi} = \frac{C_C}{C_T}.V_{DD}.m \ldots \text{(II)}$$

Where, $m = w_1L_1 + w_2L_2 \ldots + w_nL_n$. m evaluates to integer values. The CT-margin function of each gate can be related to *Vi* net voltage as follows. Consider given logic gate is associated with the CT-margin function $CT_M(k.C_C)$ (k takes integer values), then for all the input combinations that produce logic output 0, the *Vi* net voltage computed is greater than inverter trip point ($i.e., V_{Vi} > V_{INV}$) and m is greater than or equal to k (i.e., m ≥ k). Similarly, for all input combinations that produce logic output 1, m is less than k *(i.e., $m < k$)* and *Vi* net voltage is less than inverter trip-point ($V_{Vi} < V_{INV}$). Table I gives the logic design table for AND2 and OR2 gates, which lists the C_C values, aggressor weights, and margin function. It also lists the PMOS to NMOS ratio for two gates. The logic design table summarizes the mechanism and circuit aspects of crosstalk logic gates.

B. Complex Logic Gates

By increasing the fan-in (i.e., the number of input aggressors), more interesting complex logic functions can be realized because of the increased coupling capacitances and CT-margin function choices. The circuit schematic of a generic 3-input Crosstalk gate is shown in Fig.4(i). Table. II is a Logic Design table that lists C_C and w_i values, CT-margin function, and PMOS to NMOS width ratio for all 3-input complex-logic

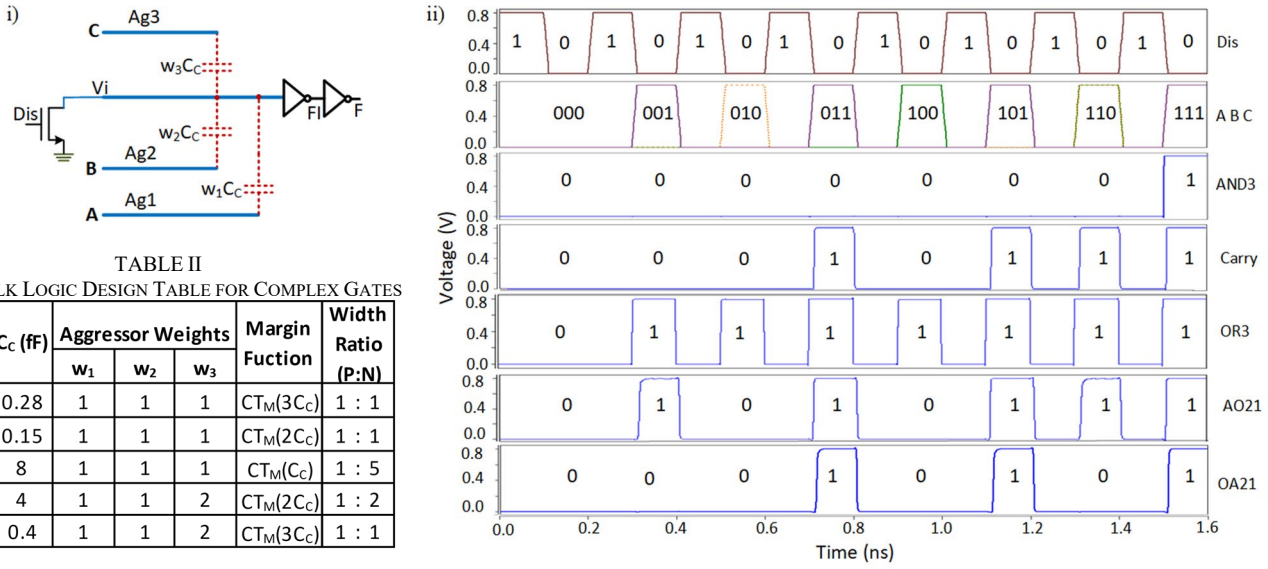

TABLE II
CROSSTALK LOGIC DESIGN TABLE FOR COMPLEX GATES

Gate	C_C (fF)	Aggressor Weights			Margin Fuction	Width Ratio (P:N)
		w_1	w_2	w_3		
AND3	0.28	1	1	1	$CT_M(3C_C)$	1 : 1
CARRY	0.15	1	1	1	$CT_M(2C_C)$	1 : 1
OR3	8	1	1	1	$CT_M(C_C)$	1 : 5
AOI21	4	1	1	2	$CT_M(2C_C)$	1 : 2
OAI21	0.4	1	1	2	$CT_M(3C_C)$	1 : 1

Fig.4 Crosstalk Complex logic Gates: (i) A generic schematic representing all 3-input complex logic functions ii) Simulations response of 3-input complex logic functions (AND3, CARRY, OR3, AO21, OA21).

functions that are implemented. For logic design, each gate receives a specific quantized C_C value and different aggressor weights(w_i) as given in the table. The input aggressors can be assigned equal or unequal coupling capacitances. Gates with equally coupled aggressors are called homogeneous CT-Logic gates and unequally coupled aggressors are called heterogeneous CT-Logic gates. These homogenous and heterogeneous coupling choices further enhance the scope of complex logic functions that can be implemented efficiently through the CT-Computing mechanism.

III. CROSS-TALK POLYMORPHIC LOGIC GATES

It can be observed from the circuit schematics, Table. I and Table. II that unlike CMOS circuit style where we have fixed patterns of series and parallel connection of switches (transistors) for each logic type, CT-logic circuits are of uniform pattern with the only difference in their coupling capacitances. That means, if the coupling capacitances from inputs to the *Vi* net can be altered at runtime, the logic behavior of the gate can also be altered. This ability to alter the runtime logic behavior could pave the way to design a new kind of polymorphic/reconfigurable logic circuits based on CT-Computing. Instead of trying to achieve the run-time alteration of coupling capacitances by controlling material properties or by constructing novel devices for this purpose, an alternate path can be chosen where the *Vi* net is coupled with an additional control aggressor (*Ct*). The transition of the signal on *Ct* would augment an extra charge/voltage on to the *Vi* net, which is equivalent to run time alteration of the capacitance coupled to the *Vi* net. This extra voltage induced on the *Vi* net would actually disturb the intended logic behavior of the gate. However, if this extra voltage induced is engineered properly,

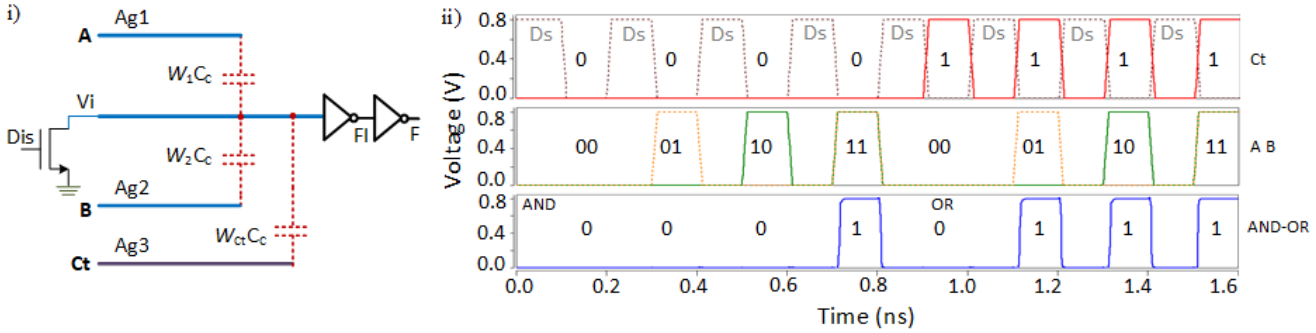

Fig.5 2-input Crosstalk-Polymorphic Logic Gate: i) AND2-OR2 Schematic, ii) AND2-OR2 Simulation response

the logic behavior of the gate can be astutely morphed such that a new functional pattern can emerge and give rise to polymorphic gates. We show polymorphism between all the logic functions discussed in the previous section; homogeneous to homogeneous logic: *AND2-OR2*, *AND3-OR3*, *AND3-CARRY*, *OR3-CARRY*; heterogeneous to heterogeneous logic: *AO21-OA21*; homogeneous to heterogeneous logic: *AO21-AND3*, *AO21-OR3*, *AO21-CARRY*, *OA21-AND3*, *OA21-OR3*, *OA21-CARRY*.

For a generic CT-Polymorphic gate, the control aggressor *Ct* will be coupled to *Vi* net through capacitance $w_{Ct}C_C$ (w_{Ct} is the weight signifying the control aggressor's strength). The *Vi* net voltage equation (II) now turns to,

$$V_{Vi} = \frac{C_C}{C_T}.V_{DD}.(m + w_{Ct}L_{Ct})$$

$$Where,\ m = w_1L_1 + w_2L_2 \ldots + w_nL_n, \text{ and}$$

$$L_{C\,t} = \begin{cases} 0\ if\ control\ signal\ Ct\ \ is\ low\ voltage \\ 1\ if\ control\ signal\ Ct\ \ is\ high\ voltage \end{cases}$$

The CT-margin function is an abstraction for logic behavior in CT-Computing. Therefore, the transformation of the CT-logic gate's behavior from one function to the other function would also mean that there is an effective transformation in their margin-functions. The CT-Polymorphic logic gate evaluates to 0 (at node *FI*) only when, $V_{Vi} > V_{INV}$. The aggressor weights and C_C are tuned such that $V_{Vi} > V_{INV}$ only when,

$$m \geq (k - w_{Ct}L_{Ct})$$

Therefore, for a CT-Polymorphic gate to evaluate to 0 at the output node *FI*, the input logic levels (L_i), thus *m* should satisfy the following conditions,

$$When\ L_{C\,t} = \begin{cases} 0, & m \geq (k) \\ 1, & m \geq (k - w_{Ct}) \end{cases}$$

Therefore, the CT-margin function transforms as follows,

$$When\ L_{C\,t} = \begin{cases} 0, & CT_M(k.C_C) \\ 1, & CT_M((k - w_{Ct}).C_C) \end{cases}$$

In other words, when $L_{C\,t} = 0$, the inverter can flip its state only when it receives the voltage through a total coupling capacitance of $k.C_C$; therefore, the gate's logic behavior corresponds to the margin function $CT_M(k.C_C)$. However, when $L_{C\,t} = 1$, an extra voltage would be induced through capacitance $w_{Ct}.C_C$, leaving only $(k\text{-}w_{Ct})C_C$ capacitance margin; i.e., the inverter can now flip its state just with the voltage induced due to capacitance greater than or equal to (k- w_{Ct})C_C. Therefore, the margin function and its corresponding logic behavior will be transformed to $CT_M((k\text{-}w_{Ct})C_C)$.

We have implemented various 2-input and 3-input CT-polymorphic logic circuits. Fig.5 shows the CT-Polymorphic AND2-OR2 Circuit and its simulation response. Table.III presents the circuit design parameters for AND2-OR2 gate,

TABLE III
CROSSTALK LOGIC DESIGN TABLE FOR AND2-OR2 GATE

Gate	C_C (fF)	Ag Weights			L_{Ct}	Margin Function	Logic Function	Width Ratio
		w_1	w_2	w_{Ct}				PMOS:NMOS
AND2-OR2	1	1	1	1	0	$CT_M(2C_C)$	AND2	1 : 1
					1	$CT_M(C_C)$	OR2	

which are C_C, input and control aggressors' weights, and PMOS and NMOS widths ratio. The table also presents the effective transformation of CT-margin function with respect to control logic $L_{C\,t}$ and its corresponding function. It can be observed

TABLE IV
CROSSTALK LOGIC DESIGN TABLE FOR 3-INPUT POLYMORPHIC GATES

Gate	C_C (fF)	Aggressor Weights				L_{Ct}	Margin Function	Logic Function	Width Ratio (P:N)
		w_1	w_2	w_3	w_{Ct}				
AND3-OR3	1	1	1	1	2	0	$CT_M(3C_C)$	AND3	1 : 2
						1	$CT_M(C_C)$	OR3	
AND3-CARRY	0.9	1	1	1	1	0	$CT_M(3C_C)$	AND3	1 : 1
						1	$CT_M(2C_C)$	CARRY	
CARRY-OR3	4.5	1	1	1	1	0	$CT_M(2C_C)$	CARRY	1 : 3
						1	$CT_M(C_C)$	OR3	
OA21-AO21	0.7	1	1	2	1	0	$CT_M(3C_C)$	OA21	1 : 2
						1	$CT_M(2C_C)$	AO21	
AND3-AO21	0.28	1	1	2	2	0	$CT_M(4C_C)$	AND3	1 : 2
						1	$CT_M(2C_C)$	AO21	
AND3-OA21	0.21	1	1	2	1	0	$CT_M(4C_C)$	AND3	1 : 2
						1	$CT_M(3C_C)$	OA21	
OA21-OR3	0.97	1	1	2	2	0	$CT_M(3C_C)$	OA21	1 : 3
						1	$CT_M(1C_C)$	OR3	
AO21-OR3	3	1	1	2	1	0	$CT_M(2C_C)$	OA21	1 : 5
						1	$CT_M(1C_C)$	OR3	
CARRY-AO21	2.2	2	2	3	1	0	$CT_M(4C_C)$	CARRY	1 : 2
						1	$CT_M(3C_C)$	AO21	
OA21-CARRY	0.6	2	2	3	1	0	$CT_M(5C_C)$	OA21	1 : 1
						1	$CT_M(4C_C)$	CARRY	

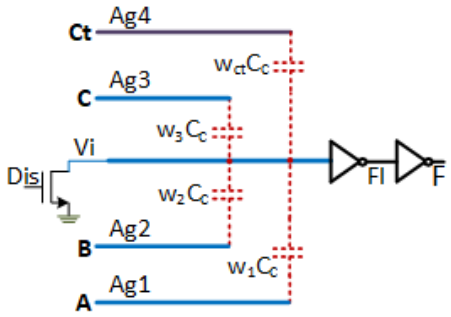

Fig.6 Generic 3-input Crosstalk-Polymorphic Logic Gate Schematic

from the simulation response (Fig.5) of the circuit that when $L_{C\,t} = 0$ the circuit responds as OR gate, whose behavior is abstracted to CT-margin function $CT_M(2C_C)$ in the table. But when $L_{C\,t} = 1$, the circuit responds as AND gate, whose behavior is abstracted to $CT_M(C_C)$ in the table. Next, we have implemented ten different types of 3-input polymorphic circuits which are listed in Table.IV. In order to limit the space, all these circuits are represented by single schematic in Fig.7 as all of these gates have uniform circuit topology with only difference in their design parameters. Table.IV lists all the circuit-design parameters for different gates. The simulation response of all the circuits are presented in Fig.7, where the first panel shows *Dis* and *Ct* signals; the second panel shows the input combinations fed through *A*, *B* and *C*; and rest of the panels show the response of different gates at node *F*. For *AND3-OR3* circuit, the inputs *A*, *B*, *C*, has the same coupling C_C (i.e., w_1=w_2= w_3=1), while *Ct* aggressor receives $2C_C$ capacitance (i.e., w_{Ct}=2). When $L_{C\,t} = 0$, the margin function for *AND3-OR3* gate is $CT_M(3C_C)$, which makes it behave as *AND3* as shown in Fig.7 panel-3. Whereas, when $L_{C\,t} = 1$, the *Ct* aggressor augments an extra charge through coupling capacitance $2C_C$ and effectively manipulates the margin function to $CT_M(C_C)$. Following the function $CT_M(C_C)$, the transition of either *A* or *B* or *C* is now sufficient to flip the inverter; thus, the gate biases and operates as an *OR3* gate as shown in Fig.7 panel-3. It can be observed that the circuit responds as *AND3* when $L_{C\,t} = 0$, for first eight input combinations (*000* to *111*), whereas, it responds as *OR3* when $L_{C\,t} = 1$, during next eight combinations (*000* to *111*). For *AND3* gate, if control aggressor is given just C_C coupling strength instead of $2C_C$ in the previous case, $CT_M(3C_C)$ manipulates to $CT_M(2C_C)$, which becomes polymorphic *AND3-CARRY* gate as given the table. The corresponding simulation response is in Fig.7 panel-4. The next gate is *AO21-OA21* which is a heterogeneous to heterogeneous logic. The coupling weights of aggressors are w_1=w_2=1, w_3=2 and w_{ct}=1 (Table.IV). The margin function, $CT_M(3C_C)$, alters to $CT_M(2C_C)$ when $L_{C\,t} = 1$ and gives *CT*-polymorphic *AO21-OA21* gate (circuit response is in panel-6). The next six gates are homogeneous to heterogeneous logic type. For *AND3-AO21* gate, the aggressor weights are w_1=w_2=1, w_3=2 and w_{ct}=2 (note that the weights of inputs are heterogeneous). The margin function for *AND3*, in this case, is $CT_M(4C_C)$. The control aggressor biases it to $CT_M(2C_C)$ and operates the gate as *AO21* (circuit response is in panel-7). In the previous case, if *Ct* is given C_C strength instead of $2C_C$, the margin function manipulates from $CT_M(4C_C)$ to $CT_M(3C_C)$, giving rise to CT-polymorphic *AND3-OA21* gate as shown in Fig.7 pane-8.

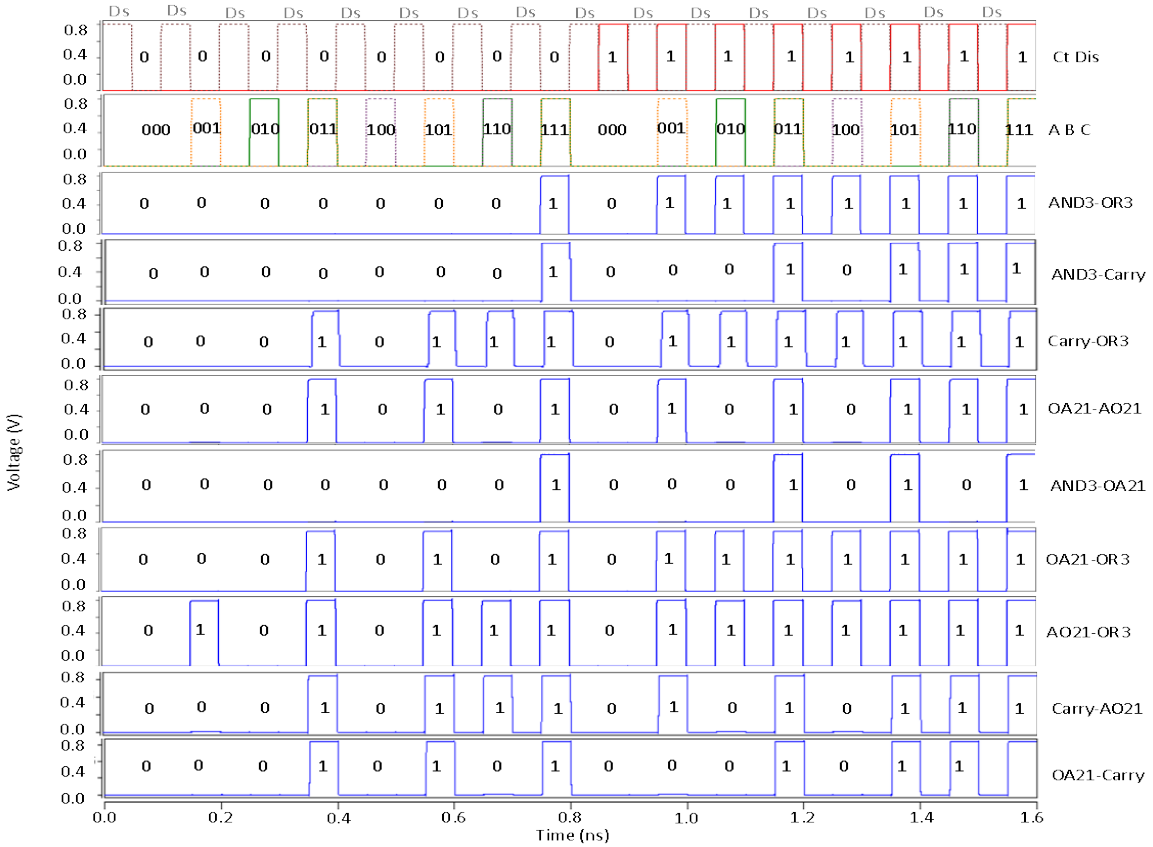

Fig.7 Simulation responses of 3-input CT-Polymorphic logic gates

Similarly, CARRY-OR3, OA21-OR3, AO21-OR3, and OA21-CARRY, and AO21-CARRY results are shown in Fig. 7.

Crosstalk-Polymorphic Cascaded Circuit Example

This section demonstrates cascading polymorphic gates to implement a block/module level polymorphic circuit. Fig.8 is a 2-bit Multiplier-Sorter-Adder circuit. The circuit uses 31 gates in total, out of which 25 are crosstalk gates, and 6 are inverters. 16 out of 25 crosstalk gates are polymorphic gates, which are efficiently employed to switch the circuit between the multiplier, sorter and adder operations. Two control signals, *C1* and *C2*, are given to a control circuitry shown in the inset figure, which generates *C3*-*C5* control signals. *C1*-*C5* signals are employed in the circuit to switch the circuit between three functions. Fig.9 shows the simulation response of the circuit; different operation modes of the circuit are annotated on top, which are, Multiplier (M), Sorter (S), and Adder (A). The first panel in the figure shows *Dis* signal; *Dis*=1 is the discharge state (DS) and *Dis*=0 is the Logic Evaluation (LE) state. The second panel shows the control signals *C1* and *C2*, whose values as 01, 11 and 10 corresponds to the multiplier, sorter, and adder operations, respectively. Third and fourth panels show the 2-bit inputs A[1:0] and B[1:0]. The subsequent four panels show the 4-bit responses of the circuit, Y[3:0]. In order to effectively demonstrate the transformation of the circuit, control signals are given such that the circuit switches alternately between multiplier, sorter, and adder modes, and in each set of these modes, common input values are fed through A1A0 and B1B0. For example, for the first input combinations, 11 and 10, the multiplier operation gives 0110 as output while the succeeding sorter and adder operations give 1110 and 0101 outputs, respectively. Similarly, for the second inputs, 10 and 01, M, S, and A operations give 0010, 1100 and 0011 outputs, respectively. In a similar fashion, few other combinations are shown in the next stages. The circuit consumes only 155 transistors in total.

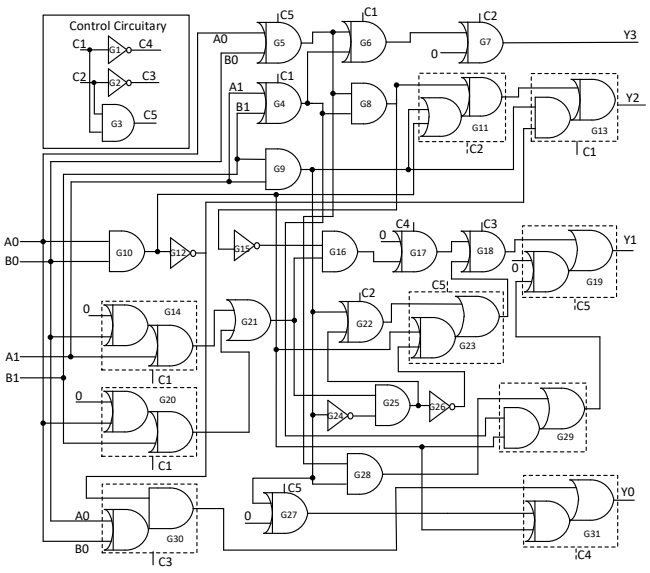

Fig.8 Crosstalk Polymorphic Multiplier-Adder-Sorter circuit

IV. COMPARISON

In this section, we compare the CT-polymorphic logic circuits with respect to existing polymorphic approaches available in the literature and disucuss its advantages and disadvantages (Table V). The reconfigurability in CT-polymorphic circuits is achieved by using the same Crosstalk aggressor-victim technique that actually performs the logic computation, which enables deliberate and very fast reconfiguration of the gates. Despite of its radically different logic and reconfigurability aspects, the working mechanism in crosstalk computing is based on well-known capacitive electrostatics, which makes it easily realizable through existing process setups and fabrication techniques. The complex gates listed for other approaches in the table are constructed by cascading polymorphic NAND-NOR, AND-OR gates

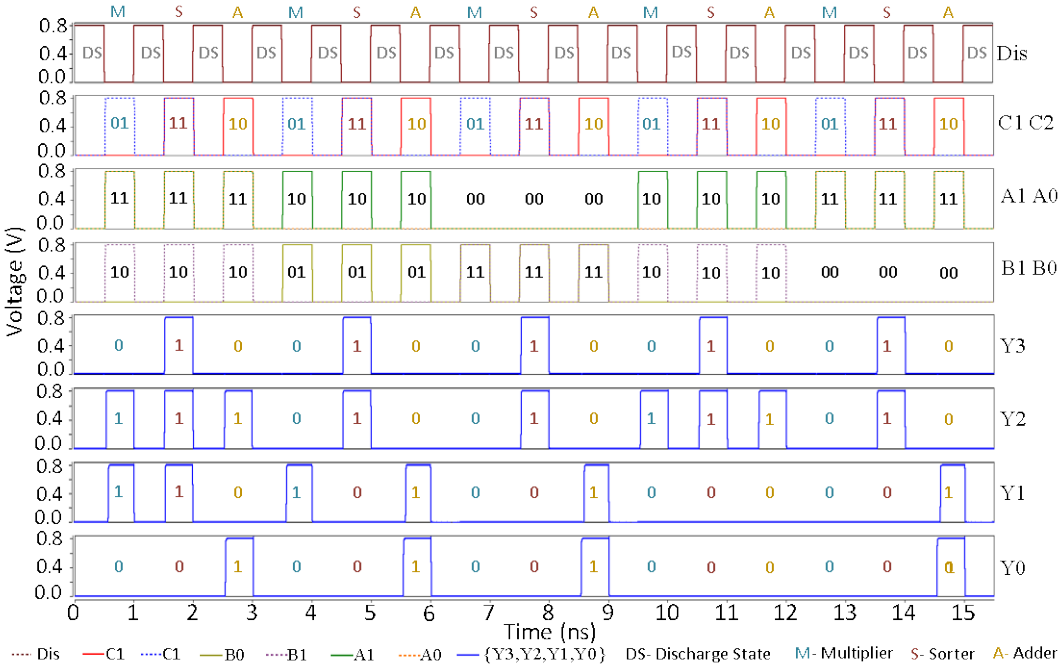

Fig.9 Crosstalk Polymorphic Multiplier/Adder/Sorter circuit simulation response

TABLE V. COMPARISON OF POLYMORPHIC TECHNOLOGIES

Technology	**CMOS**	**Evolved Circuits[3]**			**Ambipolar NWFET[7]**	**Crosstalk-Polymorphic**
Mechanism	Circuit duplication and use of multiplexers to select redundant blocks	A control voltage biases the circuits different operation	Temperature variation effects on devices bias the circuits to different modes	Power supply variation effects on devices biases the circuits to different mode	Band structure of the transistor is altered from p-type to n-type using a control gate	Signal Interference through interconnect crosstalk
Control parameter	Select Signal	Control Voltage	Temperature	Supply Voltage	Control voltage	Control Voltage
Process-Technology Node	16nm (independent)	0.35um (strongly dependent)			30nm (dependent)	16nm (friendly to advanced nodes)
Scalability Dependence	Synthesis	Evolution limitation (Genetic Algorithms)	Evolution limitation (Genetic Algorithms)	Evolution limitation (Genetic Algorithms)	Large scale fabrication of nanowires and reliable ambipolar property	-Crosstalk Coupling network -Noise Margins -Polymorphic Logic Synthesis
Trade-off Vs. Custom ASIC	Density, power and performance penalties for redundant blocks	Power and performance penalties and limited density benefits	Power and performance penalties and limited density benefits	Power and performance penalties and limited density benefits	Limited density benefits	Density, Power and Performance benefits

presented in [1] and [20]. The traditional approach ('CMOS' column in the table) is multiplexer based, where independent stand-alone circuits are designed and selected through a multiplexer. Though this approach is mainstream and can be implemented in any technology node (we have designed in 16nm), it consumes large resources as listed in the table.

To the best of our knowledge, a wide range of compact single-stage and cascaded polymorphic complex logic implementations like in Crosstalk logic were not reported in other approaches. However, the scalability limitations that needs to be overcome in CT-computing are i) ability to achieve the efficient Crosstalk coupling networks, ii) Noise margins of the CMOS inverter that limits the fan-in of the circuits, in turn the ability to construct many single stage/gate complex CT-polymorphic logic circuits (cascaded polymorphic circuits are the solution), and iii) CT-Computing friendly polymorphic logic synthesis algorithms/tools need to be developed for EDA (Electronic Design Automation) flows. In our other works [29][30], we have addressed several of these issues. The evolved circuits discussed in the table V do possess a unique merit that we can construct polymorphic circuits with interesting control parameters such as temperature, supply-voltage, light, radiation, etc.[1] (which are not done for in Crosstalk circuit style). These features make them ideal candidates for sensor-based and adaptable circuit applications. It is to be noted that the CT-circuit presented in this paper are only controlled using a control-voltage. Experimentally, evolution techniques can be also applied to Crosstalk Circuits to explore reconfigurability potentials based on all possible control parameters.

Next, to compare with emerging reconfigurable transistors we have considered ambipolar Si nanowire FET (SiNWFET) by De Marchi *et.al* [20]. In this approach, a nanowire transistor can be configured to either n-type or p-type with a control voltage. Limitations of this approach are [19][20], density benefit is limited, additional circuitry required to swap power rails for pull-up and pull-down networks, non-robust device response, and requirement of new fabrication steps in the existing process flows. Also, compared to other exotic device-based approaches [5], CT-Computing can be achieved through existing fabrication techniques. Thus, it augments the conventional CMOS based device, circuit, and manufacturing paradigms. Finally, the CT-Polymorphic approach consumes fewer transistors than any other transistors based polymorphic circuit approach in the literature. By averaging the transistor count of all the circuits in Table.V, the CT-Circuits consume 64%, 58%, and 40% less transistors compared to CMOS, evolved circuits, and Ambipolar Circuit techniques, respectively.

V. BENCHMARKING

The switching energy and performance for all the crosstalk gates presented above are characterized and benchmarked with their counterpart CMOS implementations (Table VI). The CMOS implementation is multiplexer based, where independent stand-alone circuits are designed and selected through a multiplexer. Both the circuits are implemented and benchmarked using 16nm PTM tri-gate transistor models. The benefits are huge for CT-Computing circuits. As shown in Table.VI, the CT-Polymorphic circuits achieve 2.8x density, ~1.5x performance, and ~2x power benefits. The benefits are primarily due to reduced transistor count and are projected to be higher for large-scale designs. A comparison of CMOS vs Crosstalk circuit can illustrate the source of these benefits. For an example, the AND3-CARRY polymorphic circuit, with its Boolean expression, ABCS'+ S(AB+BC+CA), requires just 5 transistors compared to 30 transistors in CMOS based implementation. Thus, in best case the transistor reduction is 6x. For the polymorphic Multiplier-Sorter-Adder unit, the benefits are 3.4x and 62% in terms of density and power with comparable performance with respect to CMOS at 16nm. It is to be noted that owing to the reduced transistor count the interconnection requirements would also be considerably less at standard-cell level.

VI. CONCLUSION

Crosstalk Logic is a novel and radically different way of doing the logic computation. The paper develops a detailed framework for polymorphic logic circuits in Crosstalk Logic and shows implementation of a wide range of crosstalk polymorphic logic gates. The gates presented are reconfigurable AND-OR, AO21-OA21, AND3-AO21, AND3-OA21, OR3-AO21, OR3-OA21, AND3-CARRY, CARRY-

TABLE VI
BENCHMARKING OF CROSSTALK LOGIC GATES WITH RESPECT TO CMOS

	Switching Energy (aJ)			Performance (ps)		
GATES	CMOS	Cross-talk	%Reduction	CMOS	Cross-talk	% Reduction
NAND2	232.1	122.3	47.31	4.12	4.06	1.43
NOR2	202.7	260.5	-28.5	5.61	5.86	-4.492
AOI21	154.4	207	-34.07	5.73	5.51	3.94
OAI21	229.3	135.2	41.03	4.36	5.17	-18.52
NAND3	347.7	112.5	67.65	4.98	4.18	16.11
Carry	1198.8	326.59	72.75	14.58	8.69	40.4
NAND2-NOR2	796.14	139.03	82.54	13.46	4.32	67.89
NAND3-NOR3	1472.6	172.02	88.32	13.21	5.12	61.22
AOI21-OAI21	698.42	190.52	72.72	9.52	5.39	43.38
NAND3-AOI21	1091.3	641.38	41.23	14.08	14.14	-0.42
NAND3-OAI21	874.99	959.44	-9.65	11.69	19.4	-65.92
NOR3-AOI21	1030.4	661.67	35.78	17.78	12.65	28.86
NOR3-OAI21	938.88	546.89	41.75	18.14	11.47	36.8
CARRY-OR3	4258.6	420.15	90.13	15.02	8.3	44.74
Carry-AND3	3059.9	289.69	90.53	16.77	7.39	55.91
Carry-AO21	2332.9	481.31	79.37	28.92	10.56	63.48
OA21-Carry	2004.2	366.91	81.69	15.67	9.97	36.35
MUL-SORT-ADD	16.2 fJ	6.104 fJ	62.41	61.5	54.4	11.56

OR3, CARRY-AO21, OA21-CARRY and Inv-Buf. Our circuit evaluation and benchmark comparisons show that CT-Polymorphic logic approach is very compact (i.e less device count) and efficient than other polymorphic approaches.

VII. REFERENCES

[1] A. Stoica, R. Zebulum, and D. Keymeulen, "Polymorphic Electronics," *Evolvable Syst. From Biol. to Hardw.,* vol. 2210, pp. 291-302, 2001.

[2] A. Stoica, R. S. Zebulum, D. Keymeulen, and J. Lohn, On polymorphic circuits and their design using evolutionary algorithms, in *Proc. of IASTED International Conference on Applied Informatics AI2002*, Insbruck, Austria, 2002.

[3] Sekanina L. (2015) Principles and Applications of Polymorphic Circuits. In: *Evolvable Hardware. Natural Computing Series*. Springer, Berlin, Heidelberg

[4] Sekanina, L., Stareček, L., Kotásek, Z., Gajda, Z.: Polymorphic Gates in Design and Test of Digital Circuits. *International Journal of Unconventional Computing*, 4(2), 2008, Philadelphia, pp. 125 – 142, ISSN 1548-7199.

[5] S. Rakheja and N. Kani, "Polymorphic spintronic logic gates for hardware security primitives — Device design and performance benchmarking," *2017 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, Newport, RI, 2017, pp. 131-132.

[6] A. Stocia, D. Keymeulen, V. Duong, and C. Salazar-Lazaro, "Automatic synthesis and fault-tolerant experiments on an evolvable hardware platform. In *IEEE Aerospace Conference Proceedings*, volume 5, pages 46547 I, 2000.

[7] R. Ruzicka and V. Simek, "More Complex Polymorphic Circuits: A Way to Implementation of Smart Dependable Systems", *ElectroScope Pilsen*, vol. 7, no. 5, pp. 1-6, 2013, ISSN 1802-4564.

[8] Sekanina, L.: Evolution of Polymorphic SelfChecking Circuits. *Proc. of Evolvable Systems: From Biology to Hardware*, Berlin, Springer, 2007, pp. 186 – 197.

[9] L. Sekanina. Design and Analysis of a New Self-Testing Adder Which Utilizes Polymorphic Gates. In Proc. of the *10th IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop DDECS 2007*, pages 246-246, Krakow, Poland, 2007. IEEE Computer Society.

[10] McDermott, M.W., and Turner, J.E.: 'Configurable NAND/NOR element'. *United States Patent 5,592,107,* January 1997.

[11] Burrows, James L. "Universal logic circuit." *U.S. Patent* 4,558,236, issued December 10, 1985.

[12] R. Ruzicka, "New Polymorphic NAND / XOR Gate 2 Known Polymorphic Gates," pp. 192–196, 2007.

[13] A. Stoica, G. Klimeck, C. Salazar-Lazaro, D. Keymeulen and A. Thakoor, "Evolutionary design of electronic devices and circuits", *Proc. of the 1999 Congress on Evolutionary Computation,* 1999-July-6-9.

[14] A. Stoica and R. Andrei, "Adaptive and Evolvable Hardware - A Multi-faceted Analysis," *Second NASA/ESA Conference on Adaptive Hardware and Systems (AHS),* pp.486-498, 2007.

[15] A. Stoica, T. Arslan, D. Keymeulen, Vu. Duong, R. Zebulum, I. Ferguson and T. Daud, "Evolutionary recovery from radiation induced faults on reconfigurable devices," in proceedings of *aerospace conference*, vol. 4, 6-13 March 2004, IEEE, pp. 2449-2457.

[16] G. W. Greenwood, "On the practicality of using intrinsic reconfiguration for fault recovery," *IEEE Trans. Evolutionary Computation*, vol. 9, no. 4, Aug. 2005, pp. 398-405.

[17] A. Stoica et al., "Taking evolutionary circuit design from experimentation to implementation: some useful techniques and a silicon demonstration," in *IEE Proceedings - Computers and Digital Techniques*, vol. 151, no. 4, pp. 295-300, 18 July 2004.

[18] W. Ditto, A. Miliotis, K. Murali, S. Sinha, and M. Spano, "Chaogates: Morphing logic gates designed to exploit dynamical patterns," *Chaos 20*, 037107 (2010)

[19] W. M. Weber, A. Heinzig, J. Trommer, D. Martin, M. Grube and T. Mikolajick, "Reconfigurable nanowire electronics - A review", *J. on Solid-State Electronics*, vol. 102, pp. 12-24, December 2014.

[20] M. De Marchi et al., "Configurable logic gates using polarity controlled silicon nanowire gate-all-around FETs," *IEEE Electron Device Lett*., vol. 35, no. 8, pp. 880–882, 2014.

[21] J. Zhang, P. E. Gaillardon, and G. De Micheli, "Dual-threshold-voltage configurable circuits with three-independent-gate silicon nanowire FETs," Proc. - *IEEE Int. Symp. Circuits Syst*., pp. 2111–2114, 2013.

[22] Yu, W. J., Kim, U. J., Kang, B. R., Lee, I. H., Lee, E. H., Lee, Y. H.: Multifunctional logic circuit using ambipolar carbon nanotube transistor. Proc. *SPIE 7399*, 739906 (2009).

[23] Paasch, G., Lindner, Th., Rost-Bietsch, C.: Operation and Properties of Ambipolar Organic Field-effect Transistors, In: *Journal of Applied Physics*, Vol. 98, No. 8, 2005, US.

[24] Naveen kumar Macha, et al., "A New Concept for Computing Using Interconnect Crosstalks," *2017 IEEE International Conference on Rebooting Computing (ICRC)*, Washington, DC, USA, December 2017.

[25] Naveen kumar Macha, Sandeep Geedipally, Bhavana Tejaswee Repalle, Md Arif Iqbal, Wafi Danesh, Mostafizur Rahman "Crosstalk based Fine-Grained Reconfiguration Techniques for Polymorphic Circuits," *IEEE/ACM NANOARCH 2018*.

[26] Naveen kumar Macha, Bhavana Tejaswini Repalle, Sandeep Geedipally, Rafael Rios, Mostafizur Rahman "A New Paradigm for Fault-Tolerant Computing with Interconnect Crosstalks," *2018 IEEE International Conference on Rebooting Computing (ICRC),* Washington, DC, USA, December 2018.

[27] N. K. Macha, B. T. Repalle, M. A. Iqbal and M. Rahman, "A New Computing Paradigm Leveraging Interconnect Noise for Digital Electronics Under Extreme Environments," *2019 IEEE Aerospace Conference*, Big Sky, MT, USA, 2019, pp. 1-8, doi: 10.1109/AERO.2019.8741746.

[28] Predictive Technology Model (PTM) Tempe AZ USA Nov. 2011.

[29] Naveen Kumar Macha, Md Arif Iqbal, Bhavana Tejaswini Repalle, Sehtab Hossain, Mostafizur Rahman, Crosstalk Noise based Configurable Computing: A New Paradigm for Digital Electronics, *IEEE INTERNATIONAL SYSTEM-ON-CHIP CONFERENCE 2021* (Under Review)

Arif Iqbal, Naveen Kumar Macha, Bhabhana T Repalle, Mostafizur Rahman, From 180nm to 7nm: Crosstalk Computing Scalability Study, *IEEE Electron Device Society S3S Conference*,2019