Evolving Side-Channel Resistant Reconfigurable Hardware for Elliptic Curve Cryptography

Bikash Poudel*, Sushil J. Louis†, and Arslan Munir‡
Department of Computer Science and Engineering
University of Nevada, Reno
Email: *bpoudel@nevada.unr.edu, †sushil@cse.unr.edu, and ‡arslan@unr.edu

***Abstract*—We propose to use a genetic algorithm to evolve novel reconfigurable hardware to implement elliptic curve cryptographic combinational logic circuits. Elliptic curve cryptography offers high security-level with a short key length making it one of the most popular public-key cryptosystems. Furthermore, there are no known sub-exponential algorithms for solving the elliptic curve discrete logarithm problem. These advantages render elliptic curve cryptography attractive for incorporating in many future cryptographic applications and protocols. However, elliptic curve cryptography has proven to be vulnerable to non-invasive side-channel analysis attacks such as timing, power, visible light, electromagnetic, and acoustic analysis attacks. In this paper, we use a genetic algorithm to address this vulnerability by evolving combinational logic circuits that correctly implement elliptic curve cryptographic hardware that is also resistant to simple timing and power analysis attacks. Using a fitness function composed of multiple objectives – maximizing correctness, minimizing propagation delays and minimizing circuit size, we can generate correct combinational logic circuits resistant to non-invasive, side channel attacks. To the best of our knowledge, this is the first work to evolve a cryptography circuit using a genetic algorithm. We implement evolved circuits in hardware on a Xilinx Kintex-7 FPGA. Results reveal that the evolutionary algorithm can successfully generate correct, and side-channel resistant combinational circuits with negligible propagation delay.**

***Index Terms*—Elliptic curve cryptography, genetic algorithms, reconfigurable hardware design, side-channel attacks**

I. Introduction and Motivation

Elliptic curve cryptography (ECC) is a widely accepted public-key cryptosystem that provides better security-per-bit than other public-key cryptosystems (e.g., RSA) and discrete logarithm cryptosystem (e.g., Elgamal) [1]. This allows ECC to have a shorter key length than RSA which engenders advantages such as reduced computing effort, and less storage requirement. These features make ECC amenable to ubiquitous embedded applications found in personal digital assistants (PDAs), cell phones, smartcards, and many devices associated with the Internet of things (IoT) paradigm. In addition, shorter key lengths reduce computing effort and make ECC faster than other public-key cryptographic approaches for private-key operations such as signature generation or key management [1]. The performance disparity between ECC and other cryptosystems expands dramatically as we move from low to high security-level cryptosystem. RSA is generally reported to be $10\times$ slower than ECC at a 128-bit security-level and $50\times$ to $100\times$ slower at a 256-bit security-level for private key operations [2], [3]. Consequently, many new cryptographic protocols are moving away from RSA towards elliptic curves. This transition is even faster in the embedded domain where ECC's cost/performance benefits quickly become significant due to the inherent resource constraints in embedded systems.

Although, cracking ECC has proved to be a mathematically difficult problem, the advent of cryptanalytic attacks on implementations, also known as side-channel attacks (SCAs), has overturned this traditional concept through the fine-grained analysis of sensitive leakages such as timing, power, visible light, electromagnetic, and acoustic side-channel information [4]. In timing analysis attacks, an adversary obtains private information by carefully measuring the time it takes for the device to carry out cryptographic operations. Such measurements are possible because different cryptographic operations take different computation time and can potentially be used to reveal the encrypted information after careful statistical analysis.

Power analysis attacks are other prevalent forms of SCAs. Power analysis attacks are classified as simple, differential, and refined power analysis attacks. In simple power analysis attacks, an attacker analyzes the power trace in order to guess which particular instruction is being executed at a certain time and what are the values of corresponding inputs and outputs. Therefore, an attacker needs exact knowledge of the implementation to mount simple power analysis attacks. However, in differential and refined power analysis attacks, an adversary uses knowledge of standard implementation techniques and exploits comprehensive statistical methods to analyze the power consumption profile and thus extract secret information. Electromagnetic analysis attacks capitalize on the electromagnetic radiation generated by electronic components while running the cryptographic algorithm to infer secret information. Another, recent SCA uses acoustic analysis of the acoustic emanations from a processor while the processor is running cryptographic procedures to extract the secret information [5].

The non-invasive SCAs on ECC implementations can be performed at low cost with high accuracy [6]. This causes unprotected ECC implementations to leak key-dependent secret information via side-channels. In order to conceal this information from adversaries, a multitude of work has been done at both the algorithmic and hardware implementation levels. At the algorithmic level, randomization of the private exponent by exponent splitting, blinding the base point P

used to generate the public key, and using randomized projective coordinates provide three common countermeasure techniques. [7], [8], [9] incorporate these algorithmic techniques to shield simple and differential power analysis attacks. However, the additional costs incurred in terms of the additional software implementation time and in terms of additional hardware implementation area are some of the potential shortcomings of these methods. In addition, others [10], [11] have proposed designing and using a side-channel resistant co-processor for ECC. However, sequential circuits (or synchronous circuits) based processor architectures proposed in [10], [11] suffer from a number of limiting factors such as increased difficulty of clock distribution, increased clock rates, decreased feature size, increased power consumption, timing closure effort, and difficulty with design reuse. The increased clock rate increases power consumption, noise, and electromagnetic emissions from the circuit. This makes sequential circuits more susceptible to power, noise, and electromagnetic emission analysis attacks. Another major drawback of sequential circuits is that an attacker is able to precisely isolate start times and end times of operations guided by the timing reference generated by the clock to perform the segmentation of power and energy side-channel information which allows an adversary to model the operational behavior of the ECC hardware through rigorous statistical analysis and thus infer the secret information [12].

To address these limitations, we propose a novel genetic algorithm (GA) based approach for designing ECC circuits that are resistant to simple timing and power analysis attacks. In contrary to [10], [11], we design a pure combinational logic circuit for ECC processors. The ECC processor uses three different combinational circuit modules to perform basic elliptic curve point operations – point addition, point doubling, and point (or scalar) multiplication. The primary advantage of using combinational circuits is that unlike sequential circuits, combinational circuits are hard to analyze because they cannot be segregated into multiple operational sub-stages. Furthermore, the effective power analysis attacks based on hamming weight and hamming distance are suitable only for the sequential logic circuits like register files and buses but not for the multi-input combinational logic circuits. The only attack that we found in the literature is power template attack proposed by Zheng et al. [13]. However, this attack could be ineffective for 160-bit input ECC processor because the attacker was successful only for 5-bit input combinational circuit (refer section IV). Furthermore, our non-conventional circuit design approach using GA can impede the attack methodology used in [13].

We used the CHC adaptive search algorithm as our GA [14] to design combinational logic circuit for ECC. To start, we wrote a genetic algorithm to generate $3, 4, 6,$ and 8-bit combinational circuits on the path towards the design of full sized 160-bit ECC circuit. The reasons for using GA to design combinational logic circuits are in two folds. First, the combinational logic circuits designed using GA would be a highly (we hoped) non-conventional circuits which bolster the security of the resulting cryptosystem. Second, GA can design large combinational logic circuits which are infeasible to design manually. Finally, results shows that our GA can generate functionally correct side-channel resistant combinational circuits for point arithmetic in elliptic curve. Our major contributions are:

- We propose a side-channel resistant reconfigurable ECC processor evolved using genetic (evolutionary) algorithm. The genetic algorithm optimizes for minimal execution time and minimal area footprint. To the best of our knowledge, this is the first work that leverages genetic algorithm to evolve a digital combinational circuit for cryptographic applications.
- We implement our proposed reconfigurable ECC processor on a Xilinx Kintex 7 field-programmable gate array (FPGA).
- We quantify the propagation delay and size of the evolved combinational logic circuits.

II. Related Work

Various previous works have studied design of side-channel resistant elliptic curve cryptosystem. A great deal of work has been done both in algorithm level and hardware level to design a secure side-channel resistant ECC. [7], [8], [9] proposed algorithm level methods to defend against side-channel attacks. These methods comprised of one or combination of following techniques: 1) key blinding technique which involved randomizing base point (used to generate public key) by using elliptic curve isomorphism or field isomorphism, 2) private key randomization by key splitting technique, 3) using randomized projective coordinate to represent the elliptic curve, and 4) using some form of double-and-add-always algorithm for scalar multiplication. However, the potential shortcomings of these methods is that they incur additional costs in terms of time and instruction count for software implementations and in terms of area for hardware implementations.

Moreover, in the hardware based solution spectrum, Liao et al. [10] proposed a non-invasive side-channel attack resistant coprocessor for ECC which provided side-channel resistance with low area-time-product overhead. The basic countermeasures used in the proposed processor relied on the underlying finite field arithmetic in randomized Montgomery domain, which could blind the intermediate value in the iterations of scalar multiplication to prevent the adversaries from cracking the private key by statistical methods. In addition, the author also optimized the modular division and multiplication algorithms to fix the operating times to resist some timing attacks. Furthermore, Lee et al. [11] proposed a power-analysis-resistant dual-field ECC processor using heterogeneous dual-processing-element architecture. The authors leveraged priority-oriented scheduling of right-to-left double-and-add-always elliptic curve scalar multiplication with randomized processing technique to achieve a power-analysis resistant dual-field ECC processor. Thus, [10] and [11] are hardware based solutions where authors designed a sequential circuit to perform ECC arithmetic such as scalar multiplication, point addition, etc. One major limitation of memory-

based sequential circuits is that it can be divided into a number of operational sub-stages that allows an adversary to perform some form of rigorous statistical analysis to model the operational behavior of the ECC hardware.

III. Elliptic Curve Cryptography

In this section, we first briefly introduce the fundamentals of ECC. Then, we elaborate on the basic point arithmetic algorithms that are necessary to form a fitness function for our evolutionary algorithm (refer section IV).

ECC [15] is based on the algebraic structure of elliptic curves over the finite fields. For our work, we use the elliptic curve over the prime finite field $\mathcal{Z}_\mathcal{P}$ where prime number $\mathcal{P} = 29$. Eq. 1 shows the elliptic curve we employ for our ECC (refer algorithm 2). The coefficients *a* and *b* are set to *4* and *20*, respectively, as an design example. For our study, we have used this simple elliptic curve. However, for real world implementation, we can use more secure elliptic curves [16].

$$\mathcal{E}(x, y) : y^2 = x^3 + ax + b \tag{1}$$

Algorithm 2 shows double-and-add algorithm for the most important operation (i.e., multiplication) in ECC, which is also used in private and public key-pair generation. Point multiplication (refer algorithm 1) is carried out between the base point on curve and a scalar value (the secret key or private key) to generate the public key which is also a point on the elliptic curve. The inability to compute the secret scalar value given the base point and public key is the elliptic curve discrete logarithm problem. The security of elliptic curve based protocols relies on elliptic curve discrete logarithm problem.

Point addition and point doubling are two other basic arithmetic operations involved in elliptic curve cryptography. Consider $M = (x_1, y_1) \in \mathcal{E}[\mathcal{Z}_\mathcal{P}]$ and $N = (x_2, y_2) \in \mathcal{E}[\mathcal{Z}_\mathcal{P}]$, where $M \neq N$. Then, point addition of M and N is denoted by $M + N = (x_3, y_3)$ and is given by,

$$(x_3, y_3) = (\lambda^2 - x_1 - x_2, \lambda(x_1 - x_3) - y_1) \tag{2}$$

where, $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$. Similarly, for point $M = (x_1, y_1) \in \mathcal{E}[\mathcal{Z}_\mathcal{P}]$ such that $M \neq -M$, point doubling of M is denoted by $2M = (x_3, y_3)$ and is given by,

$$(x_3, y_3) = (\Lambda^2 - 2x_1, \Lambda(x_1 - x_3) - y_1 \tag{3}$$

where, $\Lambda = \frac{3x_1^2 + a}{2y_1}$. Finally, point multiplication between two points $M = (x_1, y_1) \in \mathcal{E}[\mathcal{Z}_\mathcal{P}]$ and $N = (x_2, y_2) \in \mathcal{E}[\mathcal{Z}_\mathcal{P}]$, where $M \neq N$ is denoted by $M * N = (x_3, y_3)$ and is computed by using Double-and-Add algorithm as shown in algorithm 1.

Algorithm 1 Double-and-Add algorithm for point multiplication in ECC.

Input: Elliptic curve $\mathcal{E}[\mathcal{Z}_\mathcal{P}]$, an elliptic curve point N, and a scalar k of k_i bits.
Output: $M = kN$
t = number of bits of k
$\mathcal{P}$ *= prime number*
Initialization:
$M \leftarrow N$
Core Algorithm:
for $i = t - 1$ **downto** 0 **do**
$M \leftarrow (M + M)\, mod\, \mathcal{P}$
if $k_i = 1$ **then**
$M \leftarrow (M + N)\, mod\, \mathcal{P}$
end if
return (M)
end for

Algorithm 2 ECC key generation algorithm.

Input: ECC Parameters: Elliptic curve $\mathcal{E}[\mathcal{Z}_\mathcal{P}]$, an elliptic curve point G, prime number $\mathcal{P}$, private key k_{pr}, public key k_{pub}
Output: private and public key pair
Core Algorithm:
$k_{pr} \leftarrow$ *random number in [1,* $\mathcal{P}$*]*
$k_{pub} \leftarrow k_{pr} \cdot \mathrm{G}$

IV. Genetic Algorithm for Combinational Logic Circuit Design

A GA can be used as an engine to discover new designs of digital circuits because it allows one to explore a much larger space of possible designs [17], [18], [19], [20]. The prime advantage of employing GA is that the designs that are generated by GA are often radically different from those produced by top-down, human, rule-based approaches. Most of the time, evolved circuits are found to be more efficient than those created using traditional design methods [19]. We use a GA to generate combinational logic circuits that perform point operations (e.g., point addition, point doubling, and point multiplication) in elliptic curve.

In combinational logic circuits (sometimes also referred to as time-independent logic circuits), outputs are pure function of present inputs only and do not need memory of any sort to perform their internal operations. So, an attacker does not have memory (e.g., registers) as an attack surface to exploit in order to analyze the circuit under operation. Furthermore, unlike sequential circuits, combinational circuits do not use clock and does not have distinct multiple stages of operation. Therefore, for all possible inputs, all of the components of a combinational circuit (gates and wires) are functioning and contributing towards the propagation delay and power consumption of the whole circuit. Consequently, simple timing and power analysis attacks are usually not effective to learn the operational behavior of a combinational circuit.

Zheng et al. [13] proposed a power template match attack on combinational circuits. In order to mount the attack, first, the authors build power model template of a combinational circuit. The power model template is developed based on the input transitions of combinational logic circuit. Using this power model template, they estimate the average power consumption of the modeled combinational circuit. Then, they

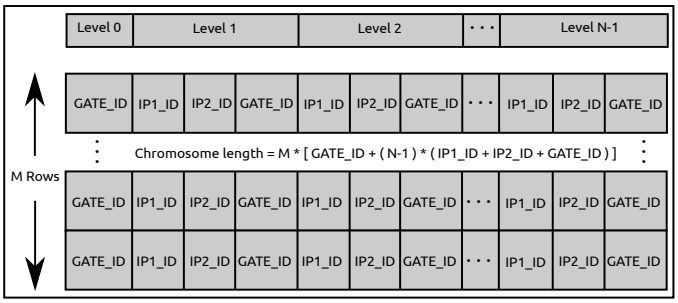

Fig. 1: *Chromosome representing a combinational circuit for point arithmetic.*

implement the combinational circuit in hardware and measure the actual power consumption. By computing the correlation of the average power consumption from power model template and actual power consumption, they are able to recover the secret key. This attack works on combinational circuits designed using the standard-cell libraries based on forward application-specific integrated circuit (ASIC) design flow with synchronous design style. It is an alarming work provided that most of the cryptographic circuit designers use the same process and same electronic design automation (EDA) tools as the authors have used. On that account, it is obvious that security risks exist in the standard-cell based design flow that has no special consideration of protection for combinational circuit design.

However, our combinational circuit design approach is secure and robust against such attack because of three reasons. First, we use a GA to evolve non-conventional combinational circuits. So, it will be difficult for an attacker to design the same circuit to mount the power template attack. Moreover, GA generates multiple circuits which are functionally same. Thus, we can dynamically switch from one circuit to another to augment the security of our combinational circuit (refer section V). Second, Zheng et al. [13] was successful to mount an attack on S-box of PRINTcipher which has 5-bit input and 3-bit output. The small input length (5-bit) made the design of power model template feasible because the circuit has only $2^5 \times 2^5 = 1024$ possible input transitions. However, our elliptic curve cryptosystem will have 160-bit input which will have $2^{160} \times 2^{160}$ possible input transitions. It is apparent that it will be infeasible to take into account all of these input transitions to build an effective power model template to mount template match attack. Third, to augment the resistance of our evolved circuit against SCAs, we can convert the combinational circuit generated by GA into multi-threshold dual-spacer dual-rail logic (MTD^3L) asynchronous circuit which has much smaller side-channel leakages [21].

In this section, we first delineate the encoding of combinational circuit in chromosome for our GA. Next, we elaborate on the fitness function used in our GA.

Encoding a Digital Circuit as a Binary Chromosome: In GA, a solution is represented by chromosome and fitness value associated with the chromosome. Usually, chromosome is a string of binary values 0's and 1's. For our digital circuit design problem, the solution is a combinational circuit. In our work, encoding of a digital combinational circuit into genotype is done by using 2D binary chromosome (refer Fig. 1). The chromosome has N vertical levels where each level has M gates. We have used eight different types of gates, which are shown in Table II. In that account, three bits are used to represent a gate in binary (e.g. GATE_ID or g in Fig. 1). Level 0 gates act as input interface that take input from external sources. Level 1 to level N-1 forms the functional circuit for target algorithm (e.g., point addition, point multiplication). Outputs of gates at level N-1 produce output values.

TABLE I: Parameters for our CHC GA

Parameter Name	Symbol	Value
Crossover Rate	$\mathcal{P}_{cross}$	0.9
Chromosome Length	$\mathcal{L}_{chrom}$	1020, 1680, 2400, 3960
Crossover Threshold	$\mathcal{X}_{th}$	0.2 $\mathcal{L}_{chrom}$
Population Size	Ψ	300
Generations	Θ	800
Elite Density	ρ	0.25 Ψ
Randomization Coefficient	γ	0.35 $\mathcal{L}_{chrom}$

We used two types of gates to build our combinational circuit: two-input one-output gates (e.g., AND, OR) and one-input one-output gates (e.g., NOT, wire). Fig. 1 shows the 2D structure of the chromosome used in our GA. The output of the gates at each level are indexed by numbers from 0 to $(M-1)$. These outputs are connected to inputs of the gates of next level. Hence, the inputs of the gates at a level are also indexed by a number from 0 to $(M-1)$. Thus, $\log_2(M)$ bits are needed to encode an index into binary. The input index IP1_ID (or i) and IP2_ID (or j) are represented by $\log_2(M)$ bits binary value. Therefore, the length of chromosome is given by $M*(g+N*(i+j))$. The length of the chromosomes that we used for our GA are shown in Table I.

We wrote GA to design four different types of combinational logic circuits. These circuits differ in the size of input point which could be $3, 4, 6,$ or 8-bit. We generated four combinational logic circuits having 10×10, 10×16, 20×10, and 20×16 circuit configurations. A 10×16 circuit has 10 levels (from 0 to 9) and each level has 16 gates. The same rule applies to the other three circuit configurations. A 10×16 circuit works on 4-bit input values. 10×10, 20×10, and 20×16 circuits work on $3, 6,$ and 8-bit input values, respectively. The input values are the points in the elliptic curve over prime field $\mathcal{Z}_{\mathcal{P}}$.

Fitness Function: A primary operation involved in a GA is the evaluation of adherence of evolved solutions to the imposed constraints. GA uses a fitness function to evaluate the competence of evolved solutions. The fitness function for our GA is based on *aggregation by variable objective weighting* [22] which uses the weighted-sum method for fitness assignment. Thereby, each objective is assigned a weight $\beta_i \in (0, 1)$ such that $\sum \beta_i = 1$, $i \in \{1, 2, \cdots, m\}$ where m is the number of objectives. The scalar fitness value is calculated by summing up the weighted objective values $\beta_i \cdot f_i(x)$ which is equal to 1. In our case, there are three governing

TABLE II: The gate equivalent and propagation delay of standard logic gates.

Gate	Gate Equivalent	Delay (ns)
NOT	1	0.0625
AND	2	0.209
OR	2	0.216
XOR	3	0.212
NAND	1	0.13
NOR	1	0.156
XNOR	3	0.211
WIRE	1	0.02

constraints (or objectives) viz., correctness in input/output behavior, minimization of propagation delay, and minimization of size of the evolved circuit. β for input/output behavior (β_1) is set to 0.5 and β for circuit size and propagation delay (β_2 and β_3) are set to 0.25 and 0.25, respectively.

Our first objective is the correctness in input/output behavior. In order to quantify the correctness in input/output behavior, we incorporate the notion of reward function which depends on the expected output and observed output of the evolved circuit. A reward function $\mathcal{R}(\mathcal{O}_i^{exp}, \mathcal{O}_i^{obs})$ is defined which counts the number of observed outputs that are equal to the expected outputs. This count is considered as a reward value. $\mathcal{O}_i^{exp}$ represents expected output of the circuit with $\mathcal{I}_i$ as input and $\mathcal{O}_i^{obs}$ represents the observed output of evolved combinational circuit. The expected output is computed by implementing point arithmetic algorithms (refer section III). $\mathcal{I}_i$, where, $i \in \{1, 2, ..., |\mathcal{I}|\}$, represents the simulation inputs which are the points on elliptic curve (refer Fig. 2). We have used these simulation points to check the input/output behavior of evolved combinational circuit. For point addition and doubling, we used these points on the elliptic curve as the inputs and the result is also a point in the curve. For, scalar multiplication, a scalar value and a point from the curve as a base point is taken as input and the output is a point in the elliptic curve. As depicted in Fig. 2, the points in the elliptic curve are divided into four rectangles. The 3-bit evolved circuit can perform all point operations for all of the points in rectangle A. 4-bit, 6-bit and 8-bit evolved circuits work for points in rectangle B, C, and D, respectively. We were able to generate a combinational circuit that can perform all point operations for all of the points in elliptic curve over prime field $\mathcal{Z}_\mathcal{P}$.

Our second design objective is the minimization of the size of evolved circuit. We estimate the necessary area for an evolved circuit using the concept of *gate equivalent* [23] which is a basic unit of measure for digital circuit complexity. This measure is more accurate than the simple number of gates concept. We formulate a function, $\mathcal{G}_{eqv}(g)$, to represent the gate equivalent value of an evolved circuit where $g \in G^{chrom}$ is an evolved combinational circuit. G^{chrom} represents the population of all evolved circuits by the GA. Finally, our third objective is the minimization of the propagation delay of evolved circuit. The finite time that a circuit takes to

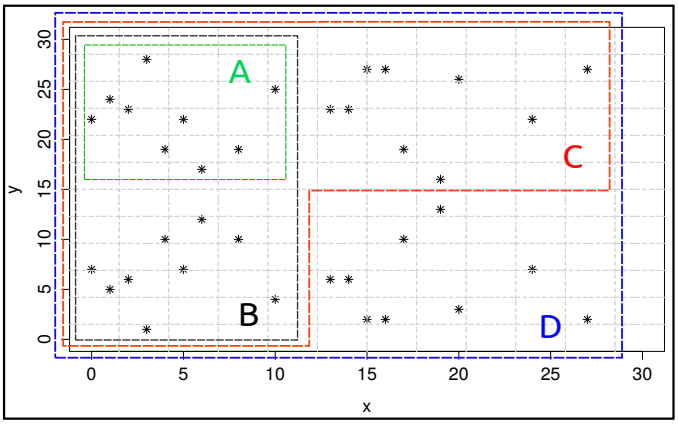

Fig. 2: *Points in elliptic curve over the prime field* $\mathcal{E}[\mathcal{Z}_{29}]$.

reflect the change in input in its output values is known as *propagation delay*. Propagation delay is different for different gates. We measure the propagation delay using the path having highest delay called the worst-case delay path (or critical path). $\mathcal{D}(g)$ represents the delay function in our fitness function. To compute the critical delay path, we model the evolved combinational circuit as a directed acyclic graph. The gate equivalent values and propagation delay values that we used for the gates in our evolved circuit are shown in Table II. The following equation shows the fitness function which is used to compute fitness of a chromosome. Here, $\mathcal{F}^{chrom}$ represents the fitness of a chromosome, $\mathcal{G}^{chrom}$ represents set of evolved circuits, and $\mathcal{L}_{chrom}$ represents the length of the chromosome.

$$\mathcal{F}^{chrom} = \beta_1 \sum_{i=1}^{|\mathcal{I}|} \mathcal{R}(\mathcal{O}_i^{exp}, \mathcal{O}_i^{obs}) + \beta_2 \cdot \frac{1}{\sum_{g \in G^{chrom}} \mathcal{G}_{eqv}(g)} + \beta_3 \cdot \frac{1}{\sum_{l \in \mathcal{L}_{chrom}, g \in G^{chrom}} \mathcal{D}(g)}$$

Genetic Algorithm: We leveraged CHC adaptive search algorithm as our GA with the parameters as listed in Table I. CHC is an elitist algorithm that uses high value of *probability of crossover* ($\mathcal{P}_{cross} = 0.9$) and no mutation. In the following, we describe the working of our version of CHC GA. Initially, a set of random individuals are taken as a starting combinational circuit. These individuals are represented by a data structure having chromosome and fitness of chromosome as components. So, basically an individual is a chromosome with certain fitness value. The initial set of individuals are the parent population which is represented by $\mathcal{G}^p$. The parent population size is set to 300.

The GA solution proceeds by generating a child population ($\mathcal{G}^c$) from a parent population by using reproduction operator called *crossover operator*. During crossover, GA selects two random individuals from $\mathcal{G}^p$. These two individuals can perform crossover only if the hamming distance between these two is greater than or equal to *crossover threshold* ($\mathcal{X}_{th}$). This is known as incest prevention in CHC GA. If these individuals

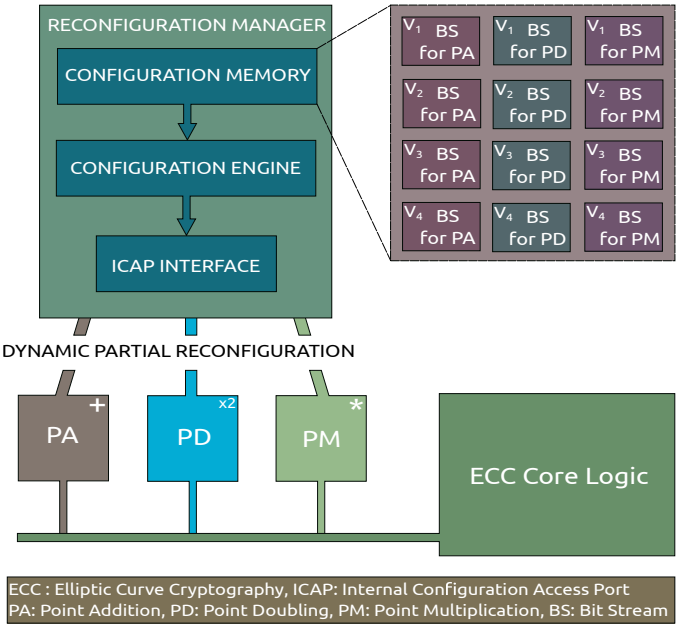

Fig. 3: *Reconfigurable architecture of elliptic curve cryptographic hardware core implemented in Kintex-7 FPGA.*

comply with $\mathcal{X}_{th}$ requirement, then, half of the bits (from random chromosome locations) that are different between two parents are exchanged. This type of crossover is called half-uniform crossover. However, if $\mathcal{X}_{th}$ is not fulfilled then $\mathcal{X}_{th}$ is decreased by 1 and another two random individuals are selected for crossover. When the decreasing $\mathcal{X}_{th}$ hits zero value, the CHC GA is restarted. The initial population for the restarted GA will have ρ elite individuals (individuals with best fitness value) from previous run of GA. The remaining $\Psi - \rho$ individuals are generated by randomly flipping γ bits of the ρ elite individuals.

After the crossover operation, child population is generated. In order to generate individuals for next generation, individuals in both the child and parent population are merged into single pool and are sorted based on highest fitness value. Ψ individuals from the sorted population with best fitness values are selected as next generation parent population.

V. Dynamic Partial Reconfiguration based Reconfigurable ECC Hardware Design

We implement the ECC hardware core in Xilinx Kintex-7 FPGA. The internal architecture of ECC hardware core is shown in Fig. 3. The ECC core is a pure combinational logic circuit. The basic components of ECC core are combinational logic circuit modules for point addition, point doubling, and point multiplication. These circuits are generated by our GA. The ECC core logic is control unit of ECC hardware core. The control logic is implemented based on finite state machine (FSM) design principle.

The novel feature of our ECC hardware core is that the point addition, point doubling, and point multiplication circuits can be changed dynamically during system run-time as our GA can generate multiple functionally correct circuits (represented by V_1 to V_4 in Fig.3) for each of these operations. Therefore, we can have multiple circuits (each of them are different from another) for same point operation. The switching of one circuit to another equivalent circuit is accomplished by using dynamic partial reconfiguration (DPR) feature of Kintex-7 FPGA [24]. The DPR of combinational circuits is controlled by a reconfiguration manager. The reconfiguration manager has a configuration memory that stores the bit-stream (or circuit netlist) of all functionally equivalent combinational circuits for a particular point operation.

Inside the reconfiguration manager, the configuration engine is the main control unit. It reads a configuration of a combinational circuit from the configuration memory and uses internal configuration access port (ICAP) interface to realize the combinational circuit in the FPGA logic fabric. The configuration engine takes signal from ECC core logic in order to decide which configuration from the configuration memory to use at a particular time.

VI. Results

A. Experimental Setup

Implementation of Genetic Algorithm: The genetic algorithm code is written in C programming language. The code is implemented on *Intel CORE i7* processor running *Ubuntu 14.04 LTS* at *3.60 GHz*.

Implementation of ECC Hardware Core: The hardware core for ECC is implemented in Xilinx Kintex-7 FPGA KC705 Evaluation Kit. The implemented hardware modules include evolved combinational circuits, controller circuits, and reconfiguration manager circuit, and are coded in very high speed integrated-circuit hardware descriptive language (VHDL). The execution time and power consumption of the evolved circuits and the ECC hardware core are obtained using the Xilinx ISE 14.7 [25] tool.

B. Evaluation Results

Evolved Combinational Circuit: Fig. 4 shows the 10×10 combinational logic circuit evolved using our GA. This combinational circuit performs point addition of two 3-bit points in the elliptic curve. The index of the output of each gate is a number from 0 to 9. The input ports of the gates on leftmost side are connected to external inputs which are two 3-bit (x,y) coordinate points in elliptic curve. The inputs to gates in the inner levels are from the output of the gates in the preceding level. As discussed in section IV, the combinational circuit has N levels with M gates in each level. The outputs of M gates in level 0 are connected to the inputs of M gates in level 1 and so on. The output of the circuit is taken from the rightmost level of M gates. The output is 3-bit (x,y) coordinate of a point in the elliptic curve which is the sum of two input points.

Fig. 5 (a) depicts the trends of the three objectives of the fitness function with respect to number of evaluations for our GA to generate 10×10 combinational logic circuit. The first objective is to maximize the correctness in input/output behavior. Second and third objectives are to minimize the propagation delay and circuit size. As depicted in Fig. 5 (a), the correctness value increases with number of evaluations and

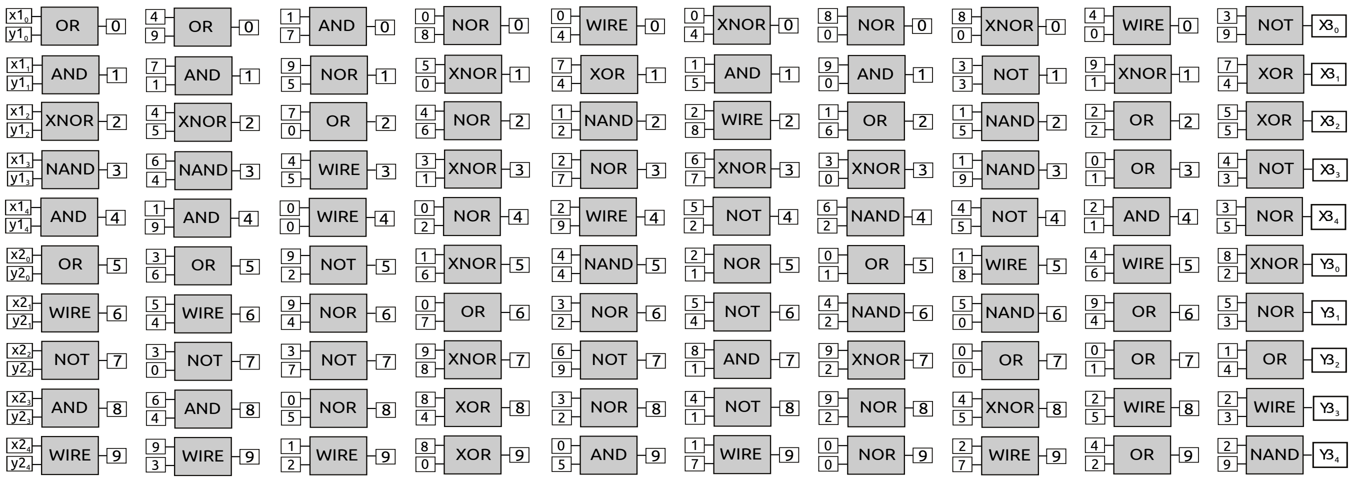

Fig. 4: *Evolved* 10×10 *combinational circuit for point addition.*

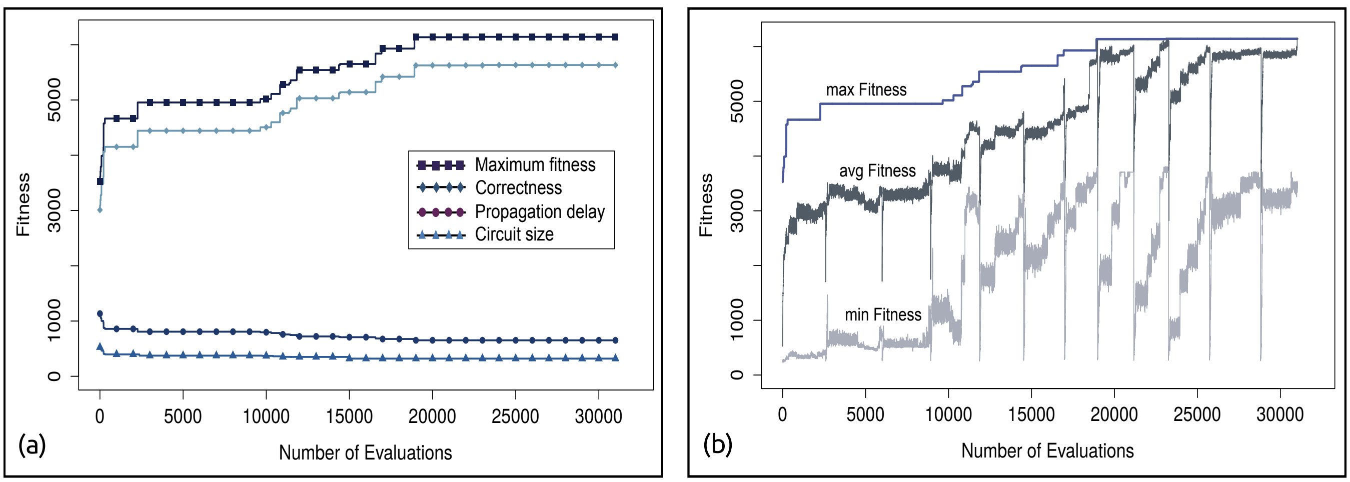

Fig. 5: *(a) Correctness, propagation delay, circuit size, and maximum fitness value of evolved* 10×10 *combinational circuit as a function of number of evaluations. (b) The maximum, average, and minimum fitness value achieved by our genetic algorithm for* 10×10 *circuit configuration.*

reaches a maximum attainable value at $23,119$ evaluations. Furthermore, the propagation delay and circuit size values in the fitness function decrease with the number of evaluations. However, when the number of evaluations crosses $15,000$, the value of propagation delay and circuit size stop decreasing. This is because the critical path (the longest delay path from input to output) in the circuit stops changing and all of the gates in the evolved combinational logic circuit are connected into the critical path.

Fig. 5 (b) depicts the averages of maximum, average, and minimum fitness values over 30 runs for our GA to generate 10×10 combinational logic circuit. The maximum fitness value increases steadily with the number of evaluations. However, the average and minimum fitness value curves can be divided into a number of segments separated by abrupt high-to-low-to-high transitions. These transitions are because of multiple GA restarts, a property of the CHC GA. When the GA (re)starts, the average fitness increases towards the current maximum fitness value. As the average fitness reaches the current maximum fitness, the GA stops making progress. This is marked by a decrease in the value of crossover threshold ($\mathcal{X}_{th}$) in our GA. When the decreasing crossover threshold $\mathcal{X}_{th}$ hits zero, the GA is restarted thus resulting in the high-to-low-to-high transitions in the value of average and minimum fitness. At each segment, average fitness increases towards the current maximum fitness value as shown in Fig. 5 (b).

We have written four GA versions with different parameter settings to generate combinational circuits for $3, 4, 6$, and 8-bit ECC. As shown in Table III, we have used 10×10, 10×16, 20×10, and 20×16 circuit configurations (or chromosome configurations) for $3, 4, 6$, and 8-bit ECC, respectively. Different circuit configurations have different chromosome lengths and different maximum achievable fitness values as shown in Table III. Results reveal that as the number of bits required to represent the points in the elliptic curve increases by one bit, the length of chromosome required to represent the combinational logic circuit increases by $1.06\times$. Hence, to design a combinational circuit for large set of points

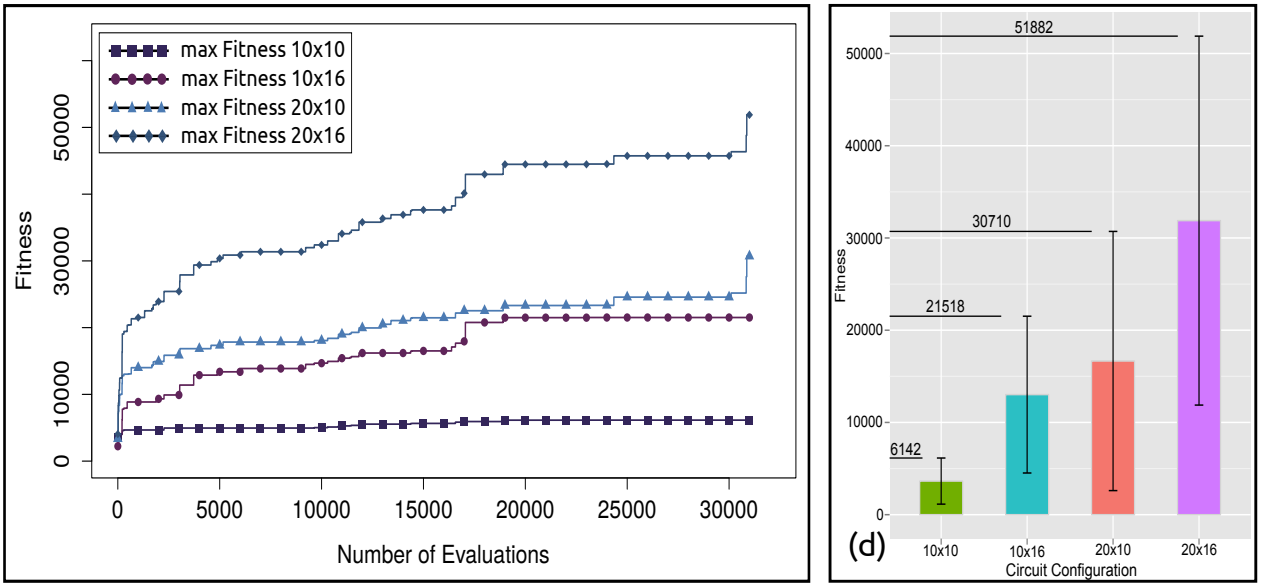

Fig. 6: *(a) The maximum fitness achieved by our GA for four circuit configurations. (b) The fitness values attained by our GA for four circuit configurations.*

TABLE III: The circuit configuration, chromosome length and maximum achievable fitness value depends on the bit-length required to represent points in elliptic curve.

Bit-length of point	Circuit Configuration	Chromosome Length	Maximum Fitness
3-bit	10×10	1020-bit	6142
4-bit	10×16	1680-bit	21518
6-bit	20×10	2400-bit	30710
8-bit	20×16	3960-bit	51882

in elliptic curve, we need much larger chromosome length. Consequently, the GA will run slow and will take much longer time to generate a useful circuit. In order to solve this problem, we have proposed to divide the points in the elliptic curve into a number of sub-groups having a certain fixed number of points as shown in Fig. 2. Then, for each sub-group, we can design a combinational logic circuit. As discussed in section V, we can use DPR feature of FPGA to switch from one combinational logic circuit to another to operate on different points on elliptic curve available in different sub-groups.

Fig. 6 (a) pictures the curve for maximum fitness with respect to number of evaluations for four different circuit configurations. The plot shows that 10×10 and 10×16 circuit configurations reach their maximum attainable fitness values at $18,121$ and $20,156$ evaluations, respectively. However, 20×10 and 20×16 circuit configurations attain their maximum achievable fitness at $31,256$ and $31,589$ evaluations, respectively, which is, on average, $1.64\times$ slower than 10×10 and 10×16 circuit configurations convergence on maximum achievable fitness. These results indicate that generating a larger combinational circuit takes significantly longer time. Although, smaller circuit converges faster than larger circuits, the rate of progress in the value of maximum fitness is same for all circuit configurations, which assures that any circuit configuration can be realized by our GA.

Fig. 6 (b) illustrates the maximum, average, and minimum fitness values obtained by our GA for four circuit configurations. The bar graphs represent the average fitness value of the GA and the lower and upper ends of the range-bars represent the minimum and maximum fitness values. The fitness values are averaged over 30 runs.

Propagation Delay and Side-Channel Analysis: Table IV shows the values of propagation delay and circuit size of the evolved 10×10 combinational logic circuit. We estimated the size and the propagation delay of the evolved circuit using the concept of *gate equivalent* [23] which is a basic unit of measure for digital circuit complexity. The propagation delay of the evolved circuits for point addition, and point multiplication differs by 2.61%. For point addition and point doubling, the difference in propagation delay is 0.25%. The difference in propagation delay of these circuits is low enough, and hence, simple timing analysis cannot reveal whether the current elliptic curve point operation is addition, doubling, or multiplication. Furthermore, the size of the evolved circuit for point addition and point multiplication differs by 2.53%. The difference in the size of point addition and point doubling is 1.71%. Moreover, all of the gates in the 10×10 circuit for point addition, point doubling, or point multiplication will always be in the ON state during the operation. Hence, 10×10 circuits for point addition, point doubling, and point multiplication consumes comparable power and so cannot be distinguished using simple power analysis attack.

Finally, the security strength of our ECC circuits relies on the way we design these circuits using GA. The advantage of using a GA is that it produces non-conventional combinational

TABLE IV: The propagation delay and size of the evolved 10×10 circuit. (refer Table II)

Point Arithmetic	Propagation Delay (ns)	Circuit Size[a]
Point Addition	1.569	237
Point Multiplication	1.610	243
Point Doubling	1.565	233

[a]The unit of circuit size is *gate equivalent*.

logic circuits which are different from those designed using standard cell-based integrated circuit design flow. As discussed in section IV, power analysis attacks, such as, differential power analysis and correlation power analysis are based on hamming weight and hamming distance [13]. These attacks are suitable for power analysis attacks on sequential logic circuits (e.g., clocked registers and buses). However, neither of these attacks are suitable for multi-input combinational logic circuits. The only attack tailored for combinational logic circuits is a power template attack proposed by [13]. In order to mount this attack, the attacker needs to construct a power template according to input transitions of the combinational logic circuit under attack. Zheng et al. [13] were successfully able to attack 5-bit input and 3-bit output S-box of PRINTcipher (described earlier) which has only $2^5 \times 2^5 = 1024$ possible input transitions. However, our elliptic curve cryptosystem will have a 160-bit input which has $2^{160} \times 2^{160}$ possible input transitions. Hence, it is currently infeasible to account all of these input transitions in building an effective power template to mount template match attack for 160-bit combinational logic circuit for ECC.

VII. Conclusion

In this paper, we have used a genetic algorithm (GA) to design a combinational logic circuit for elliptic curve cryptography (ECC). To start, we have used GA to design $3, 4, 6$ and 8-bit ECC circuits on the path towards realizing full-sized 160-bit ECC circuits. The GA reliably designs combinational logic circuits for $3, 4, 6$, and 8-bit point operations in elliptic curve over the prime field $\mathcal{Z}_\mathcal{P}$. In addition, we quantify the propagation delays and sizes of the evolved circuits and show the circuits designed by the GA are resistant to simple timing and power analysis side-channel attacks. These results indicate viability of the GA for designing side-channel resistant functionally correct combinational logic circuits for ECC.

Acknowledgments

This work was supported by the National Science Foundation (NSF) (NSF-CRII-CPS-1564801). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

References

[1] K. Lauter, "The advantages of elliptic curve cryptography for wireless security," *IEEE Wireless Communications*, vol. 11, no. 1, Feb 2004.

[2] K. Maletsky, "RSA vs ECC comparison for embedded systems," Security ICs, Atmel, Tech. Rep., 2015.

[3] Crypto++. (2017, Jan) Security level. [Online]. Available: https://www.cryptopp.com/wiki/Security_Level

[4] S. Skorobogatov. (2011, May) Side-channel attacks: new directions and horizons. [Online]. Available: https://www.cl.cam.ac.uk/~sps32/ECRYPT2011_2.pdf

[5] M. A. Al Faruque, S. R. Chhetri, A. Canedo, and J. Wan, "Acoustic side-channel attacks on additive manufacturing systems," in *Proceedings of the 7th International Conference on Cyber-Physical Systems ICCPC 2016*, Vienna, Austria, Apr 2016, pp. 19:1 – 19:10.

[6] Y. Zhou and D. Feng. (2017, Jan) Side-channel attacks: Ten years after its publication and the impacts on cryptographic module security testing. [Online]. Available: http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-3/physec/papers/physecpaper19.pdf

[7] H. Meshgi, M. Khazaee, B. Kasiri, and H. Shahhoseini, "An efficient algorithm resistant to SPA and DPA variants in ECC," in *2008 1st IFIP Wireless Days*, Dubai, UAE, Nov 2008, pp. 1–5.

[8] J. Lee, Y. Chen, T. C.Y., H. C. Chang, and C. Y. Lee, "A 521-bit dual-field elliptic curve cryptographic processor with power analysis resistance," in *2010 Proceedings of ESSCIRC*, Sevilla, Spain, Sept 2010, pp. 206–209.

[9] M. Joye and C. Tymen, "Protections against differential analysis for elliptic curve cryptography," in *Proceedings of the Third International Workshop on Cryptographic Hardware and Embedded Systems CHES 2001*, London, UK, May 2001, pp. 377–390.

[10] K. Liao, X. Cui, N. Liao, T. Wang, D. Yu, and X. Cui, *High-Performance Noninvasive Side-Channel Attack Resistant ECC Coprocessor for GF(2^m)*. IEEE, 2016, pp. 727–738.

[11] J. W. Lee, S. C. Chung, H. C. Chang, and C. Y. Lee, "Efficient power-analysis-resistant dual-field elliptic curve cryptographic processor using heterogeneous dual-processing-element architecture," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 1, pp. 49–61, Jan 2014.

[12] B. Yang, K. Wu, and R. Karri, "Scan based side channel attack on dedicated hardware implementations of data encryption standard," in *2004 International Conference on Test*, Vienna, Austria, Oct 2004, pp. 339–344.

[13] G. Zheng, G. Dawu, Y. Kan, L. Junrong, and H. Yuming, "A novel method for power analysis based on combinational logic in block cipher circuit," *Chinese Journal of Electronics*, vol. 23, no. 1, Jan 2014.

[14] L. J. Eshelman, "The CHC adaptive search algorithm : How to have safe search when engaging in nontraditional genetic recombination," *Foundations of Genetic Algorithms*, pp. 265–283, 1991.

[15] J. W. Bos, J. A. Halderman, N. Heninger, J. Moore, M. Naehrig, and E. Wustrow, *Elliptic Curve Cryptography in Practice*. Springer Berlin Heidelberg, 2014, pp. 157–175.

[16] D. J. Bernstein and T. Lange. (2017, March) Safecurves. [Online]. Available: https://safecurves.cr.yp.to/

[17] R. Liu, S.-y. Zeng, L. Ding, L. Kang, H. Li, Y. Chen, Y. Liu, and Y. Han, "An efficient multi-objective evolutionary algorithm for combinational circuit design," in *First NASA/ESA Conference on Adaptive Hardware and Systems (AHS'06)*, Istanbul, Turkey, June 2006, pp. 215–221.

[18] A. T. Soliman and H. M. Abbas, "Combinational circuit design using evolutionary algorithms," in *CCECE 2003 - Canadian Conference on Electrical and Computer Engineering. Toward a Caring and Humane Technology*, Montreal, Canada, May 2003, pp. 251–254.

[19] J. F. Miller, D. Job, and V. K. Vassilev, "Principles in the evolutionary design of digital circuits—part i," *Genetic Programming and Evolvable Machines*, vol. 1, no. 1-2, pp. 7–35, April 2000.

[20] S. J. Louis, "Genetic learning for combinational logic design," *Soft Computing*, vol. 9, no. 1, pp. 38–43, 2005.

[21] M. Linder, J. Di, and S. C. Smith, "Multi-threshold dual-spacer dual-rail delay-insensitive logic (mtd3l): A low overhead secure ic design methodology," *Journal of Low Power Electronics and Applications*, vol. 3, no. 4, pp. 300–336, 2013.

[22] K. Tan, E. Khor, and T. Lee, *Multiobjective Evolutionary Algorithms and Applications*. Springer, 2005.

[23] M. D. Ercegovac, T. Lang, and J. Moreno, *Introduction to digital systems*. John Wiley, 1999.

[24] Xilinx, "Vivado design suite user guide, partial reconfiguration, UG909 (v2016.1)," Xilinx, All Programmable, Tech. Rep., April 2016.

[25] xilinx. (2016, Dec) ISE design suite. [Online]. Available: https://www.xilinx.com/products/design-tools/ise-design-suite.html