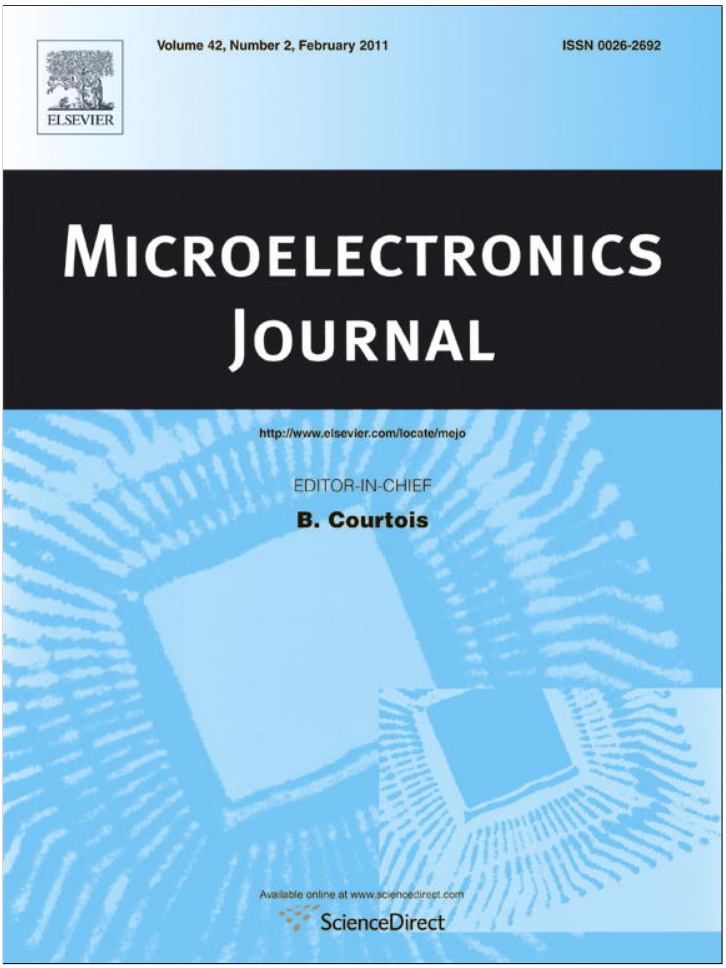

Microelectronics Journal 42 (2011) 462–471

Contents lists available at ScienceDirect

Microelectronics Journal

journal homepage: www.elsevier.com/locate/mejo

Low power asynchronous circuit back-end design flow

Behnam Ghavami *, Hossein Pedram

Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran

ARTICLE INFO

Article history:
Received 2 November 2009
Received in revised form
15 December 2010
Accepted 20 December 2010
Available online 7 January 2011

Keywords:
Low power
Asynchronous circuits
STA
Probabilistic timed Petri-Net

ABSTRACT

This paper introduces a framework for the synthesis of low leakage power asynchronous circuits while maintaining performance requirements. In the proposed framework, a high-level description of the system is received and then the corresponding specification will be decomposed into smaller circuits which is possible to be directly mapped into predefined circuit templates. The proposed flow has the advantage of exploiting a new performance metric and presents an efficient methodology for static estimation of average performance of asynchronous circuits with choices at the template level. The leakage reduction is done via simultaneous supply voltage selection, multiple threshold voltage assignment and template sizing. The power reduction techniques are properly encoded in a quantum genetic algorithm and evaluated simultaneously. Experimental results are given for a number of 90 nm related benchmark circuits and show that this method reduces the total power by close to an order of magnitude, with no or negligible performance penalty.

1. Introduction

Process scaling and aggressive performance improvements have resulted in power consumption becoming a first-order design criterion [1,2]. Power dissipation in electronic circuits consists of dynamic power, static power, and short circuit power consumption. Traditionally, designers have been less concerned with leakage power due to its negligible effect on total power consumption in older technology nodes. However, as transistor sizes continue to shrink, the share of leakage current in total power consumption increases [1].

In addition to leakage power consumption concerns, today high performance VLSI design is challenged by synchronization problems as the clock frequency is increasing by the technology improvements [1]. However, in asynchronous circuits, local signaling eliminates the need for global synchronization and this result in some potential advantages. Beside the elimination of the clock skew and tolerating interconnect delays, asynchronous circuits are more tolerant to process variations and external voltage fluctuations. They are more modularly synthesizable and potentially faster [1,3]. Despite these advantages, asynchronous circuits have some drawbacks such as the lack of automatic synthesis tool and evaluation and optimization methods. So, it will be worthy to work on a design framework for asynchronous circuits.

Most leakage power reduction techniques cannot be applied directly to asynchronous circuits in the same way that it can be done for the synchronous circuits. The basic problem is that the complex interaction of concurrent processes makes static performance estimation of asynchronous circuit very difficult. While synchronous performance estimation is based on a static critical path analysis affected only by the delay of components and interconnecting wires, it has been shown that the performance of an asynchronous circuit depends on factors like the number of tokens in the circuit and the value of input data items; therefore evaluating the performance of asynchronous circuit is more complex. Therefore, it is necessary to mind about appropriate performance evaluation method and leakage power optimization considerations while proposing CAD tools for asynchronous circuits.

Many academic tools and techniques have been developed to address the automatic synthesis of asynchronous circuits [4,5]; however, the lack of a unified framework to automated synthesis and static performance estimation and power optimization is highlighted. This paper proposes the following considerations for a integrated design framework of a low leakage power and high performance asynchronous circuit: (1) A Probabilistic Timed Petri-Net model is used to analyze the performance of template based asynchronous circuits; (2) A Novel method for static performance analysis of real asynchronous circuits including choices is presented; and (3) A CAD framework which simultaneously exploit multiple supply voltage assignment, multiple threshold voltage assignment and template sizing for designing real asynchronous circuits. We have looked at the joint optimization of multiple techniques in real asynchronous circuits. It helps to achieve maximum power savings compared to a sequential application of a single variable optimization. Also the method can be easily extended to include other techniques such as multiple gate oxide assignment.

The remainder of this paper is organized in the following manner. In Section 2 a brief overview of asynchronous circuits is presented to give the reader more knowledge about the features of asynchronous

* Corresponding author. Tel.: +98 2164542701.
E-mail addresses: Ghavamib@aut.ac.ir (B. Ghavami), pedram@aut.ac.ir (H. Pedram).

0026-2692/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mejo.2010.12.001

circuits. Section 3 reviews different leakage power reduction techniques and provides a survey of previous works. Section 4 introduces the proposed power optimization framework. Section 5 presents a synthesis tool for QDI asynchronous circuits. Section 6 describes the Probabilistic Timed Petri-Nets model as the dominant performance analysis model and Section 7 introduces a novel static performance analysis for asynchronous circuits with choices. Section 8 introduces the proposed parameter assignment methodology for the decomposed circuit. In Section 8 we present experimental results using some related benchmarks. Conclusions appear in Section 9.

2. Background

2.1. Asynchronous circuits

Asynchronous circuits represent a class of circuits not controlled by a global clock but rely on exchanging local requests and acknowledge signaling for the purpose of synchronization. Among the various methods of asynchronous design, the Quasi delay insensitive (QDI) method is capable of having better performance and low power consumption [8].

These circuits are composed of individual modules which communicate to each other by means of point-to-point communication channels. Therefore, a given module becomes active when it senses the presence of an incoming data. It then performs the computation and sends the result via output channels. Communications through channels are controlled by handshake protocols [8]. While the first generation of asynchronous synthesis tools was mainly focused on control synthesis (make them inapplicable for practical circuits), the new generation of synthesis methods targets the use of pre-designed asynchronous templates [4].

2.2. Leakage power reduction techniques and related works

Researchers have proposed different circuit techniques to reduce leakage power without impacting performance using the available timing slack. In this section, we survey the related work on multiple-Vth and multiple-Vdd assignment and gate sizing techniques which exploit the timing slack to reduce the leakage power of the circuit [1,2,10,9,12].

Due to the quadratic relationship between dynamic power consumption and Vdd, reducing the supply voltage is the most effective way to lower the dynamic power, at the expense of increasing gate delay. In order to prevent the negative effect on performance, the threshold voltage (Vth) must be reduced proportionally with the supply voltage so that a sufficient driving current is maintained. This reduction in the Vth causes an exponential increase in leakage power, which in turn can raise the static power of the device to unacceptable levels.

To counter the loss in performance while improving the power efficiency, multiple Vdd [6] and multiple Vth [9] techniques have been proposed. The gates on critical paths operate at the higher Vdd or lower Vth, while those non-critical paths operate at the lower Vdd or higher Vth, thereby reducing overall power consumption without performance degradation. On the other hand, in gate sizing techniques [10], logic gates on critical paths may be sized up to meet timing requirement, at the expense of higher power consumption; while those on non-critical paths can be sized down to reduce the power consumption. Some good works about these three techniques is presented in [13,12].

However, these techniques cannot be used for asynchronous circuits in the same way that it can be done for synchronous one as the performance analysis of asynchronous circuit is completely different. While synchronous performance estimation is based on a static critical path analysis affected only by the delay of components and interconnecting wires, it has been shown that the performance of an asynchronous circuit depends on dynamic factors like the number of tokens in the circuit [15]. In the clocked circuits, the critical path has a clear beginning and a clear end created by latches while no clear separation is available in asynchronous circuits.

Some techniques have been proposed in asynchronous circuit context to reduce the power consumptions; for example, combining dynamic voltage scaling and adaptive body biasing in [16], dual-threshold voltage technique to reduce leakage power dissipation of asynchronous linear-pipelines in [8], multi-threshold C-Element and multi-threshold micropipeline circuits in [18], and in [13,14], a technique for design of dual-Vth asynchronous circuits. However, the reported techniques are either not evaluated for automatic design [16–18], or just focus on design of dual-Vth linear asynchronous circuits which do not consider choices [13,14], not considering real asynchronous circuits with choices which is focused in this paper.

This paper proposes an integrated design framework of a low leakage power and high performance asynchronous circuit. Many aspects similar to the proposed methodology have been addressed for other design areas [19–21].

3. Power optimization framework

Fig. 1 shows the general structure of the proposed leakage power optimization framework which can be considered as the following five individual portions: QDI synthesis tool, static performance analyzing, transistor's parameters assigning, layout synthesis, and simulation at various levels. The design process starts with a high level specification of circuits. The framework uses Verilog-CSP [24], an extension to standard Verilog for the purpose of expressing the input description. READ and WRITE macro-operations are added to the Verilog in order to model the handshake operation on communication channels which is proposed in [24].

By applying a set of preserving function transformations to the high level specification, decomposition step divides the original circuit into predefined elements as a netlist. The output of decomposition step supplies the input of Model Generator (MG) and Static Ranges Computer (SRC) for the purpose of performance analyzing. MG converts the Verilog-CSP description of predefined templates to a Probabilistic Timed Petri-Net (PTPN) model and then, SRG computes the ranges of conditional variables in choice templates. Token Assigner, a Petri-Net simulation engine which supports dependent choice resolution and probabilistic models for choices, runs the PTPN model of circuit and provides the dynamic information of the original circuit such as token assignment and resolve the location of data tokens in a Petri-Net netlist.

The developed core simulator is based on systemC [22]. Static Timing Analyzer (STA) runs for the performance estimation of the synthesized circuit which is presented in PTPN format. Parameter Assigner (PA) employed STA for determining the value of transistors' parameters of each template (such as Vdd, Vth and Size). Then Template Synthesizer (TSYN) receives an intermediate source code containing a number of pre-designed template compatible modules and optionally a top-level netlist, and generates a netlist of predefined templates with dual rail ports. The output of TSYN is a netlist of the available standard library elements. Hence, the proposed flow is provided for the standard cell library in order to generate the final layout applying standard layout tools. The simulation flow is intended to verify the correctness of the synthesized circuit at all levels of abstraction. In the following sections we describe the functionality of these stages.

4. AsyncTool: synthesis of QDI asynchronous circuits

AsyncTool [5] is an asynchronous synthesis tool developed for automatic synthesis of QDI asynchronous circuits. Behavioral

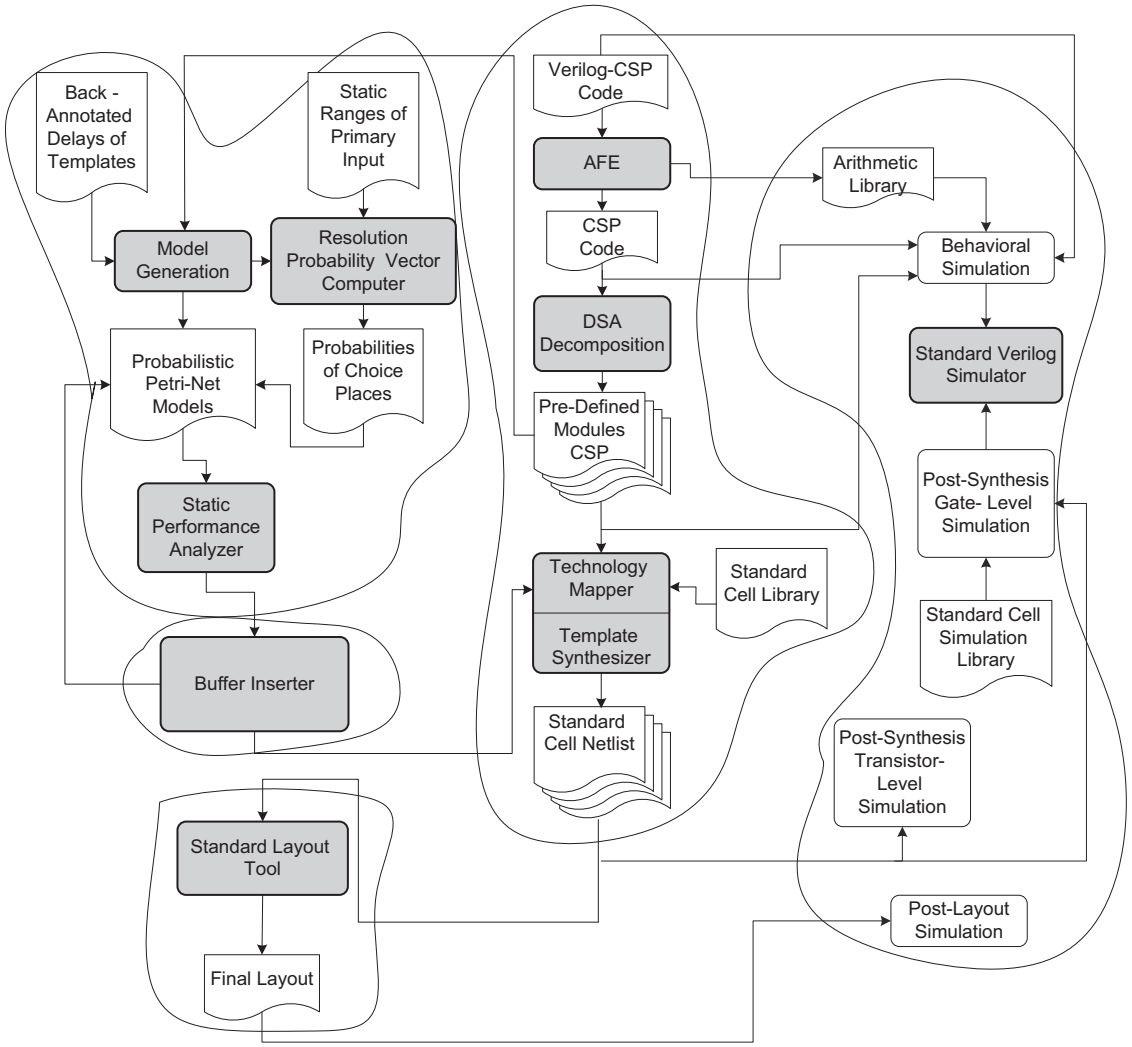

Fig. 1. Power reduction framework and its interface with asynchronous synthesizer.

```
...
`READ(F,a)
z = f(a);
`WRITE(W,z)
...
```

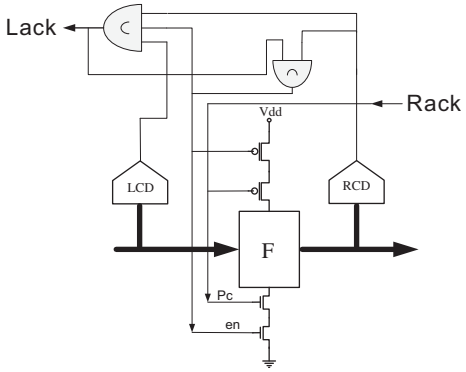

Fig. 2. Internal structure of PCFB template (C: C-element gate, RCD/LCD: Right/Left completion detection).

description will be converted to a netlist of standard-cell elements through several steps of synthesis flow. For simpler synthesis, first arithmetic operations are extracted from the code and the major steps of synthesis only works on the codes without any arithmetic operations. This is done by the Arithmetic Function Extractor (AFE) which also replaces the arithmetic functions by standard library modules. The two major steps in AsyncTool synthesis are Decomposition and Template Synthesizer (TSYN).

AsyncTool uses Pre-Charge logic Full-Buffer (PCFB) templates for its predefined templates. A PCFB template [8] is an asynchronous buffer circuit that in each cycle of its operation reads some inputs, performs a particular calculation, and then writes the results to one or more of its output ports. Fig. 2 shows the internal structure of PCFB schematic and its corresponding Verilog-CSP presentation.

5. Performance model for template based asynchronous circuits

The proposed performance model is based on Time Petri-Nets which have been already used as a performance model to analyze concurrent systems [25]. In the following, we describe necessary Petri-Net background and then introduce our performance model.

5.1. Background

Petri-Nets are used as an elegant modeling formalism to model concurrency, synchronization, and choice in many applications including asynchronous circuit modeling [25]. A Petri-Net is a four-tuple $N=(P,T,F,m_0)$ where P is a finite set of places, T is a finite set of transitions and $F \subseteq (P \times T) \cup (T \times P)$ is a flow relation, and $m_0 \in \aleph^{|p|}$ is the initial marking. A marking is a token assignment for the

places and represents the state of the system. The preset of an element $x \subseteq (P \cup T)$ is defined as $\bullet x = \{y \in P \cup T | (y,x) \in F\}$ and its postset is defined as $x\bullet = \{y \in P \cup T | (x,y) \in F\}$. Formally, a marking is a mapping $M: P \rightarrow \{0,1,2,...\}$ where the number of tokens in place p under marking M is denoted by $M(p)$. If $M(p) > 0$, then the place p has tokens within. All the places in our system are 1-bounded which means that they maximally can contain only one token. A transition t is enabled at marking M if $M(p) \geq 1, \forall p \in \bullet t$. An enabled transition may fire eventually in a feasible asynchronous circuit specification. The firing of t removes one token from each place in its pre-set and inserts one token to each place in its post-set.

A Petri-Net is usually represented as a bipartite graph in which P and T are its nodes. For any two nodes $x$and y, if $(x,y) \in F$ then there is a directed arc from x to y. Timed Petri-Net is a Petri-Net in which some transitions or places can be annotated with delays. The semantics of a Timed Petri-Net state that a transition t fires after tokens in places $p \in \bullet t$ reside in P at least $d(p)$ time units. Different delay models can be used such as fixed, uniform bounded or normalized delays.

A cycle c is a sequence of places $p1,p2,\cdots,p1$ connected by arcs and transitions whose first and last place is the same. The cycle metric ($CM(c_k)$) is the sum of the delays of all associated places along the cycle c_k, $D(c_k)$, divided by the number of tokens that reside in the cycle k, $m_0(c_k)$, is defined as:

$$CM(c_k) = D(c_k)/m_0(c_k)$$

$$\text{where } D(c_k) = \sum_{\forall i \in C_k} d_i \tag{1}$$

The cycle time of a Timed Petri-Net is defined as the largest cycle metric among all cycles in the Timed Petri-Net [26]. The intuition behind this well-known result is that the performance of any computation modeled with a Time Petri-Nets is dictated by the cycle time of the Time Petri-Nets and thus the largest cycle metric.

$$C_{Time} = \text{Max}[CM(c_k)], \quad \forall c_k \in TPN \tag{2}$$

5.2. Simple template model

In Petri-Net model, the simplest form of a full buffer is a simple buffer that only reads a value from its input and writes it to its output. This behavior can be modeled simply as shown in Fig. 3. Transition *tW* is analogous to the write statement while place *pWa* emulates the write acknowledge. Similarly *pF* can be seen as the dual for read statement while *tRa* is the corresponding acknowledge.

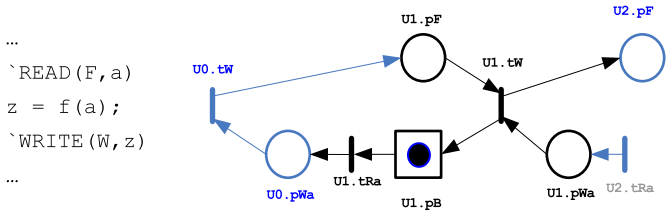

Fig. 3. PCFB template is modeled using Timed Petri-Nets.

This simple template model is very similar to FBCN model presented in [32]. We have considered delays on the places, therefore forward delay and backward delay can be put on *pF* and *pB*. In this model, the $d(f)$ represents the forward latency of a channel equals to the delay through an empty channel (and associated cell), while the corresponding $d(p)$ represents the backward latency of channel equaling to the time it takes the handshaking circuitry within the neighboring cells to reset, enabling a second token to flow.

The values of these parameters are the normalized delays which are back annotated from the associated cell layout. The cycle time of a deterministic circuit is captured by the maximum cycle metric of the corresponding Timed Petri-Net model (Eq. (2)) which can be resolved using traditional Maximum Mean-Cycle algorithms [26]. The throughput of the circuit is the reciprocal of this value.

5.3. Conditional template model using Probabilistic Timed Petri-Net

It can be shown that any conditional statement in the specification of an asynchronous circuit leads to at least one conditional template in the final implementation. As an example, in Data-Driven Decomposition [4] every variable that is read or written in a conditional block will produce a conditional write or conditional read template. Conditional templates are referred to those templates that their behavior is dependent on the value of data token. We model the conditional templates by means of Probabilistic Timed Petri-Net (P-TPN) concept. For the formal definition of P-TPN, we are defining a few terminologies.

[Def. 1]: If transitions *ta* and *tb* are sharing an input place, they are said to be in choice relation, and denoted by choice *p(ta, tb)*.

[Def. 2]: In a choice place *p(ta, tb)*, the transitions followed by this choice should be associated with probabilities of choice. The probability of choice for this transition is denoted by *Prob (Transition$_i$)*, which for all transitions followed by one choice:

[Def. 3]: A vector which contains the probability's value of the all choices in a circuit is named as Resolution Probability Vector. Resolution Probability Vector of a circuit contains *M* choices can be presented as follow:

$$RPV = \left[(P_{1,1},P_{1,2},P_{1,3},..,P_{1,k}),(P_{2,1},P_{2,2},P_{2,3},..,P_{2,l}),...,(P_{M,1},P_{M,2},P_{M,3},..,P_{M,o})\right]$$

where

$$\sum_{i=1}^{k} P_{1,i} = 1, \quad \sum_{i=1}^{l} P_{2,i} = 1,...., \quad \sum_{i=1}^{o} P_{M,i} = 1 \tag{3}$$

Choice place *p* is an extended free choice if all its post-set transitions can be enabled regardless of external influence. In other words selection of the transition to fire is probabilistic. A special case of extended-free choice is a free-choice place *p* for which all output transitions of *p* have only one input place. A place *p* is a unique choice place, if no two output transitions of *p* can be enabled at the same time. In this case, the selection of the firing transition is not probabilistic. In P-TPN model, each free choice place *p*, is assigned

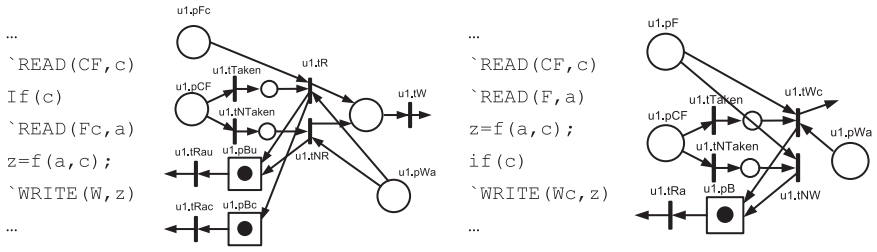

Fig. 4. Left model includes a conditional read and right includes conditional write.

a probability resolution to resolve the choice, i.e. select the firing transition among the possibilities. Transitions with higher probabilities are more likely to consume tokens whenever their input places contain tokens.

Conditional template in our modeling approach will contain choice and merge constructs. Fig. 4 shows two examples with their corresponding high level code segments. During the elaboration of the model, some choices are defined to be dependent on other choices and the probability vectors of dependent choices are selected at analysis process based on the state of the master choice.

During model generation, based on the specification of the template a proper model is selected from model library; however, automatic generation of such models is also possible.

6. Static performance analysis technique

Considering a deterministic asynchronous circuit, from the performance point of view, the important issues are the handshaking behavior and forward and backward delays. However, this is correct for every circuit where the data flow is completely independent of the values of the data token and this is not a valid assumption for practical asynchronous circuits where there are lots of conditional statements in the high level specification. The advantage of asynchronous design compared to synchronous design is to allow the circuit performance to be dictated by the average case (not worst-case), whereas the Maximum Mean-Cycle method does not seem to attempt to analyze the performance with an average case of conditional circuits at all. The weak point of Maximum Mean-Cycle method is that it does not consider the choice probabilities of the events. Exploiting Maximum Mean-Cycle method to performance analysis of a Time Petri-Nets with choices leads to achieving an inappropriate and pessimistic performance metric, while it does not take the probability of choices. One of the most important objectives of this work is to extend the concept of stochastic cycle metric using Probabilistic Timed Petri-Net.

6.1. A novel average-case performance metric for circuits with conditional statements

This section describes a novel performance metric for asynchronous circuits which reflects on the average case for conditional cases. The main idea of this metric is that it considers the choice probabilities of events. For illustration we assume that an arbitrary Probabilistic Timed Petri-Net (PTPN) model has only one choice place with n outcome transitions. In this case the PTPN can be transferred to n Timed Petri-Net which all of them are completely similar to the original PTPN except that the choice place is converted to a normal place and only one of the outcome transitions is participated in construction of each Timed Petri-Nets. Each of these Timed Petri-Nets has a cycle time which can be determined by Eq. (2). The probability of building each Time Petri-Net is related to the probability of associated outcome transition. As a result, the average cycle time for P-TPN can be resolved as below:

$$C_{Time_{\text{Probabilistic}-TPN}} = P_1\left(C_{Time_{TPN-1}}\right) + P_2\left(C_{Time_{TPN-2}}\right) + \ldots + P_n\left(C_{Time_{TPN-n}}\right) \quad (4)$$

where $C_{Time_{TPNT-i}}$ is the cycle time of a Timed Petri-Net which contains the ith transition and P_i is the probability correlated to ith transition. For the case that a PTPN has more than one choice place, the proposed manner can be exploited recursively as follows:

$$C_{Time_{\text{Probabilistic}-TPN}} = \sum_{\forall TPN-k}\left(C_{Time_{TPN-k}} \prod_{\forall choice-place_i \in TPN-k} P_i\right) \quad (5)$$

The time complexity of traditional approach to compute the cycle time of a TPN is $O(N_P^3)$ [26],where N_P is the number of elements in places set in TPN, as the result the time complexity of computing the proposed cycle time of a PTPN is $O(N_P^3 n^{N-\{CP\}})$ where N-{CP} is the number of choice places set in the PTPN and n is the maximum number of outcome transitions in one choice place. To improve the time complexity of this method, a new procedure described in Fig. 5 is proposed. This method is based on finding all of the elementary cycles in a directed graph. The key characteristic of this procedure is that it sorts the cycles of the directed graph merely one time. The time complexity of this procedure is $O(N_p^4)$ which is related to step 1.

For each of the choice places, the *Resolution Probability Vector Computer* (*RPVC*) must provide a probability vector. The main idea behind the *RPVC* is to use the static range of the primary inputs of the circuit to determine the static range of internal signals similar to compiler-based techniques. Of course, it must be done considering the circuit specification. When static range of each internal variable is known we can simply judge about the probability of the conditions to provide the probability vectors and choose among the various options for probability models. The next plan of the *RPVC* is to make a probability model for the choice places. In this model the dependency of one choice to the other choices and dependency of choices to variables can be resolved. Based on the nature of dependencies, a model is chosen among the following possibilities for a choice place:

- *Independent probability model:* This model is completely independent of sate of system and is chosen whenever the outcome transitions do not depend on the state of any other choice.
- *Dependent probability model*: This is used for the cases where the outcome transitions depend on the previous firing of the choice itself or depends on the transition firing of any other choice places.

```
Cycle time computer
begin
 1. Find all of the elementary cycles in the PTPN, using the method
    introduced in[34].
 2. Sort the cycle metrics using a quick sort algorithm.
 3. Determine the set of outcome transitions and correlated places for each
    built Time Petri-Net.
 4. Convert the arranged list to a set of n^CP arranged lists via eliminating
    the cycles which don't participate in each TPN.
 5. Determine the cycle metric for each list(TPN).
 6. Compute the cycle time of the P-TPN using Eq.5.
End
```

Fig. 5. Procedure to determine the cycle time of a P-TPN.

A choice place that depends on a conditional variable which is manipulated in a conditional statement is in some way dependent on the other choices. For both preventing deadlock and obtaining more precise performance estimation, these conditional decisions must be resolved. For each condition in every module, a choice place will be generated during model elaboration phase. Regarding the resulted dependency directives, the *RPVC* forms a set of dependent choices and selects one of these choices as the master and all the other choices in the set will be considered as its slaves. In the upcoming sections we will explain how the static ranges and probability can be calculated.

6.2. Static ranges computation

In the *RPVC*, there exists a table of static ranges for each conditional variable. Each static range entry is tagged with a list of conditions which determine its validity region that includes all of the conditions of the nested conditional blocks inside which the variable is assigned a value. Condition list can be aggregated by logical AND to have a simpler tag with single conditional expression.

The set of all tagged static ranges of a variable v is shown by $TSR(v)$. Each $r \in TSR(v)$ is an ordered threesome in the form $< r.ct, r.vt, r.sr >$ where *r.ct* is the conditional expression tag, *r.vt* is the variable expression tag and *r.sr* is the static range which is shown by a set of disjointed intervals generally. Set of static ranges for input variables must be placed by the user or can be set with the default values.

To be able to consider the effect of a countered loop on static ranges, the process of the computation of static ranges must be supported to compute the minimum and maximum number of consequent occurrence of some conditional statements. For example, in a countered loop, range of a variable which is sum of two other variables is dependent on the number of iterations. At the start of analysis, the *RPVC* detects countered loops. Recalling that the nested loops are not allowed in our specification, the general form of a countered loop is as follows:

```
always
begin
    …
    if (Counter == Var_final)
        Counter=Var_initial;
    else
        Counter=Counter ± Var_inc;
    …
end
```

Assuming parametric ranges of a variable involved in the generic form of a countered loop, the maximum and minimum loop iterations are computed as:

$$\text{Max-Iter} = \left\lceil \frac{(\text{Max}(\text{Var}_{\text{initial}}) - \text{Min}(\text{Var}_{\text{final}}))}{\text{Min}(\text{Var}_{\text{inc}})} \right\rceil$$

$$\text{Min-Iter} = \left\lceil \frac{(\text{Min}(\text{Var}_{\text{initial}}) - \text{Max}(\text{Var}_{\text{final}}))}{\text{Max}(\text{Var}_{\text{inc}})} \right\rceil \quad (6)$$

Selection of the probability models is done based on a set of simple rules. Briefly dependent probability model is selected based on the content of tagged static ranges tables for the variables. Dependent model is used for the cases that value of a variable is changed regarding to a condition on that or other variable.

6.3. Choice probabilities computation

Computing the probability of conditions is discussed in this section. Consider that *CP* is a choice place and its outcome transitions will be fired based on the conditional variable $CV(X > Y)$. The static range of variables X and Y is determined now. Assuming uniform distribution of the variables, we can determine the probability of the outcome transitions for the correlated choice places easily.

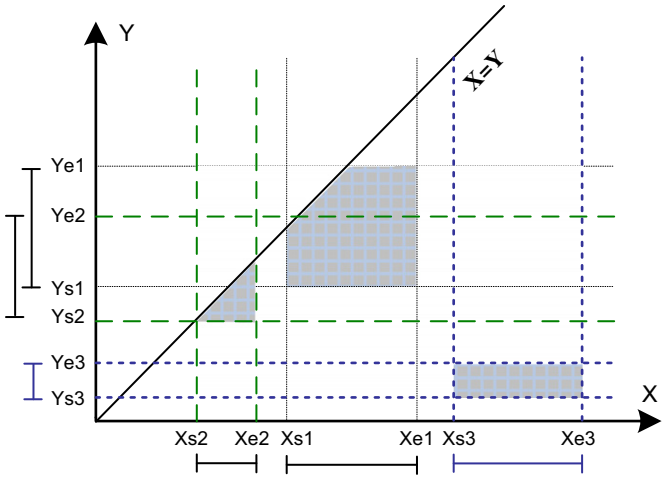

Fig. 6. Dependent choice place with jointed intervals.

In conditional circuits, both data flow and value of data tokens are dependent on the selections made by conditional statements. As the result, in the case of dependent choice, there exist a number of static ranges for each choice place. Each of these ranges is coupled to one probability value. Fig. 6 shows the intervals of static ranges for an arbitrary dependent choice model. Since the value of probability for appearance of each range can be dedicated, the overall probability can be computed using Eq. (7):

$$\begin{aligned} P(X > Y) = {} & [P_1(P(X(xs_1,xe_1) > Y(ys_1,ye_1)))] \\ & + [P_2(P(X(xs_2,xe_2) > Y(ys_2,ye_2)))] \\ & + \ldots + [P_i(P(X(xs_i,xe_i) > Y(ys_i,ye_i)))] \end{aligned} \quad (7)$$

where P_i is the determined probability associated to the ranges $X(xs_i,xe_i)$ and $Y(ys_i,ye_i)$, and $P(X(xs_i,xe_i) > Y(ys_i,ye_i))$ is the probability of $(X > Y)$ when X and Y are between (xs_i,xe_i) and (ys_i,ye_i), respectively and it can be calculated easily. A hybrid model of dependent choice model and disjointed intervals is also possible. With the use of the discussed methods, calculating the probability of this model is straightforward.

7. Template's parameters assignment methodology

Quantum genetic algorithm (QGA) is characterized by principles of quantum computing including concepts of qubits and superposition of states [27]. This algorithm is concentrated on the quantum-inspired evolutionary computing for a classical computer [28]. In QGA, there is no wide spectrum of parameters and operators; hence, makes it as a faster and an effective heuristic algorithm.

Our power optimization flow uses a quantum genetic algorithm [27,28]. The circuit configuration information, such as supply voltage assignment, is encoded into qubit. A qubit may be in '1' state, '0' state, or in any superposition of the two. So, the state of a qubit can be presented as $|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle$ where $|\alpha|^2$ and $|\beta|^2$ gives the probability that the qubit will be found in '0' state and '1' state, respectively. This algorithm is a probabilistic algorithm and is similar to classical genetic algorithms and it also maintains a population of n qubit individuals at generation g in

$$Q(g) = \{q_1^g, q_2^g, \ldots, q_n^g\} \quad (8)$$

where q_j^g is a qubit individual defined as

$$q_j^g = \begin{bmatrix} V^g_{th-h\,j1} & V^g_{th-h\,j2} & \cdots & V^g_{th-h\,jm} \\ V^g_{th-l\,j1} & V^g_{th-l\,j2} & \cdots & V^g_{th-l\,jm} \\ V^g_{dd-h\,j1} & V^g_{dd-h\,j2} & \cdots & V^g_{dd-h\,jm} \\ V^g_{dd-l\,j1} & V^g_{dd-l\,j2} & \cdots & V^g_{dd-l\,jm} \\ S^g_{1\,j1} & S^g_{1\,j2} & \cdots & S^g_{1\,jm} \\ S^g_{2\,j1} & S^g_{2\,j2} & \cdots & S^g_{2\,jm} \\ \cdots & \cdots & \cdots & \cdots \\ S^g_{8\,j1} & S^g_{8\,j2} & \cdots & S^g_{8\,jm} \end{bmatrix} \quad (9)$$

where m is the number of templates in the circuits and

$$|V_{th-h}|^2 + |V_{th-l}|^2 = 1,$$
$$|V_{dd-h}|^2 + |V_{dd-l}|^2 = 1,$$
$$|S_1|^2 + |S_2|^2 + \ldots + |S_8|^2 = 1.$$

The optimization flow begins with a random generated initial population, which consists of many randomly generated circuit configurations. The optimization flow is an iterative procedure. The qubit individuals with better fitness will survive at each generation and is applied update one operation to be a new set of qubit individuals—or new circuit configuration. The iteration continues until the termination criterion is met. After these procedures, the best solution among $P(t)$ is selected in the step, and if the solution is fitter than the stored best solution, the stored solution is changed by the new one. More detail can be found in [6].

7.1. Binary population encoding

We can encode all the tuning variables into a binary individual string. Fig. 7 shows both the structure of a binary individual and an encoding example, representing m templates in a circuit. This encoding example is based on the assumption that we use a dual-Vdd, dual-Vth library with eight discrete sizes for each type of template. For example, in Fig. 7, the chromosome shows that Template 1 is assigned to use lower Vdd, higher Vth and has the size with number 6. The type of each template and the circuit topology information are known prior to calculate the power, delay and area based on the binary population configuration.

7.2. Fitness function

The fitness function, which decides the surviving chance for a specific qubit individual, is related to the power consumption, area and the performance of the circuit.

7.2.1. Power

Our goal is to find a configuration such that the power consumption for the circuit is as low as possible. Therefore, the fitness of a chromosome should be related to the power consumption of that particular configuration. For the dynamic power of the template, the switching activity is obtained by exhaustive TPTN simulation with the assumption that the input probabilities of being high or low are equal and independent.

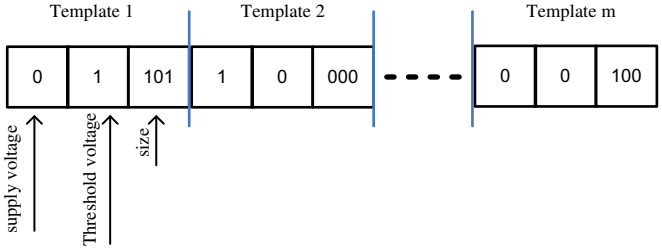

Fig. 7. Structure of a chromosome.

Leakage current of an asynchronous circuit is the sum total of the leakage currents of all of the templates. The leakage of a template depends on the number of transistors that are turned off; hence, on the inputs. In our study, we calculate the gate leakage under each input pattern. The leakage numbers under different input patterns are used to drive at a weighted average of leakage numbers. Subsequently, the leakage of a gate at either low or high threshold voltage is represented with a value, which is the weighted average of its leakage under various input patterns.

When using multiple supply voltages in a circuit, level converters are required whenever a logic gate at the lower supply has to drive a gate at the higher voltage [31]. To model a level converter in TPTN, a simple buffer is placed in connection channels of the related circuit and all needed communications are established. The overall power consumption for the circuit should also include the level converters' power consumptions.

7.2.2. Area

If the area in a circuit is larger than the area constraint, the configuration is not acceptable and the corresponding individual qubits should have little chance to survive. The area from level converters and large template are taken into account.

7.2.3. Performance

It should be noted that the power optimization is under a specified timing constraint. In asynchronous circuits, the performance is defined similar to throughput, i.e. the reverse of global cycle metric. If the global cycle metric in a circuit is larger than the timing metric requirement, the configuration is not desirable and the corresponding chromosome should have little chance to survive. Again, the delay from level converters is taken into account.

Therefore, fitness (cost) function is as:

$$\Phi = \frac{1}{P_{\max} - P_{\min}} P_{Total} + \frac{1}{A_{\max} - A_{\min}} A_{Total} + \alpha\left(\frac{1}{CM_{\max} - CM_{\min}} CM\right) \quad (10)$$

where P_{Total} and A_{Total} is the total power and area of the circuits, CM is the largest cycle metric and $P_{\max}$, P_{min}, $A_{\max}$, A_{min}, $CM_{\max}$ and $CM_{\min}$ are constants which are used for normalization and α is user specified constants. Since the performance is a bottleneck parameter in this work, we penalize the performance reduction with a large constant multiplied (α).

7.3. Control parameters

While generating the initial population, we have to set an appropriate population size. If the population size is too small, the genetic diversity within the population may not increase for many generations. On the other hand, a large population size increases the computation time for each generation but it may take fewer generations to find the best solution. Extensive simulations on a wide range of functions help us to conclude that a small population of size 10–15 performs very well. We set a maximum generation and specify that if the power reduction is less than 0.0005% during the last 200 generations, the evolution stops without going through all generations.

8. Experimental results

To demonstrate the efficiency of the proposed performance driven synthesis flow, we conducted experiments on a suite of benchmark circuits, using a SUN Ultra Sparc 10 workstation with

1 gigabyte of memory. At first, the proposed performance estimation method is evaluated. We compare the acquired estimated results with the results of Petri-Net simulation at the template-level abstraction and also those are achieved in the post-synthesis simulation. Then, we show the power optimization results. At last, the proposed average-case power optimization scheme is compared with the worst-case power optimization ones by synthesis of an asynchronous Reed–Solomon decoder as the benchmark.

8.1. Performance estimation results

We have synthesized a Hamming encoder with un-optimized progression of AsyncTool [5] and performed Petri-Net and post-synthesis simulations to determine the average cycle time and the average throughput. Its Petri-Net has the 27 choices and three level of dependency exist in 5 choices (others have 2 level of dependency). Average values, which must be considered as performance metric, are completely dependent on the value of the inputs. A set of stimuli (150 stimuli vectors) is applied to the Petri-Net and post-synthesis simulation to compute the average performance metrics. Data items in the stimuli are chosen in completely random process with a known static range for each input. We have developed a Petri-Net simulation engine which supports simple choices. The core simulator is based on SystemC. SystemC models for transition and places are developed and the tool is also able to automatically elaborate a detailed SystemC model for each input Petri- Net. Post-synthesis simulation is done with a Standard Verilog Simulator (Such as Modelsim). The proposed static performance estimation method is also applied, however, no stimuli is needed as the simulation is totally probabilistic.

Fig. 8 shows the average throughput for both simulations and the static analysis manner. As it can be seen in the figure the estimated throughput follows the average Petri-Net and post-synthesis simulation result. The total error is computed as 5.78% and 7.26% compared to Petri-Net and post-synthesis simulation, respectively, which is acceptable considering the gain in runtime. Higher precision can be achieved using more precise static ranges for primary inputs. Another reason of difference in obtained estimated results comparing to simulation ones is the number of initial tokens in the circuit that is affect on the accuracy of performance estimation.

Post-synthesis and Petri-Net simulation takes 51 and 37 s for the average throughput, respectively, while this time for the probabilistic simulation is only about 2.4 s which leads to a gain up to 21 and 15 in runtime.

8.2. Power optimization results

To test our method, we construct a dual-Vth and dual-Vdd and octal-size standard cell-library. The library was characterized using Berkeley 65-nm BSIM predictive model [29]. Based on the conclusion in [30] that the optimal second Vdd in dual-Vth system should be ~50% of the higher supply voltage, the possible gate supply voltages for our library are 1.1 and 0.55 V. For NMOS (PMOS) transistors, the high threshold voltage and the low threshold voltage are 0.18 V (−0.18 V) and 0.10 V (−0.10 V), respectively. The level converter implementation is borrowed from [11].

The ISCAS'98 benchmark circuits are chosen to map to our library. The AsynchTool is employed to synthesis benchmarks. Then developed tool automatically translates the decomposed circuit to its Petri-Net equivalent. Inputs and outputs of the circuit are connected to each other in Petri-Net structure to form a closed loop system. Initially all tokens placed in input nodes. The sizes of TPTNs range from 260 to 4500 nodes. The circuits were first optimized for maximum speed, and then, were optimized for lowest power consumption. The runtime for our benchmark ranges from 10 to 400 s, depending on circuit sizes and the structure of TPTN model of circuit.

Fig. 9 presents the leakage power and dynamic power breakdown for maximal speed optimized circuit and minimal power optimized circuit, respectively. In *Max-Performance* case, the leakage power accounts for average 79% of the total power while the dynamic power accounts for 21% of the overall power. After applying all power reduction techniques, the static power accounts for only about 40% of the overall power while dynamic power accounts for 60%. A significant part of overall power reduction are from the static power reduction by using these techniques together.

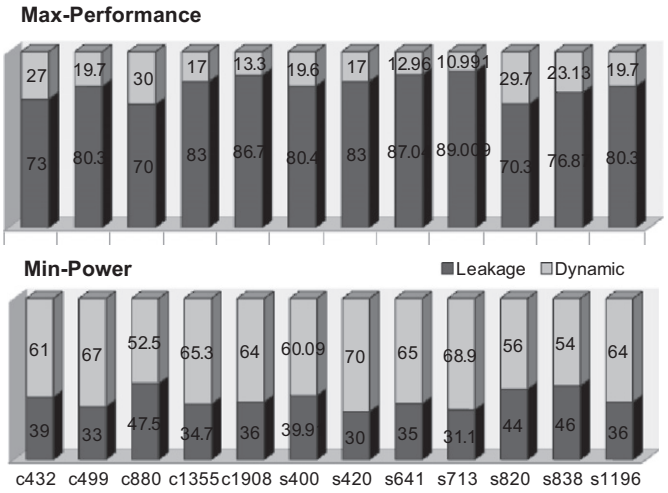

Fig. 9. Power breakdown for speed/power optimized circuits (top/down respectively).

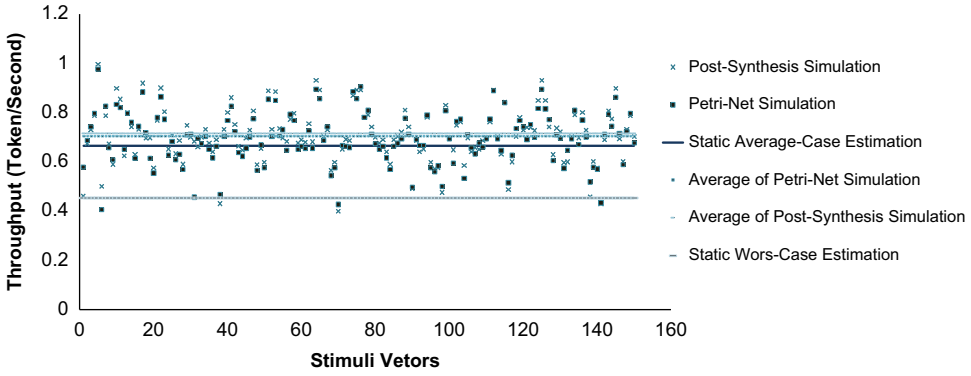

Fig. 8. Average throughput comparison.

In Fig. 10, we use the circuit S641 as an example to illustrate the dynamics of proposed flow. The effectively lowest total power is obtained when the targeted performance is relaxed by 25%. As expected, both dynamic and leakage power decrease as the performance constraint is relaxed further. Promoting more relaxation would not help much. Since our target in this work is that the optimization of power consumption with lowest performance penalty, so we agree to up to 5% performance penalty in optimized circuits.

Fig. 11 shows the comparison of power consumption with only one low power technique available at a time to one equipped with three low power techniques. We can easily see from the figure that the dual-Vdd is the most effective way to reduce power while the dual-Vth is the less effective one. However, dual-Vth can still maintain good timing constraint achievements compared to the other three techniques. The curve with all available techniques can easily maintain low power consumption without huge incurring performance degradation.

To verify the real performance of originated circuits, we quantify the dynamic performance of both synthesized non-optimized and optimized circuits by a Timed Petri-Net simulator (again for the sake of blind review, we do not cite its name here). Fig. 12, which identifies the circuit throughput, shows that there is only 2.5% performance penalty on average in case of optimized circuits. It is more important to note that this scheme is easy to extend for more complicated standard cell-library.

8.3. Comparison to worst-case optimized circuits

In this section we used a circuit to better explore the efficiently of the proposed optimization flow in practice. We selected a Reed–Solomon decoder as a test-bench [38]. The general optimized synthesis approach has been evaluated on this circuit and separately on its modules and the results have been shown in Table 1. Then, we conducted two power driven synthesis experiments on this benchmark: average-case (this work) and worst-case [14] power driven synthesis methods.

A set of stimuli (1000 stimuli vectors) is applied to the post-synthesis simulation to compute the average performance metrics of all designs. The power comparison between un-optimized and average-case optimized circuits demonstrates that an optimized circuit typically has significant improvement (up to 7 X) compared to its un-optimized synthesized circuit. We compare the average-case optimizer to worst-case optimizer and report the percentage improvement. Table 1 compares the worst-case and average-case power optimizers. The results demonstrate that our average case optimizer can yield significant power improvements over the traditional worst-case optimizer. This improvement may be attributed to the fact that our method uses low-Vth and high-Vdd and big-size only on highly-frequent critical cycles.

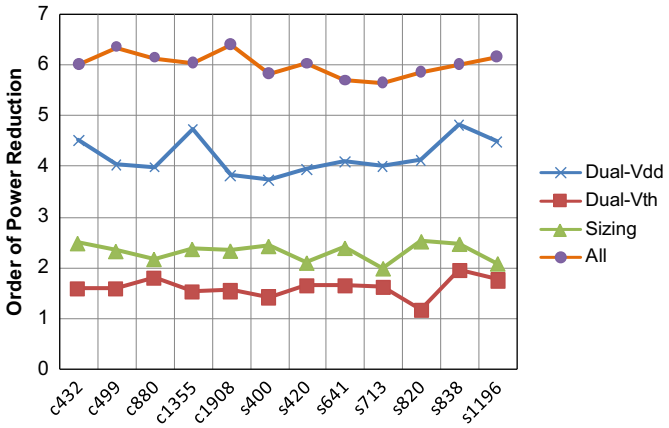

Fig. 11. Different technique comparisons.

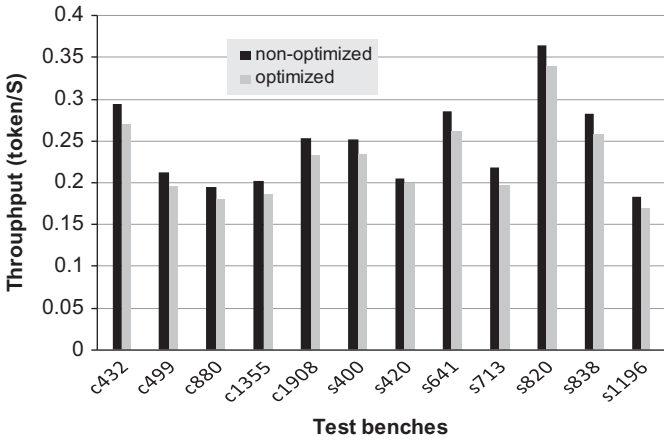

Fig. 12. Throughput (token/second) of optimized and non-optimized asynchronous circuits.

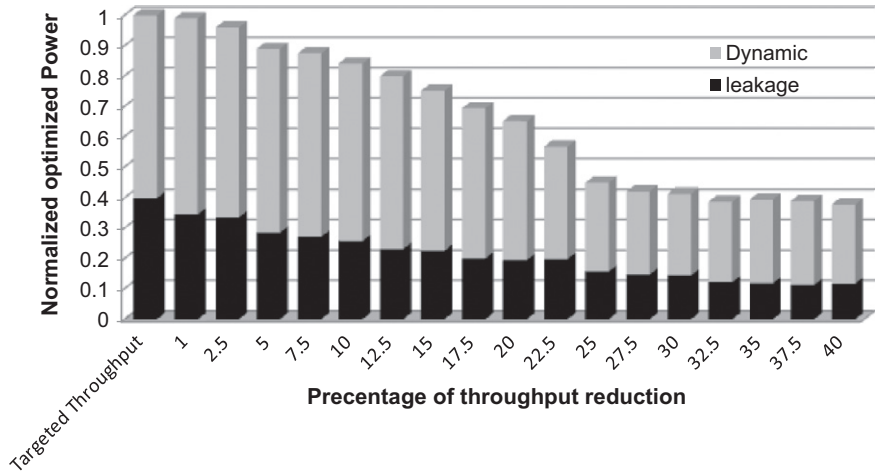

Fig. 10. Power and performance dynamics of proposed optimization flow.

Table 1
Comparison of average performance and worst-case performance optimization.

Modules of test-bench	Un-optimized		Worst-case optimization		Average-case optimization	
	Average throughput	Total power	Average throughput	Total power	Average throughput	Total power
Reed–Solomon	2.372 Gbps	16,804	2.838 Gbps	13,331	3.178 Gbps	19,997

9. Conclusion

An efficient design framework for optimizing reducing total power consumption of asynchronous circuits while maintaining the high performance of circuits is presented. The decomposed circuit is used to generate a Probabilistic Timed Petri-Net model which captures the dynamic behavior of the system. The proposed assigning threshold-voltage, supply-voltage and template sizing method is based on a quantum genetic algorithm. The experimental results on a set of ISCAS benchmark circuits show that our proposed technique can achieve on average 5X–7X savings for power consumption, while there is only 2.5% performance penalty in average. Our results demonstrate that our techniques can simultaneously lead to significantly lower power consumption and significantly smaller area when compared tour previous worst-case technique [14].

One advantage of our approach is that the parallel nature of quantum genetic algorithms suggests parallel processing as the natural route to explore. Implementation of a parallel version of this method is the one of our future works. An extension of our approach to consider the number of tokens in a circuit is an interesting area of future research.

References

[1] S.G. Narendra, A. Chandrakasan (Eds.), Springer, , 2005 ISBN 978-0-387-25737-2.
[2] D..Duarte, N. Vijaykrishnan, M.J. Irwin, H.S. Kim, G. McFarland, Impact of Scaling on the Effectiveness of Dynamic Power Reduction Schemes, in: Proceedings of International Conference on Computer Design (ICCAD), 2002, pp. 382–387.
[3] V. ortex superscalar processor. Fulcrum Corporation. 〈http://www.fulcrum micro.com/techchip〉gallery.html, 2001.
[4] C.G. Wong, Alain J. Martin, High-Level Synthesis of Asynchronous Systems by Data Driven Decomposition, in: Proceedings of Design Automation Conference (DAC), 2003.
[5] B. Ghavami, H. Pedram, M. Najibi, An EDA Tool for Implementation of Low Power and Secure Crypto-Chips, International Journal of Computers and Electrical Engineering (Elsivier Ltd.), to be published in, 2008.
[6] B. Ghavami, H..Pedram, Sub-Threshold Leakage Reduction of Asynchronous Pipelines Using Dependency—Graph Abstract Model, in: Proceedings of International Conference on Very Large Scale Integration (VLSI–SOC), pp. 2008.
[8] Steve Furber Jens Sparso, Principles of Asynchronous Circuit Design—A System Perspective, Kluwer Academic Publishers, 2002.
[9] Kimiyoshi Usami Mark Horowitz, Clustered Voltage Scaling Technique for Low-Power Design, in: Proceedings of International Symposium on Low Power Design (ISLDP), 1995 pp. 3–8.
[10] Q. Wang, S. Vrudhula, Algorithms for Minimizing Standby Power in Deep Submicron, Dual-Vt CMOS Circuits, IEEE Transactions on CAD (TCAD) 21 (2002) 306–318.
[11] J. Rabaey, A. Chandrakasan, B. Nikolic, Digital Integrated Circuits: A Design Perspective, Second Edition, Prentice Hall, NJ, 2003.
[12] M. Hamada, Y. Ootaguro, Utilizing Surplus Timing for Power Reduction, in: Proceedings of Custom Integrated Circuits Conference (CICC), 2001, pp. 89–92.
[13] B. Ghavami, H. Pedram, Design of Dual Threshold Voltages Asynchronous Circuits, in: Proceedings of International Symposium on Low Power Electronics and Design (ISLPED), 2008.
[14] B. Ghavami, M. Khosraviani, H. Pedram, Power Optimization of Asynchronous Circuits through Simultaneous Vdd and Vth Assignment and Template Sizing, in: Proceedings of EUROMICRO Conference on Digital System Design Architectures, Methods and Tools (DSD), 2008.
[15] A. Xie, P.A. Beerel, Performance Analysis of Asynchronous Circuits and Systems using Stochastic Timed Petri Nets, in: Proceedings of International Workshop on Hardware Design and Petri Nets, 1999.
[16] E. Labonne, G. Sicard, M. Renaudin, Dynamic Voltage Scaling and Adaptive Body Biasing Study for Asynchronous Design, Research Report, TIMA–RR-04/06-01—FR, 2006.
[17] B. Ghavami, H. Pedram, Dual-Threshold Voltage Technique for Asynchronous Pre-Charge Full Buffer Linear-Pipelines, 6th IEEE DACS, Dallas, USA, 2007.
[18] Raghid Shreih, Maitham Shams, Implementation of asynchronous pipeline circuits in multi-threshold CMOS technologies, in: Proceedings of Great Lakes Symposium on VLSI (GLSVLSI), 2008.
[19] R. Gupta, G.De Micheli, A Cosynthesis Approach to Embedded System Design Automation, Design Automation of Embedded Systems, vol. 1, Kluwer Academic Publishers, 1996.
[20] A. Kalavade, E. Lee, A Global Criticality / Local Phase Driven Algorithms for the Constraint Hardware/Software Partition Problem, in: Proceedings of the International Workshop on Hardware Software Codesign, 1994, pp. 42–48.
[21] N. Petrellis, A. Birbas, M. Birbas, E. Mariatos, Allocation of Hardware and Software Resources by Accessing IP or Standard Component Libraries,in: Proceedings of the International Workshop on IP Based Synthesis and System Design, December, 14–15, 1999, Grenoble, pp. 43–48.
[22] Braham Lane, SystemC Language Reference Manual, Copyright © 2003 Open SystemC Initiative, San Jose, CA.
[24] Arash Seifhashemi, Hossein Pedram, Verilog HDL, Powered by PLI: A Suitable Framework for Describing and Modeling Asynchronous Circuits at All Levels of Abstraction, in: Proceedings of Design Automation Conference (DAC), 2003.
[25] J.L. Peterson, Petri-Net Theory and the Modeling of Systems, Prentice-Hall, Englewood Cliffs, N.J, 1981.
[26] R.M. Karp, A characterization of the minimum cycle mean in a diagraph, Discrete Mathematics 23 (1978) 309–311.
[27] K.-H. Han, J.-H. Kim, Genetic Quantum Algorithm and Its Application to Combinatorial Optimization Problem, 2, IEEE Press, 2000 1354–1360.
[28] K.-H. Han, Quantum-inspired Evolutionary Algorithm, Ph.D. dissertation, Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology, Daejeon, Korea, 2003.
[29] B.J. Sheu, D.L. Scharfetter, P.K. Ko, M.C. Teng, BSIM: Berkeley Short-Channel IGFET Model for MOS Transistors, IEEE Journal of Solid-State Circuits SC-22 (4) (1987) 558–566.
[30] Srivastava, D. Sylvester, Minimizing Total Power by Simultaneous Vdd/Vth Assignment, in: Proceedings of Asia-South Pacific Design Automation Conference (ASPDAC), 2003, pp. 400–403.
[31] S.M. Burns, A.J. Martin, Performance Analysis and Optimization of Asynchronous circuits, Advanced Research in VLSI Conference, March 1991, Santa Cruz, CA.
[32] P.A. Beerel, N.H. Kim, A. Lines, M. Davies, Slack Matching Asynchronous Designs, in: Proceedings of International Symposium on Asynchronous Circuits and Systems (ASYNC), 2006.
[38] K. Saleh, Design and Implementation of an Asynchronous Forward Error Correction Circuit, MSC. Dissertation, Department of Computer Engineering, Amirkabir University of Technology, Tehran, Iran, 2004.