2015 IEEE 13th International Conference on Embedded and Ubiquitous Computing

# Automatic Design of Low-Power VLSI Circuits: Accurate and Approximate Multipliers

Vojtech Mrazek
Faculty of Information Technology,
Brno University of Technology,
Czech Republic
Email: imrazek@fit.vutbr.cz

Zdenek Vasicek
Faculty of Information Technology,
Brno University of Technology,
Czech Republic
Email: vasicek@fit.vutbr.cz

***Abstract*—In order to satisfy a constant need of reducing energy consumption of electronic devices, the approximate computing paradigm has been introduced in recent years. This paradigm is based on the fact that there are applications that are inherently capable of absorbing some errors in computation. Multimedia signal processing represents a typical example that allows for quality to be traded off for power.**

**Typicaly, the approximate circuits are designed at gate level. This paper introduces an automatic design method that is able to operate directly at transistor level which offers a great potential for discovering novel implementations of approximate circuits. The method combines a stochastic search algorithm with transistor-level circuit simulator and is able to handle the circuits consisting of hundreds of transistors. The goal of the search strategy is to improve the power consumption. To estimate power consumption, an algorithm based on transistor switching activity is proposed.**

**A design of 4-bit multiplier was chosen as a case study. Two scenarios were considered. Firstly, the proposed method is applied to improve the power consumption of a common 4-bit multiplier and a 4-bit multiplier consisting of manually designed 2-bit multipliers. In both cases, approx. 3% power reduction was achieved. Then, it is demonstrated that a noticeable improvement can be obtained when the multipliers are designed using a hybrid approach operating at transistor as well as gate level. We discovered a novel implementation of an approximate 4-bit multiplier which has approximately by 40% better power-delay product and exhibits 14% lower worst-case error compared to the best known 4-bit multiplier consisting of 2-bit manually optimized approximate multipliers.**

## I. Introduction

With the increasing degree of integration and scaling, the power consumption of VLSI systems has emerged as a pressing issue. The advances in technology enabled to place more transistors in the same area. As a consequence of that, mobile devices have evolved to very powerful computers. With more transistors, it was possible to add high-resolution cameras and exploit the information from a camera sensor in real-time. Advanced applications require high performance, yet must operate at low power to avoid draining the battery.

The power consumption can be reduced at different levels, such as the architectural, circuit, layout, and the fabrication process technology level. An obvious method to reduce power consumption is to shut down part of a circuit when it is not in operating conditions. The average power dissipation can be reduced also by reducing the switching activity of a given logic circuit. The switching activity can be reduced at the technology mapping phase and logic design phase [1]. However, a considerable potential for power saving exists at the circuit design level [2]. This is possible because all the important parameters governing the power dissipation, as transition activity, switching capacitance, and short-circuit currents, are strongly influenced by the chosen logic style such as conventional or complementary CMOS and various variants of pass-transistor logic.

In recent years, an approximate computing was established to investigate how computer systems can be made better, i.e. more energy efficient, faster, or less complex. Approximate computing exploits the fact that some applications (e.g. in multimedia processing) are inherently error resilient due to the limited human perception capabilities [3]. Two main methodologies have been used to achieve higher power efficiency. Voltage over-scaling, addressed in early works (see e.g.[4]), and logic approximation in which the Boolean functions of the underlying cores in computing circuits, such as adders and multipliers, are approximated by less complex implementations.

In order to avoid manual modification of accurate circuits, systematic methods capable of performing approximations have been introduced recently [5], [6], [7]. These methods typically start with a gate-level description of the accurate circuit and an error constraint that specifies the type of error that can be accepted. Various error criteria are used to evaluate the quality of an approximate arithmetic circuit. The average error magnitude, defined as the sum of absolute differences in magnitude between the original and approximate circuits averaged over all inputs, represents the most common error criteria.

The goal of this paper is to introduce and evaluate an approach that is able to optimize the VLSI circuits directly at transistor level. To the best of our knowledge, this is the first work addressing the problem of power efficiency optimization that introduces a systematic design method which operates directly at transistor-level netlists. Our work is motivated by the fact that a great potential in power savings can be achieved at the level of MOS transistors. Various logic styles, for example, could be combined at this level which may leads to novel and efficient circuit structures.

As we move down to the transistor-level implementation, however, many new challenges arise because of the increas-

978-1-4673-8299-1/15 $31.00 © 2015 IEEE
DOI 10.1109/EUC.2015.20

ing complexity. The widely applied approach to power and delay calculation employing analog circuit simulators cannot routinely be applicable because simulation runtime begins to be critical even for smaller circuits. One way to overcome this problem is to introduce heuristic approaches operating in symbolic domain. In this work, we propose to combine functional equivalence checking with a new method for power consumption estimation.

Because the design of low-power variants of key arithmetic circuits such as adders and multipliers represent the intensively studied area, especially in context of approximate computing (see e.g. [3], [8], [9]), we suggest to evaluate the proposed method using them. In particular, power-aware optimization of an accurate (i.e. fully working) as well as approximate versions of a 4-bit multiplier was chosen as a case study. The 4-bit multiplier was chosen because the accurate implementation consists of more than four hundreds of transistors which do not represent a trivial case.

## II. Proposed methodology

To achieve our goal, i.e. to optimize the power consumption of an existing circuit, the circuit is firstly compiled (synthesized) to a gate-level netlist. Then the netlist is converted to the transistor-level representation using the standard cell library for ASICs. Finally, the circuit is encoded using an internal representation which is designed to perform the circuit transformations efficiently. The encoded original circuit is gradually modified using a set of transformation operators. To explore the search space efficiently, an iterative stochastic search algorithm is used. This algorithm produces a set of various modifications of the original circuit. The search process is guided by a user-defined objective function that assigns a fitness score to each candidate circuit that was obtained by applying a transformation to a parental individual. The objective function has to be constructed to meet the design goal. In our case, only the power consumption is reflected in the objective function.

It is well known fact that the quality of solutions discovered by the stochastic search algorithm is proportional to the number of explored design alternatives. The usage of HSPICE for verification, delay and power consumption calculation is, however, a very time-consuming process. The runtime varies from a few minutes to several hours depending on the size of the circuit and the required precision. Therefore, a functional transistor-level equivalence checking algorithm in combination with a probabilistic approach for the power estimation is proposed. The combination of these techniques helps to avoid a need of executing the time-consuming HSPICE simulation in the each iteration of the search algorithm. In addition to that, this approach helps to prevent the limited portability of the discovered implementations. When we tried to use only the HSPICE simulator for functional verification of the candidate circuits, we observed that the final implementations usually were closely linked to the fabrication technology that was chosen to evaluate the correct function of the design alternatives. Mostly, an implementation was discovered that worked perfectly for 180nm process, however, when a different technology was used (e.g. $.35\mu\mathrm{m}$ or $90\mathrm{nm}$), the measured output response was erroneous. The malfunctions were caused mainly by the presence of nets with floating value. A combination of logic values at primary inputs caused that some control signals (i.e. signals that are connected to a gate electrode of a transistor) left undriven. In that case, parasitic capacitances of the driven transistors could be misused to accomplish a correct function of the whole circuit. This issue is illustrated in Figure 1. If the proposed transistor-level equivalence checking algorithm is applied, these situations could easily be detected and eliminated.

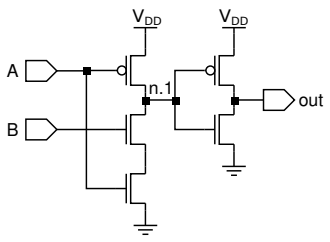


Fig. 1. A structure (formerly AND gate) that was utilized in an optimized variant of a 2-bit multiplier. The correct function was validated by means of a simulation conducted by HSPICE using $180\mathrm{nm}$ TSMC process. When $V_{DD}$ is applied to the input $A$ and input $B$ is tied to ground, signal $n.1$ is left undriven. The voltage at $n.1$ depends on the previous state of $n.1$ and its waveform is determined by the parasitic capacitances of the gate electrodes of n-MOS as well as p-MOS transistors that are driven by $n.1$. Such a structure is unwanted bacause of the limited portability across various technology processes.

The proposed method works as follows. First, the functional transistor-level equivalence checking is performed for the design alternatives (i.e. circuits modified by the transformation operators). Only the circuits that passed this test, i.e. those implementing the same logic function as the original circuit, are accepted and evaluated for the power consumption.

### A. Design Space Exploration

In order to efficiently explore the search space corresponding to the proposed circuit encoding, the iterative stochastic algorithm known as *$(1 + \lambda)$ evolutionary strategy*, described in Algorithm 1, was applied. The principle is to gradually improve the initial design by executing multiple iterations of transformations. The initial solution for the population oriented evolutionary strategy is initialized by an original design that ought to be optimized. In each iteration, a population of candidate circuits consisting of the best actual design (a parent) and $\lambda$ design alternatives (offspring) generated from the parent is evaluated. The design alternatives are created using a set of randomly generated transformations. The search is guided by the fitness function which determines how good a particular design is. In the case when two or more offspring have received the same fitness score in the previous population, the individual that did not serve as a parent in the previous population is selected as a new parent. This strategy is used to ensure the diversity of the population. The evolutionary process is terminated if a given number of iterations (denoted as $N_G$) is achieved or a required solution is found.

For the reason of simplicity, a single-objective optimization is considered. The fitness value of each candidate circuit is equal to the estimated value of power consumption. The goal is to minimize the value of the objective function. Due to various assumptions and simplifications, the power estimation as well as the equivalence checking algorithm has a limited accuracy.

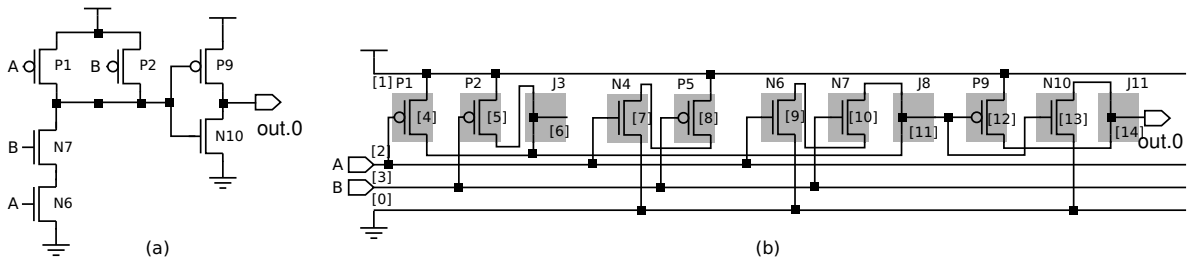


Fig. 2. (b) Example of a candidate circuit encoded using $n$=11 three-terminal nodes that implements (a) Boolean function AND consisting of 6 transistors. Parameters are as follows: $n_i$=4 (i.e. four primary inputs are used; the first two primary inputs are dedicated to the power supply rails; the second and third input correspond with input variable $A$ and $B$), $n_o$=1 (i.e. the circuit has a single primary output denoted as $out.0$). The circuit is encoded as *(1, 2, pmos)(1, 3, pmos)(4, 5, junction)(0, 2, nmos)(7, 3, pmos)(0, 2, nmos)(9, 3, nmos)(10, 6,junction)(1, 11, pmos)(0,11, nmos)(13, 12, junction)(14)*. Each triplet encodes configuration of a single node. For example, the fifth triplet $(7, 3, pmos)$ encodes the configuration of node P5 – the first terminal of P5 is connected to a signal labeled as 7 (drain of node N4), the second terminal is connected to a signal labeled as 3 (primary input B), and the node behaves as p-MOS transistor. The last number encodes the connection of the primary output. Note that the nodes N4 a P5 are redundant because they do not drive any node.

For example, a conductive path between the supply voltage and ground may temporarily occur. This path, however, does not necessarily have to be visible to the steady-state analysis. To address this problem and to mitigate potentially invalid design alternatives, HSPICE simulator is executed regularly (line 17–23 of Algorithm 1).

### B. Circuit representation and transformation operators

Each circuit having $n_i$ primary inputs, $n_o$ outputs and consisting of $N$ transistors is represented using $n$ nodes ($n \geq N$) arranged as a one-dimensional array. Each node has three terminals – two input terminals and one output terminal. The input terminals can independently be connected either to the output terminal of a node placed in previous columns or to one of the primary circuit inputs. Each node can act as a wire, junction, n-MOS transistor or p-MOS transistor. The wire node connects the first input terminal with the output terminal. Junction node is able to combine two input signals and one output signal together. As a consequence of that, loops and multiple connections are natively supported.

**Input**: original design
**Output**: optimized design
1 Let $C_1$ = original design, $T$ = 1;
2 **while** $i < N_G$ **do**
3  **while** $j < \lambda$ **do**
4  do pick $\gamma$ transformation operations at random;
5  apply the transformations to $C_i$ to yield $O_j$;
6  do perform a functional equivalence;
7  **if** $O_j$ *is valid* **then**
8  use switching activity from step 6 to estimate the power consumption $P_j$;
9  let the fitness $F_j = P_j$
10  **else**
11  $F_j = -1$
12  **end**
13  **end**
14  determine $O_k$, $k \in \{1, \ldots, \lambda\}$ having the best $F_k$;
15  **if** $O_k$ *is better or equal than* $C_i$ **then**
16  let $C_{i+1}$ = $O_k$;
17  **if** $T - i >$ *limit* **then**
18  run HSPICE to perform in-depth simulation;
19  **if** $C_{i+1}$ *is worse than* $C_T$ **then**
20  replace $C_{i+1}$ with $C_T$;
21  **end**
22  let $T$ = $i$;
23  **end**
24  **else**
25  let $C_{i+1}$ = $C_i$;
26  **end**
27 **end**

**Algorithm 1:** Design space exploration algorithm

The following encoding scheme is utilized. The primary inputs and node outputs are labeled $0, 1, \ldots, n_i - 1$ and $n_i, n_i+1, \ldots, n_i+n-1$, respectively. A candidate solution is represented by $n$ triplets $(i_1, i_2, f)$ determining for each node its function $f$ (i.e. whether it acts as wire, junction, n-MOS, or p-MOS), and label of nodes $i_1$ and $i_2$ that are connected to the source pins. In addition to that, a tuple consisting of $n_o$ indices is utilized to specify the indices of nodes where the primary outputs are connected to. Note that the first two primary inputs are reserved for power supply rails.

Figure 2 demonstrates the principle of proposed encoding on a AND circuit implemented using CMOS logic. In this example, a flexibility of the proposed encoding scheme is shown. A candidate circuit is encoded using 11 nodes, however, only some of them are active and contribute to the final netlist. The activity of a node is determined as follows. A node is active if its output is (a) connected to any of the primary outputs or (b) to the input of an active node. According to this definition, node 4 and node 5 represent inactive nodes.

Two transformation operators are considered in this work. The first operator can change function of a node to a different one. The second operator can change connection of a single terminal of a chosen node to a different node or a primary input. Note that only transistor nodes are allowed to be connected to the primary inputs.

### C. Transistor-level functional equivalence

The transistor-level equivalence checking (mentioned in line 6 of Algorithm 1) performs checks between the original (i.e. reference) design and its transistor-level implementation. The objective is to determine whether the transistor level implementation captures the same Boolean function. Even if

the equivalence checking can be performed directly at the transistor-level, it is better to have the reference design at the gate-level.

We implemented a simplified version of a switch-level circuit simulator IRSIM [10] to address the problem of the equivalence checking. The transistor-level circuit is simulated using a multi-level discrete event-driven simulator which is utilized to determine the response for each input combination. The calculated output values are compared to the truth table derived from the gate-level description of the reference circuit. To be able to model various degradations such as threshold drop effect, the simulator operates on eight discrete levels: 0 (logic zero), $L$ (degraded logic zero), $l$ (double-degraded logic zero), 1 (logic one), $H$ (degraded logic one), $h$ (double-degraded logic one), $Z$ (high-impedance state), and $X$ (invalid value).

It is clear that this technique cannot be applied on large circuits due to the exponential dependency between the number of primary inputs and the number of simulation steps. However, even if the proposed circuit simulator does not scale well, it is very fast considering our target application. According to the experimental evaluation, this approach is more than three orders faster than executing HSPICE. For larger designs, i.e. circuits having more than about ten primary inputs, we recommend to employ a circuit checker operating in symbolic domain. For example, Bryant et al. proposed a symbolic approach designed for verification of transistor circuits in the MOS technology [11]. The advantage of this approach is that it preserves the generality and accuracy as the switch-level simulation. It can capture effects of bidirectional transistors, stored charge, and multiple signal strengths.

### *D. Power consumption estimation*

One of the most accurate and straight-forward method for the power estimation is to perform a circuit simulation by means of a SPICE simulator. However, it was shown that the simulation results are usually strongly pattern-dependent [12]. Hence large numbers of the input patterns would have to be simulated. This can become computationally very expensive, especially for large circuits.

In order to avoid the time-consuming exhaustive simulation, we propose to use a probabilistic method for active mode power estimation (see line 8 of Algorithm 1). The problem of estimating the power consumption can be reduced to the task of computing steady-state transition probabilities. To estimate the power consumption, we generalized the approach introduced in [13]. The method assumes that the energy dissipation of a CMOS circuit is directly related to the switching activity. Despite the increasing importance of the static power caused by the technology scaling, the assumption seems to be valid even for modern technology processes [12], [14].

The switching activity is determined according to the signal and transition probabilities. The signal probability $P_n(x = V)$ at a signal $x$ is defined as the average fraction of clock cycles in which the steady state value of $x$ is a equal to the logic value $V \in \Lambda$, where $\Lambda$ is a set of possible logic values. In our case, six logic values are considered: logic low, logic high and two degraded variants for each logic value, i.e. $\Lambda = \{0, 1, L, H, l, h\}$. Let state $S(x) = (g, sd)$ of a transistor

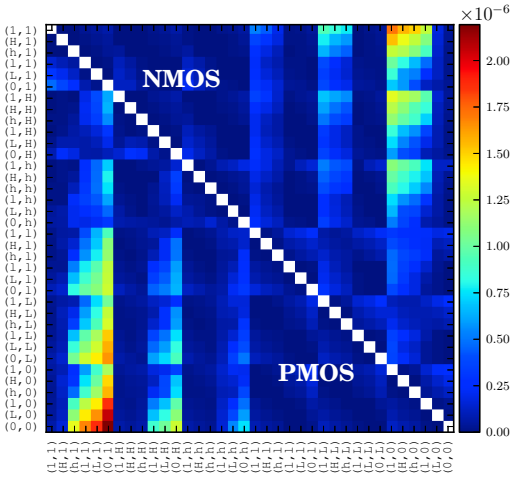


Fig. 3. The average power consumption of n-MOS (p-MOS) transistor when its state represented as (g,sd) changed to state (g',sd') at 100MHz and 180nm.

node $x$ be defined by the actual logic values present at the signals corresponding to its three terminals, where $g, sd \in \Lambda$. Assuming that a circuit is valid, i.e. no short-circuits can occur, then $g$ and $sd$ can be determined as follows. In case that the source and drain terminal have the same polarity, i.e. they are both positive (negative), the value of $sd$ is determined by the stronger value and $g$ is equal to the logic value of a signal connected to the gate terminal. Otherwise, if the transistor is closed, $sd$ is determined as the lowest logic value (highest logic value in case of p-MOS) and $g$ is calculated as the complement of the actual logic value at a signal connected to the gate. The latter equation tries to model a situation when an open transistor starts to discharge the load capacitance.

Given a state $S(x)$, we can calculate a transistor state probability $P_s(S(x) = y)$ for each $y \in (a, b)$ defined as the average fraction of the clock cycles in which a transistor node $x$ remained in the state $S(x)$, where $a, b \in \Lambda$. Note that the state probabilities can be calculated simultaneously with the circuit simulation which is used to perform the functional equivalence checking. Hence, no additional computational overhead is introduced. Let us assume that primary inputs are uncorrelated and that they are changing instantaneously at global clock edges. Then, the transition probability $P_{tr}^{A \to B}(x) = P_{tr}(S(x)^t = A \wedge S(x)^{t+1} = B)$ of a node $x$ defined as the average fraction of clock cycles in which the state of the node $x$ changed from $A$ to $B$ at a given point $t$ in time can be expressed as $P_{tr}(S(x)^t = A \wedge S(x)^{t+1} = B) = P_s(S(x)^t = A) \cdot P_s(S(x)^{t+1} = B)$. It means that it is not necessary to determine the conditional probabilities. This simplification can be introduced thanks to the fact that the simulation-based approach inherently takes into account the correlation caused at internal nodes in the circuit due to reconvergence of the input signals or reconvergent fan-out.

Given the transition probabilities, the power consumption can be calculated as:

$$Pwr_{est} = \sum_{\forall x} \sum_{A,B \in \Lambda \times \Lambda} C_{load}(x) Pwr(A, B) P_{tr}^{A \to B}(x),$$

where $C_{load}(x)$ is a constant related to the load capacitance being charged/discharged of a transistor node $x$ and $Pwr(A, B)$ is a constant related to the power consumed by a transistor running at frequency $f$ when its state changed from $A$ to $B$. Note that $Pwr$ is a technology dependent factor that have to be characterized in advance using a HSPICE simulator for n-MOS and p-MOS transistors.

The average power consumption measured using HSPICE for zero-body-biased MOS transistors operating at the nominal supply voltage (VDD = 1.8 V) and $f = 100$ MHz for 180nm TSMC process and $C_{load} = 7.2$ fF, corresponding with load capacitance of an inverter, is shown in Figure 3. It can be observed that MOS transistors consume highest power when they completely charge or discharge the load capacitances, i.e. when $sd$ changes from 0 to 1 and vice versa. If a transistor is open, then the power consumption increases proportionally with the gradient between $sd$ of the actual and subsequent state. Unfortunately, the dependence between the power consumption and the gradient is nonlinear. Note that only one half of the total number of transitions have to be stored in memory due to the diagonal axis of symmetry, i.e. $Pwr(A, B) = Pwr(B, A)$.

### *E. Delay and Power Evaluation*

The power consumption and delay parameters of the discovered design alternatives are measured using HSPICE simulator. A cascade of two inverters is applied on each primary input in order to feed the circuit with more realistic signal waveforms. Each primary output is loaded with a capacitive load which is modeled using two transistors. The circuit structure used to evaluate the power consumption is shown in Figure 4. Although HSPICE is equipped with a built-in power estimation command, it has been shown that the obtained results can introduce significant errors [12]. Hence, to accurately estimate the power consumption, four independent voltage sources $V_{supply}$, $V_{gnd}$, $V_{p-bias}$ and $V_{n-bias}$ are used to supply the transistors. In addition to that, a dummy voltage source ($V_{dummy_i}$) is connected to each primary output $OUT_i$ to measure the output current $I(V_{dummy_i})$. The power consumed by the circuit with $PI$ inputs and $PO$ outputs is defined as

$$\text{Average Power} = \frac{1}{t}\int_0^t P_{inst} \cdot dt,$$

where the instantaneous power $P_{inst}$ is measured as

$$P_{inst} = -P(V_{supply}) - P(V_{p-bias}) - P(V_{gnd}) - P(V_{n-bias}) - \sum_{i=1}^{PI} P(V_{IN_i}) - \sum_{j=1}^{PO} I(V_{dummy_j})V(OUT_j)$$

The input signals are generated using a LFSR voltage source generators and the simulation is performed using $N$ patterns, where $N = 8 \cdot 2^{PI} \cdot 10^4$. This method was chosen to reduce the pattern-dependency effect. Since the LFSR signal generators operate at 100 MHz, one input pattern corresponds to 10 ns of simulation.

The similar approach is applied during the search space exploration (line 18 of Algorithm 1). The reduced number of patterns ($N = 20 \cdot 2^{PI}$) is used, however. The goal of this step is to apply a set of random samples to verify that the current design alternative $C_{i+1}$ performs the required logic function. In addition to that, it is necessary to ensure that the design alternative $C_{i+1}$ does not have its power consumption worse than the previously evaluated solution $C_T$.

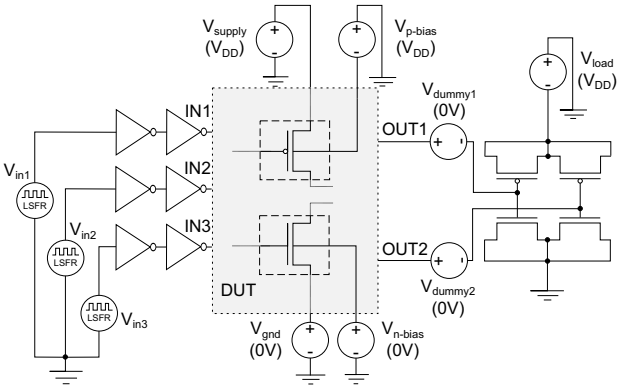


Fig. 4. Principle of the power measurement using independent voltage sources

The delay of a discovered circuit is determined using the piecewise-linear signal generators with the rise and fall times equal to 5 ps. The input signals are designed to produce a subset of all different transitions from an input pattern to another one. In total, $50 \cdot 10^3$ different transitions are generated. The delay is measured from the moment the input signal reaches the primary inputs till the latest of the output values reaches the primary output. The primary outputs are loaded with a capacitive load created using two transistors.

## III. Experimental Results

### *A. Accuracy of the power consumption estimation algorithm*

Firstly, we evaluated the power estimation heuristics proposed in Section II-D. In our work, we apply the power estimation heuristics within transformational design space exploration. Hence the ability to quantify relative dependencies of the design power consumption is much more important than the ability to capture absolute values. For comparison of the values it is only needed to achieve a high *fidelity value* [15].

Let $R$ be a set of $n$ reference values and $E$ be a set of $n$ estimated values. Let $T$ be a set of $n$ benchmark circuits that were used to calculate $R_i$ and $E_i$. The fidelity describing the quality of the estimate with respect to its ability to quantify relative dependencies of the tuples reference/estimation values is defined as

$$\text{Fidelity} = 100\ \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \mu_{ij}, \tag{1}$$

where $\mu_{ij}$ is determined as

$$\mu_{ij} = \begin{cases} 1, & \text{if } \begin{cases} R_i > R_j \wedge E_i > E_j \\ R_i = R_j \wedge E_i = E_j \\ R_i < R_j \wedge E_i < E_j \end{cases} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

The fidelity for a set of arithmetic benchmark circuits is reported in Table I. For each benchmark circuit $C$ 500 design alternatives generated during the design space exploration were included in $T_C$. The power consumption ($E_i$) was estimated using the method described in Section II-D. The reference values ($R_i$) were calculated using HSPICE simulator and 180 nm technology.

TABLE I. FIDELITY OF THE PROPOSED POWER ESTIMATION APPROACH EVALUATED FOR VARIOUS BENCHMARK CIRCUITS

| | | | # transistors | | power (uW) | | |
|---|---|---|---|---|---|---|---|
| **Circuit** | **PI** | **PO** | **min** | **max** | **mean** | **dev.** | **fidelity** |
| 1b half adder | 2 | 2 | 9 | 20 | 50.4 | 1.1 | 99 % |
| 1b full adder | 3 | 2 | 14 | 48 | 99.4 | 20.7 | 85 % |
| 2b half adder | 4 | 3 | 24 | 88 | 217.1 | 7.5 | 97 % |
| 2b full adder | 5 | 3 | 47 | 51 | 169.6 | 2.9 | 74 % |
| 2b multiplier | 4 | 4 | 33 | 54 | 72.5 | 4.3 | 87 % |
| 2b multiplier approx. | 4 | 4 | 24 | 30 | 49.6 | 1.3 | 84 % |
| 4b multiplier | 8 | 8 | 407 | 490 | 553.1 | 44.9 | 98 % |
| 4b multiplier approx. 1 | 8 | 8 | 178 | 216 | 297.2 | 22.9 | 97 % |
| 4b multiplier approx. 2 | 8 | 8 | 350 | 378 | 387.4 | 5.4 | 87 % |
| 4b multiplier approx. 3 | 8 | 8 | 324 | 350 | 454.5 | 9.9 | 100 % |

It can be observed that the fidelity is $74\%$ in the worst case. It means that the estimated values correlate with HSPICE results in $185 \cdot 10^3$ out of $250 \cdot 10^3$ cases. If only the larger circuits are considered (i.e. 4-bit multipliers), the fidelity is not worse than $87\%$ and the average fidelity is $91\%$. Unfortunately there is no clear dependency between fidelity and either average power, its standard deviation, or the number of transistors employed by a design alternative.

### *B. Design of low-power multipliers*

We have implemented the proposed approach and evaluated its performance in the design of common 2-bit and 4-bit arithmetic circuits such as adders and multipliers. These circuits were chosen because they are widely used in digital signal processing. Hence a small improvement in the performance of these key circuits may yield in a large power saving [3], [16]. In addition to that, it is possible to compare our results with the state-of-the-art implementations.

The goal was to optimize the power consumption of various existing designs. Due to the limited space, three experiments will be presented in this paper: (1) optimization of an exact 4-bit multiplier, (2) optimization of the best known approximate 4-bit multiplier, and (3) automatic design of approximate 4-bit multiplier. While the first two experiments were chosen to demonstrate ability of the proposed algorithm to improve already optimized designs, the latter experiment was chosen to investigate whether the stochastic search is able to produce solutions that are able to compete with the best known one.

The experiments were conducted on 180 nm TSMC process with 1.8 V supply voltage. To evaluate the portability of the discovered implementations, the final netlists were analyzed in 45 nm, 90 nm, 250 nm, and 350 nm technology process. Only the circuits exhibiting a full voltage swing on the outputs were accepted during the design space exploration. This restriction ensured that the discovered implementations may be used as a basic building block in larger multipliers. The transistor-level implementations were converted from the optimized gate-level description. Each gate was replaced using a common CMOS implementation from ASIC library. Note that we do not consider sizing of transistors because this can be done afterwards and in iterative manner by identifying the transistors in the critical path [14]. It means that the results presented in following sections can be furthermore improved if the sizing is applied. Power consumption estimated using the proposed algorithm is employed to determine the fitness value. HSPICE was executed every hundred iterations to verify the functionality of the latest candidate design alternative. More than $80 \cdot 10^3$ randomly generated input vectors were used to perform this task.

For each problem, three independent runs of the proposed algorithm were executed in parallel. The following settings were utilized: $\lambda = 5$, $\gamma = 5$. The runs were executed for 375 000 iterations each. The fixed number of iterations enabled us to limit the runtime of the optimization. The experiments were conducted on Intel Xeon E5-2630 running at 2.30 GHz. The time of the optimization of the 4-bit multipliers is less than 82 hours. To improve the convergence of the search algorithm, the best solution was determined every 2 500 iterations. The best design was shared among all three runs and runs that did not yet achieve the same or better reduction in power consumption were reinitialized with the best design.

Finally, ten results with the highest fitness score were chosen to be evaluated for the power consumption and delay using a large number of transitions as described in Section II-E. Three results having the lowest power consumption were evaluated under various technology processes (45 nm, 90 nm, 180 nm and 250 nm) in order to analyze whether the netlists are technology dependent. The power consumption is measured for input signal operating at 300 MHz (250 nm), 500 MHz (180 nm), 1 GHz (90 nm) and 1.3 GHz (45 nm). The SPICE netlists of the discovered implementations presented in this paper can be downloaded at http://www.fit.vutbr.cz/~imrazek/euc2015.

*1) Optimization of an exact 4-bit multiplier:* The goal of the first experiment was to optimize a common 4-bit multiplier. We chose a compact implementation whose gate-level description consists of 59 two-input gates and exhibits delay of 15 logic gates. When converted to the transistor-level, the multiplier contains 444 transistors.

The parameters of the original implementation and the three best discovered alternatives are summarized in Table II. The total power consumption, worst-case delay and power-delay product (PDP) are included. The best value in each column is emphasized in bold. If we focus only on the results obtained using 180 nm TSMC process, it can be concluded that the proposed method was able to improve the total power consumption in all cases. Circuit OPT3 has about 4% lower power consumption compared to the original CMOS implementation. Situation regarding delay, however, is completely different. Only the first variant labeled as OPT1 exhibits 7% improvement. Overall, OPT1 seems to provide the best results if PDP is considered.

It is evident that the proposed method is able to pro-

duce various design alternatives having noticeably different parameters. Interestingly, we have obtained results with an improvement in delay even if it was not specified in the fitness function. If we construct a pareto front, OPT1 and OPT3 represent two alternatives that are worth to implement. While design alternative OPT3 is fast, OPT1 consumes less power.

TABLE II. POWER, DELAY AND POWER-DELAY PRODUCT FOR ACCURATE 4-BIT MULTIPLIERS AND DIFFERENT TECHNOLOGY PROCESSES

| | **Total power** ($10^{-4}$) | | | | **Delay** ($10^{-9}$) | | | | **PDP** ($10^{-13}$) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 250 | 180 | 90 | 45 | 250 | 180 | 90 | 45 | 250 | 180 | 90 | 45 |
| ORIG | 14.44 | 7.95 | 3.43 | 1.46 | 2.18 | 1.69 | 0.64 | 0.70 | 31.48 | 13.43 | 2.19 | 1.02 |
| OPT1 | 14.29 | 7.87 | 3.38 | 1.45 | **2.04** | **1.57** | **0.58** | **0.70** | **29.15** | **12.36** | **1.96** | **1.01** |
| impr. | 1.0% | 0.9% | 1.5% | 1.0% | 6.4% | 7.1% | 9.4% | 0.0% | 7.4% | 8.0% | 11% | 1.0% |
| OPT2 | 13.91 | 7.65 | 3.34 | **1.41** | 2.34 | 1.78 | 0.70 | 0.93 | 32.55 | 13.62 | 2.34 | 1.32 |
| impr. | 3.7% | 3.7% | 2.5% | 3.3% | -7.3% | -5.3% | -9.4% | -33% | -3.4% | -1.4% | -6.7% | -28% |
| OPT3 | **13.86** | **7.63** | **3.33** | **1.41** | 2.33 | 1.75 | 0.70 | 0.93 | 32.29 | 13.36 | 2.33 | 1.31 |
| impr. | 4.0% | 4.0% | 2.8% | 3.8% | -6.9% | -3.6% | -9.4% | -33% | -2.6% | 0.6% | -6.3% | -28% |

A relative stable reduction in power consumption was achieved if different technology processes are considered. This result indicates that the obtained implementations are relative robust. The delay, however, is very sensitive to the technology process. Only 1% improvement was achieved at 45 nm. It is necessary to note, however, that this result was expected because at lower technology nodes (45nm and below) the leakage current in active mode becomes almost comparable to switching currents. If our goal was to obtain better results for 45 nm technology, it would be necessary to optimize a given circuit using the target (i.e. 45 nm) technology.

*2) Optimization of an approximate 4-bit multiplier:* In the second experiment, optimization of an approximate 4-bit multiplier introduced in [16] is considered. The multiplier is constructed using an inaccurate 2-bit manually designed and optimized multiplier. The circuit consists of 49 gates and exhibits delay of 11 logic gates, 3.125% mean error and 22.22% worst-case error. When represented using the CMOS logic, 360 transistors are required. Note that a new architecture with significantly lower error was introduced recently [17], however, the number of gates required to implement 4-bit approximate multiplier is noticeably higher.

TABLE III. POWER, DELAY AND POWER-DELAY PRODUCT FOR APPROXIMATE 4-BIT MULTIPLIERS AND DIFFERENT TECHNOLOGY PROCESSES

| | **Total power** ($10^{-4}$) | | | | **Delay** ($10^{-9}$) | | | | **PDP** ($10^{-13}$) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 250 | 180 | 90 | 45 | 250 | 180 | 90 | 45 | 250 | 180 | 90 | 45 |
| ORIG | 12.26 | 6.87 | 2.90 | 1.25 | 1.76 | 1.34 | 0.52 | 0.47 | 21.58 | 9.21 | 1.51 | 0.59 |
| OPT1 | 12.09 | 6.79 | 2.86 | 1.24 | **1.51** | **1.16** | **0.47** | **0.47** | **18.26** | **7.87** | **1.35** | **0.58** |
| impr. | 1.4% | 1.2% | 1.4% | 1.4% | 14% | 13% | 9.6% | 0.0% | 15% | 15% | 11% | 1.4% |
| OPT2 | **12.12** | **6.81** | **2.88** | **1.25** | 1.52 | **1.16** | **0.47** | **0.47** | 18.42 | 7.90 | 1.35 | 0.59 |
| impr. | 1.1% | 0.9% | 0.9% | 0.6% | 14% | 13% | 9.6% | 0.0% | 15% | 14% | 10% | 0.6% |
| OPT3 | 12.15 | 6.82 | 2.89 | 1.25 | 1.57 | **1.16** | **0.47** | **0.47** | 19.08 | 7.91 | 1.36 | 0.59 |
| impr. | 0.9% | 0.7% | 0.6% | 0.2% | 11% | 13% | 9.6% | 0.0% | 12% | 14% | 10% | 0.2% |

The parameters of the original approximate multiplier and the three best discovered alternatives are summarized in Table III. In contrast with the first experiment, the improvement in power consumption is not greater than 1.5%. Nevertheless, the worst-case delay was improved significantly. Circuit OPT1 is approx. 13% faster than the original version. As a consequence of that, 15% reduction in PDP was achieved.

The power and delay reduction is relatively stable for different technology processes with an exception of implementations at 45 nm process having a small PDP improvement.

*3) Automatic design of approximate 4-bit multipliers:* The objective of the third experiment was to investigate whether it is possible to automatically design an alternative implementation exhibiting significantly better power and delay parameters compared to the manually created multiplier evaluated in the previous section. To accomplish this task, we used a hybrid approach based on two-level optimization which combines a gate-level optimizer for the approximate circuits proposed in [18] with the transistor-level optimizer introduced in this paper.

First, we employed the gate-level optimizer that was initialized with the gate-level representation of the exact 4-bit multiplier utilized in the first experiment. The goal of the optimizer was to modify the original circuit to obtain an approximate 4-bit multiplier having 3% mean error and consisting of the lowest possible number of gates (i.e. the power consumption is optimized indirectly). We used the same experimental setup as it was used in [18]. The optimizer discovered an approximate 4-bit multiplier consisting of 30 gates having delay of 10 gates, mean-error equal to 3.140% and worst-case error equal to 8%.

TABLE IV. POWER, DELAY AND POWER-DELAY PRODUCT FOR APPROXIMATE 4-BIT MULTIPLIERS OPTIMIZED USING HYBRID APPROACH

| | **Total power** ($10^{-4}$) | | | | **Delay** ($10^{-9}$) | | | | **PDP** ($10^{-13}$) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 250 | 180 | 90 | 45 | 250 | 180 | 90 | 45 | 250 | 180 | 90 | 45 |
| ORIG | 8.61 | 4.82 | 2.02 | 0.87 | 1.63 | 1.16 | 0.47 | 0.47 | 14.03 | 5.60 | 0.95 | 0.41 |
| OPT1 | 7.58 | 4.25 | 1.78 | 0.77 | **1.51** | **1.16** | **0.47** | **0.47** | 11.44 | 4.93 | 0.84 | **0.36** |
| impr. | 12% | 12% | 12% | 12% | 7.4% | 0.0% | 0.0% | 0.0% | 18% | 12% | 12% | 12% |
| OPT2 | **7.54** | **4.22** | **1.77** | **0.77** | **1.51** | **1.16** | 0.52 | **0.47** | **11.38** | **4.89** | **0.92** | **0.36** |
| impr. | 12% | 13% | 12% | 12% | 7.4% | 0.0% | -11% | 0.0% | 19% | 13% | 2.8% | 12% |
| OPT3 | 7.55 | 4.25 | 1.78 | 0.77 | 1.63 | 1.19 | 0.58 | **0.47** | 12.30 | 5.06 | 1.03 | **0.36** |
| impr. | 12% | 12% | 12% | 11% | 0.0% | -2.6% | -23% | 0.0% | 12% | 9.6% | -8.9% | 11% |

In the second step, the discovered design was optimized using the transistor-level optimizer. Parameters of the approximate multiplier discovered using the gate-level optimizer (denoted as ORIG) as well as the variants optimized using the proposed approach (denoted as $\mathrm{OPT}_n$) are given in Table IV. The transistor-level optimizer was able to improve the power consumption as well as the PDP of the initial implementation by 13%.

Compared to the improvements presented in Table II and Table III, a huge power consumption reduction was achieved by combining gate-level and transistor-level approaches. The obtained results suggest that our hypothesis about the existence of a great potential in power reduction at transitor-level seems

to be valid. The proposed transistor-level optimizer was able to substantially improve parameters of the already optimized circuits.

In addition to that, if we compare parameters of the optimized multiplier OPT2 given in Table IV with the best alternative of the manually designed multiplier (see OPT1 in Table III), it can be seen that a design with nearly the same mean error but significantly lower power consumption was obtained. The power consumption as well as PDP was reduced by approx. 40%. Our 4-bit multiplier has not only lower power consumption, but also significantly better worst-case error. Compared to the exact 4-bit multiplier the power consumption was reduced by 46% and PDP was improved by 60% for 180 nm technology process.

## IV. Conclusion

We proposed and evaluated a method for automatic design of low-power circuits. The method is based on a stochastic algorithm that gradually improves the original circuit. Even if our approach allows defining of an arbitrary design objective, we focused on optimizing the power consumption only. The goal of this paper was to (a) introduce the novel and systematic approach to the design of small low-power circuits, and (b) demonstrate that a noticeable reduction in the power consumption can be achieved if a design is optimized directly at the transistor level.

As a case study a design of low-power approximate multipliers consisting of several hundreds of transistors was chosen. The objective was to improve the power consumption of some selected existing solutions. To address this problem, we introduced and evaluated an algorithm for estimating the power consumption according to the switching activity of the transistors.

The results confirmed that the proposed method is able to improve the power consumption as well as the working frequency of the already optimized accurate and approximate versions of basic arithmetic circuits. A dramatic improvement in the power consumption was achieved when a combination of the gate-level and transistor-level optimizer was used. We discovered a novel implementation of an approximate 4-bit multiplier which has approximately by 40% better power-delay product and exhibits 14% lower worst-case error compared to the best known 4-bit multiplier published in [16] consisting of 2-bit manually optimized approximate multipliers.

We believe that the more complex circuits may provide greater potential for the power saving. The success of the proposed method is caused by the fact that the suggested encoding does not limit the logic style used to implement a given function. For example, it allows to use pass-transistor logic in some parts of the original CMOS implementation to improve speed or power consumption. To investigate this hypothesis, combination of the gate-level and transistor-level optimizers should be investigated more detailed in the future.

## Acknowledgement

This work was supported by the Czech science foundation project 14-04197S – Advanced Methods for Evolutionary Design of Complex Digital Circuits.

## References

[1] D. Soudris, C. Piguet, and C. Goutis, *Designing CMOS Circuits for Low Power*, ser. European low-power initiative for electronic system design. Springer, 2002.

[2] R. Zimmermann and W. Fichtner, "Low-power logic styles: Cmos versus pass-transistor logic," *Solid-State Circuits, IEEE Journal of*, vol. 32, no. 7, pp. 1079–1090, Jul 1997.

[3] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "Impact: Imprecise adders for low-power approximate computing," in *Proceedings of the 17th IEEE/ACM International Symposium on Low-power Electronics and Design*, 2011, pp. 409–414.

[4] D. Mohapatra, "Approximate computing: Enabling voltage over-scaling in multimedia applications," Ph.D. dissertation, Purdue University, 2011.

[5] S. Venkataramani, A. Sabne, V. J. Kozhikkottu, K. Roy, and A. Raghunathan, "Salsa: systematic logic synthesis of approximate circuits," in *The 49th Annual Design Automation Conference 2012, DAC '12*. ACM, 2012, pp. 796–801.

[6] S. Venkataramani, K. Roy, and A. Raghunathan, "Substitute-and-simplify: a unified design paradigm for approximate and quality configurable circuits," in *Design, Automation and Test in Europe, DATE'13*. EDA Consortium San Jose, CA, USA, 2013, pp. 1367–1372.

[7] K. Nepal, Y. Li, R. I. Bahar, and S. Reda, "Abacus: A technique for automated behavioral synthesis of approximate computing circuits," in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE '14. EDA Consortium, 2014, pp. 1–6.

[8] F. Farshchi, M. Abrishami, and S. Fakhraie, "New approximate multiplier for low power digital signal processing," in *Computer Architecture and Digital Systems (CADS), 17th CSI International Symposium on*, Oct 2013, pp. 25–30.

[9] C.-H. Lin and I.-C. Lin, "High accuracy approximate multiplier with error correction," in *Computer Design (ICCD), 2013 IEEE 31st International Conference on*, Oct 2013, pp. 33–38.

[10] A. Salz and M. Horowitz, "Irsim: An incremental mos switch-level simulator," in *Design Automation, 1989. 26th Conference on*, June 1989, pp. 173–178.

[11] R. Bryant, "Extraction of gate-level models from transistor circuits by four-valued symbolic analysis," in *The Best of ICCAD*, A. Kuehlmann, Ed. Springer US, 2003, pp. 337–346.

[12] R. Kumar, Z. Liu, and V. Kursun, "Technique for accurate power and energy measurement with the computer-aided design tools," *Journal of Circuits, Systems and Computers*, vol. 17, no. 03, pp. 399–421, 2008.

[13] J. Monteiro, S. Devadas, A. Ghosh, K. Keutzer, and J. White, "Estimation of average switching activity in combinational logic circuits using symbolic simulation," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 16, no. 1, pp. 121–127, Jan 1997.

[14] A. Shams, T. Darwish, and M. Bayoumi, "Performance analysis of low-power 1-bit cmos full adder cells," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 10, no. 1, pp. 20–29, Feb 2002.

[15] D. D. Gajski, F. Vahid, S. Narayan, and J. Gong, *Specification and Design of Embedded Systems*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1994.

[16] P. Kulkarni, P. Gupta, and M. D. Ercegovac, "Trading accuracy for power in a multiplier architecture," *Journal of Low Power Electronics*, vol. 7, no. 4, pp. 490–501, 2011.

[17] C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in *Proceedings of the Conference on Design, Automation & Test in Europe*, 2014, pp. 95:1–95:4.

[18] L. Sekanina and Z. Vasicek, "Approximate circuit design by means of evolvable hardware," in *Evolvable Systems (ICES), IEEE International Conference on*. IEEE Computer Society, April 2013, pp. 21–28.