Evolutionary Design of Transistor Level Digital Circuits using Discrete Simulation

Vojtech Mrazek and Zdenek Vasicek

Brno University of Technology, Faculty of Information Technology,
Božetěchova 2, 612 66 Brno, Czech Republic
{imrazek,vasicek}@fit.vutbr.cz

Abstract. The objective of the paper is to introduce a new approach to the evolutionary design of digital circuits conducted directly at transistor level. In order to improve the time consuming evaluation of candidate solutions, a discrete event-driven simulator was introduced. The proposed simulator operates on multiple logic levels to achieve reasonable trade-off between performance and precision. A suitable level of abstraction reflecting the behavior of real MOSFET transistors is utilized to minimize the production of incorrectly working circuits. The proposed approach is evaluated in evolution of basic logic circuits having more than 20 transistors. The goal of an evolutionary algorithm is to design a circuit having the minimal number of transistors and exhibiting the minimal delay. In addition to that, various parameter settings are investigated to increase the success rate of the evolutionary design.

1 Introduction

In recent years, a lot of papers showing the merits of evolutionary design techniques in the field of digital circuits design have been published. Implementation of various combinational circuits competitive to the circuits designed using conventional approaches have been obtained by using cartesian genetic programming (CGP) which is considered to be the most efficient technique to perform the gate-level evolutionary design [2–4, 7].

However, while the gate-level evolutionary design represents an intensively studied research area, the synthesis of transistor-level digital circuits remains, in contrast with design of transistor-level analog circuits, on a peripheral concern of the researchers despite the fact that even some basic logic expressions can be implemented much effectively at transistor level. Only few papers were devoted to evolution of digital circuits directly at transistor level. Zaloudek et al. published an approach based on a simple simulator which was designed to quickly evaluate the candidate solutions [11]. Unfortunately, a rough approximation of transistor behavior caused that this approach produced many incorrectly working circuits. Trefzer used another technique to evolve some basic logic gates [6]. Instead of using a time consuming analog circuit simulator, a reconfigurable analog transistor array was employed. However, it was shown that many of the discovered solutions relied on some properties of the utilized reconfigurable array. About

50% of the evolved circuits failed in the analog simulation. Walker et al. used a different technique to evolve transistor-level circuits [8]. In order to speed up the time consuming evaluation of candidate solutions, a cluster of SPICE-based simulators was utilized. Even if it was possible to evolve correct solutions, only small problem instances could be investigated due to the overhead of SPICE simulators.

A new approach to the evolutionary design of digital circuits is introduced in this paper. In this work, the evolutionary approach operates directly at transistor level. Since the evolutionary-based approach requires generating a large number of candidate solutions, it is necessary to minimize the time needed to evaluate the candidate circuits in order to obtain a satisfactory success rate. However, a reasonable level of abstraction must be applied to avoid production of incorrectly working circuits. In order to address this issue, a discrete simulator which operates on multiple logic levels is proposed. It is expected that this approach enables to achieve reasonable trade-off between performance and precision.

The paper is organized as follows. Section 2 discusses behavior of real unipolar transistors. Section 3 introduces the proposed method. Section 4 summarizes and analyses the obtained results. The analysis of the discovered circuits is performed using a SPICE simulator. Finally, concluding remarks are given in Section 5.

2 Behavior of MOSFET transistors

Behavior of the MOSFET transistors can be described at various levels of abstraction.

At the most accurate level, transistor circuits are modeled using a complex system of equations having tens of parameters that are derived from the underlying device physics. In order to accurately simulate the transistor level circuits, SPICE-based simulators are usually used. Apart from the commercial simulators such as HSPICE or PSPICE, there exist also academic tools such as ngSPICE. Even if the SPICE-based simulators provide a wide variety of MOS transistor models with various trade-offs between complexity and accuracy, the runtime grows rapidly with the increasing size of the simulated circuits. To reduce the time of simulation, a multithreaded version of SPICE simulator or an FPGA-based hardware acceleration can be utilized [1].

On the other hand, so-called switch-level model can be used [10]. A switch-level simulator models MOS circuits using a network of transistors acting as switches. Each transistor can be in one of three discrete states – open, closed or unknown. Compared to the SPICE-based simulators, the speed of the simulation is improved in orders of a magnitude. This model can acquire aspects that cannot be expressed at gate model, however, the accuracy is naturally lower compared to the approach mentioned in the previous paragraph. For example, the value of threshold voltage influencing state of the transistors is completely ignored. Moreover, the accuracy of simulation decreases as the transistors shrinks.

2.1 Discrete model suitable for evolutionary design

As it was discussed in the Section 1, a fast simulator is needed to enable the evolutionary design to sufficiently explore the search space. Simultaneously, reasonable accuracy is required to evolve the correct circuits that will work in real environment. In order to meet these requirements, we propose to utilize a discrete simulator which exhibits speed of the switch-level simulators and accuracy of the SPICE-based simulators. We propose to use a model (abstracted from dynamic parameters such as power consumption or delay) based on the switch-level transistor model extended to a threshold drop degradation effect.

Threshold voltage, commonly abbreviated as V_{tp} (V_{tn}), is the minimum gate-to-source voltage differential that is needed to create a conducting path between the source and drain terminals. As a consequence of the threshold voltage, degraded voltage values can be presented in MOS circuits. An open n-MOS transistor is known to pass logic 0 (i.e. V_{ss}) well but logic 1 (i.e. V_{dd}) poorly. This loss is known as threshold drop. An attempt to pass logic 1 never gives value above $V_{dd} - V_{tn}$. Similarly, p-MOS transistor is known to pass logic 0 poorly. The reduction in voltage swing can be beneficial to the power consumption. The designer has to be careful, however, because the degraded output may cause circuit malfunction.

As a target technology, TSMC with feature size equal to 0.25 μm is chosen. The following parameters of p-MOS and n-MOS transistors will be utilized in MOS circuits. The length of the n-MOS transistor is $L_N = 0.25\ \mu m$, width is $W_N = 0.5\ \mu m$. p-MOS transistors have $L_P = 0.25\ \mu m$ and $W_P = 2W_N = 1\ \mu m$. According to the simulation, $V_{tn} = 0.987\ V$ and $V_{tp} = 0.717\ V$. In order to

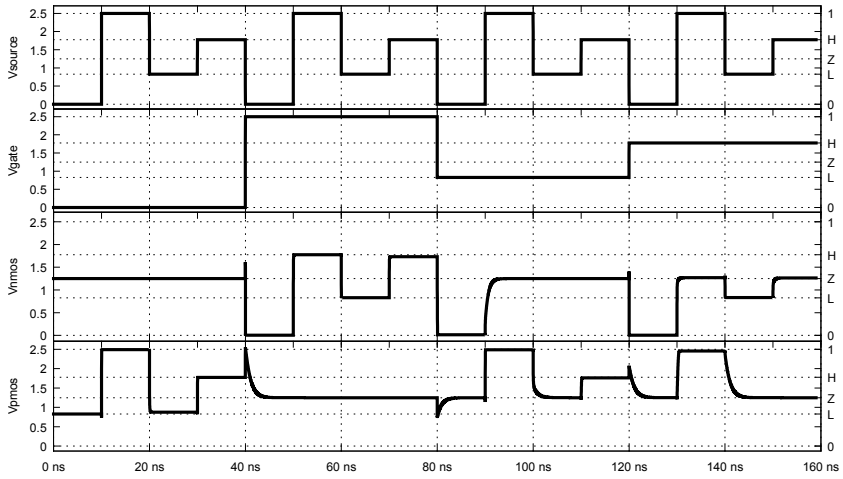

Fig. 1. Output waveforms for p-MOS and n-MOS transistors for various voltage applied to the source and gate terminals. The waveform was obtained using an analog SPICE simulator, a TSMC 0.25 μm technology and 2.5V power supply. The corresponding discrete values are shown on the right side.

Table 1. Behavior of n-MOS and p-MOS transistors modeled using six discrete values.

n-MOS							*p-MOS*						
gate	source						gate	source					
	1	H	L	0	Z	X		1	H	L	0	Z	X
1	H	X	L	0	Z	X	1	Z	Z	Z	Z	Z	X
H	X	X	L	0	Z	X	H	Z	Z	Z	Z	Z	X
L	Z	Z	Z	Z	Z	X	L	1	H	X	X	Z	X
0	Z	Z	Z	Z	Z	X	0	1	H	X	L	Z	X
Z	Z	Z	Z	Z	Z	X	Z	Z	Z	Z	Z	Z	X
X	X	X	X	X	X	X	X	X	X	X	X	X	X

support various implementations of digital circuits, we will distinguish among six voltage levels: logic 0 (denoted as '0'), logic 1 ('1'), degraded 0 (V_{tp}, 'L'), degraded 1 ($V_{dd} - V_{tn}$, 'H'), high impedance ('Z') and undefined value ('X'). A SPICE-based simulator was used to derive the discrete model. The results of simulation are given in Figure 1. The fourth terminal of p-MOS (n-MOS) is connected to V_{dd} (V_{ss}). In order to detect high impedance state, outputs of p-MOS and n-MOS transistors are connected to a voltage divider.

Let us discuss behavior of n-MOS transistor (p-MOS works analogically). If logic 0 is applied to the gate, the transistor is closed and its output is in a high impedance state. The similar situation occurs if 'L' is used. However, if $V_{gate} =$'L' and $V_{source} =$'0', the transistor is not completely closed. As we do not want to model strength of the signal values, we need to suppose that the output is in a high impedance state. This little inaccuracy does not constitute any serious problem due to the presence of stronger values within a circuit. If logic 1 or 'H' is applied to the gate, the transistor is open. Logic 0 as well as 'L' connected to the source are fully transferred to output, but logic 1 and 'H' are degraded. As we can see, the double degraded value can not be recognized from high impedance state. Hence, we have to avoid the double degradations that may cause malfunctions.

The behaviour of n-MOS and p-MOS transistors which follows the results obtained from the SPICE-based simulation valid for the chosen technology and power supply is summarized in Table 1.

3 The proposed method

3.1 Circuit representation

In order to evolve complex digital circuits at the transistor level a suitable representation enabling to encode bidirectional graph structures containing junctions is needed. To address this problem, we proposed an encoding inspired by CGP[2].

Each digital circuit having n_i primary inputs and n_o primary outputs (i.e. a candidate solution) is represented using an array of nodes arranged in n_c columns and n_r rows. Each node consists of two source terminals and one output

terminal. Each node can act as p-MOS transistor, n-MOS transistor, or junction. The utilized nodes are shown in Figure 2. Source terminals of each node can independently be connected to the output terminal of a node placed in previous l columns. In addition to that, source terminals of any transistor node can be connected to one of the primary circuit inputs.

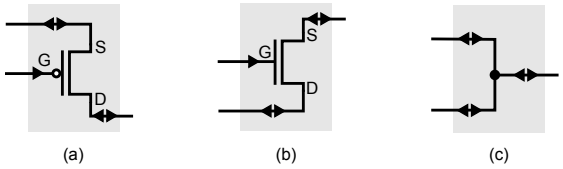

Fig. 2. Basic building blocks of transistor-level circuits: (a) p-MOS transistor, (b) n-MOS transistor, and (c) junction that combines two signals together. If a proper voltage is applied on the gate electrode denoted as G (V_{ss} for p-mos, V_{dd} for n-mos), transistor connects its source electrode (denoted as S) with drain (D). Possible directions of signal flow which have to be considered during the evaluation are shown.

Presence of the junction node represents the main feature of the proposed technique. This 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.

The following encoding scheme is utilized. The primary inputs and node outputs are labeled from 0 to $n_i + n_c \cdot n_r - 1$. A candidate solution is represented in the chromosome by $n_c \cdot n_r$ triplets (x_1, x_2, f) determining for each node its function f, and label of nodes x_1 and x_2 connected to the source terminals.

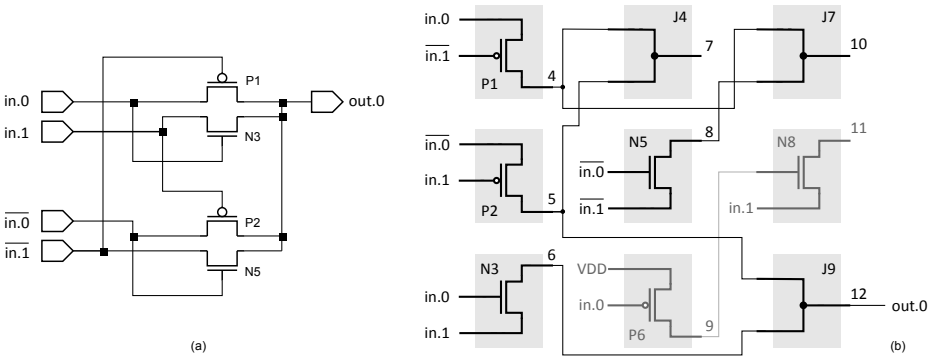

Fig. 3. Example of a candidate circuit implementing function XNOR using eight transistors (four transistors are used to implement inverted variables $\overline{in.0}$ and $\overline{in.1}$). Parameters are as follows: n_i=4 (0,V_{dd},in.0,in.1), n_o=1 (out.0), n_c=3, n_r=3, l=2. Chromosome: *(2,-3,pmos)(-2,3,pmos)(3,2,nmos)(4,5,junction)(-3,-2,nmos)(1,2,pmos)(4,8,junction)(9,3,nmos)(5,6,junction)(12).*

Apart from that, negative indices $-2 - n_i < x_i \leq -2$ are allowed in case of x_i. The negative value indicates that the inverted primary variable labeled as $|x_i|$ is required. The last part of the chromosome contains n_o integers specifying the labels of nodes where the n_o primary outputs are connected to. The first two primary inputs are reserved for power supply rails.

Figure 3 demonstrates the principle of utilized encoding on a XNOR circuit implemented using pass-transistor logic. The shown chromosome encodes a candidate circuit using eight nodes, however, only some of them contribute to the phenotype and are active.

3.2 Evaluation of the candidate solutions

Evaluation of the candidate solutions encoded using the proposed representation consists of two steps.

Firstly, set of active nodes is determined. Only the active nodes are considered during the evaluation. The inactive nodes are ignored. Potentially unwanted nodes causing short-circuits can be removed in this step. A node is active if either (a) its output is connected to any of the primary outputs, or (b) it is a transistor node and its output is connected to the source of an active node, or (c) it is a junction node whose source terminal is connected to an active node. The detection of active nodes can be performed in linear time complexity.

Then, multi-level discrete event-driven simulator is utilized to determine response for each input combination. The advantage of this approach is that only necessary nodes are updated if there is a change of a value. The following steps are used to determine output value of for a given input combination. Firstly, outputs of all nodes are initialized to the value 'Z'. Then, value 0 and 1 are assigned to the first two primary inputs. This change triggers re-evaluation of all the nodes connected directly to the power supply rails. Each node determines its new output value and propagates it to all related nodes. As an open transistor connect source with drain, bidirectional data-flow have to be utilized. It means that the new value must be propagated to the nodes connected not only to the drain but also to source terminal. Similarly, junctions have to propagate the new value to all terminals. The new value of a junction node is calculated as the strongest value presented on all the terminals. The new value of a transistor node is determined according to the value connected to the source as well as drain. During the evaluation of a new output value of a transistor node, the new calculated value is compared with current value at drain terminal. If the values are not compatible, short circuit exception is raised. Otherwise, the stronger value is propagated to all related nodes. The relation between the discrete values is as follows: 'Z' $\prec$ 'L' $\prec$ '0' $\prec$ 'X'; 'Z' $\prec$ 'H' $\prec$ '1' $\prec$ 'X'. It means that if at least one of the values is equal to 'X', 'X' is propagated to all related nodes.

Each transistor has associated a state which determines whether the transistor is in direct or reverse mode. The current flows from drain to source in reverse mode. It happens when 'Z' is assigned to source terminal and a value different from 'Z' is connected to the drain. This state helps to avoid situation in which a double degradation could happen.

In order to avoid malfunction circuits, final test is performed at the end of the simulation. If there is at least a single transistor with 'Z' state assigned to its gate terminal, short circuit exception is raised.

The principle of discrete simulation will be demonstrated for in_0=1 and in_1=0 using the candidate circuit shown in Figure 3. The primary inputs are successively initialized to the following values: V_{dd} (i.e., the primary input with index 0) ← '1', $V_{ss}(1)$ ← '0', $in_0(2)$ ← '1', $in_1(3)$ ← '0'. Then the inverted values are assigned $\overline{in_0}$(-2) ←'0', $\overline{in_1}$(-3)←'1' As no power rail is used in the example, the first two assignments do not trigger any reevaluation. However, assignment of value '1' to in_0 causes that P1 and N3 are evaluated. Nor P1 nor N3 have fully specified inputs, thus these changes do not generate any new event. In the next step, in_1 connected to P2 and N3 is assigned. Now, the node N3 has fully specified inputs and the new calculated value '0' is propagated through drain to the node J9. Then, the value of $\overline{in_0}$ is changed to '0'. As a consequence of that, P2 is evaluated to 'L' and propagated through J4 to J9. In addition to that, N5 is refreshed. Because there is a stronger value, '0', assigned to the other pin of J9, the '0' is propagated back to the output terminal of transistor P2 and junction J4. The, $\overline{in_1}$ becomes to be logical '1'. Transistor P1 is closed, so the drain is in high impedance state. This value is propagated to J4, however '0' presented at the second terminal is stronger and it is propagated back to P1 and then to J7. The last transistor which has to be evaluated is the closed transistor N5 with 'Z' at its output. High impedance state is delivered to J7, but J7 already contains a stronger value '0'. Primary output is connected to the node J9 which has value '0' on its output. This value corresponds with the XNOR specification, so the circuit produces a valid output for the used input vector.

3.3 Search strategy

As a search algorithm, $(1 + \lambda)$ evolutionary strategy is utilized [2]. The initial population is randomly generated. Every new population consists of the best individual and λ offspring created using a point mutation operator which modifies h randomly selected genes. In the case when two or more individuals have received the same fitness score in the previous population, the individual which did not serve as a parent in the previous population will be selected as a new parent. This strategy is used to ensure the diversity of population. The evolution is terminated when a predefined number of generations is exhausted or a required solution is found.

The search is guided by the fitness function which determines how good the current candidate circuit is. For evolution of logic circuits, all possible input combinations have to be applied at the candidate circuit inputs. The output values are collected and the goal is to minimize the difference between obtained responses and required Truth table. In order to smooth the search space, the fitness value is constructed as follows. If an obtained output value equals to the expected one, 5 points are added to the fitness value. If the calculated value exhibits the same polarity but represents degraded voltage, 2 points are used.

Otherwise, no point is added because the response is invalid. Additional penalties may be applied. If there is a short-circuit exception asserted during the simulation, the simulation is terminated and penalty is applied to the total fitness value. Similarly, if the simulator exceeds the predefined number of steps (i.e. node outputs are not in stable state), the simulation is terminated and the fitness value is penalized. As soon as a fully working solution is found, the number of utilized transistors is reduced. Two points are added for each unused node and one point for node which acts as junction. Note that the transistors required to implement inverted input of the utilized variables are considered.

4 Experimental results

The proposed method was evaluated in the evolution of basic logic circuits as well as some benchmark circuits whose conventional solutions consist of up to 30 transistors. In particular, we tried to evolve XOR and XNOR gate, 3 bit majority, 1 bit full adder and benchmark circuits b1, c17, newtag, mc, daio and lion from *LGSynth benchmarks*. The goal of the experiments was to evolve fully functional implementations exhibiting full voltage swing on the outputs.

In order to investigate the effect of array size, three arrangements are used for each benchmark circuit. The first two configurations utilize a single row of nodes, while the third uses an array consisting of two rows. The total number of nodes was chosen according to the number of transistors required to implement a given function using a conventional design approach.

In addition to that, the impact of various connection possibilities was investigated. Firstly, the presence of inverted input variables introduced in Section 3.1 and its impact on the success-rate was studied. Then, additional restriction to the connection of source terminal of p-MOS and drain terminal of n-MOS was applied. We prevent to connect this electrode directly to the primary inputs. As a consequence of that, implementations with higher operating frequency can be evolved. This setup is denoted as 'S/D←N', while the unrestricted setup is denoted as 'S/D←I+N'.

The results were obtained from 20 independent runs using the following experimental setup: $\lambda = 4, l = n_c, h = 5$. The evolution is terminated after 8 hours or when no improvement was achieved within the last hour. All the successfully evolved solutions were validated using a SPICE simulator.

The results were compared with a reference implementation described at gate-level and implemented using standard cells.

The impact of the introduced restriction and the presence of implicit inverters is evaluated by means of a *success proportion* [9]. Success proportion is the cumulative probability of success calculated by the number of runs that have found a solution at or before generation i divided by the total number of runs in the experiment. A successful run is such a run in which a fully working solution was discovered. The results for two chosen benchmark circuits are given in Figure 4. As it can be seen, the usage of implicit inverters significantly increased the performance of the evolutionary design. On the other hand, the restriction

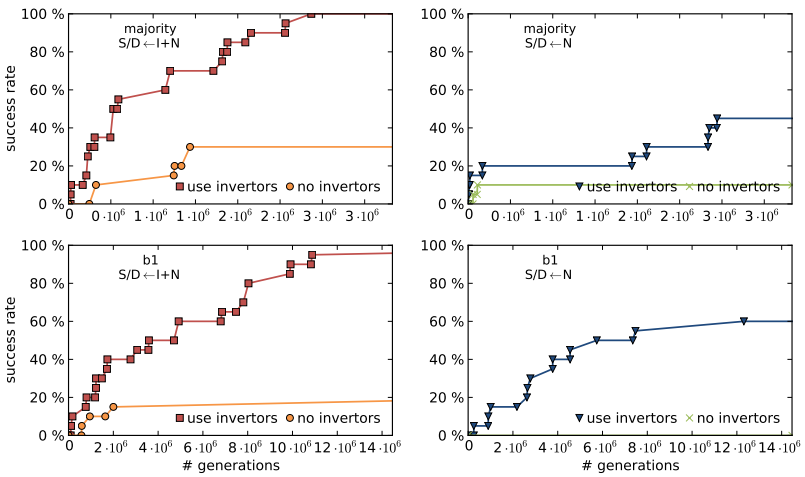

Fig. 4. Success proportion of the evolutionary design of 'majority' and 'b1' benchmark circuits. The array consisting of a single row and 30 columns for 'majority' and 60 columns for 'b1' are used.

applied to the source (drain) terminals of p-MOS (n-MOS) nodes reduce the performance of the evolution. Substantially higher number of generations are needed to achieve the same success rate.

The success rate of the evolutionary design for the chosen digital circuits is summarized in Table 2. In addition to that, we analyzed the evolved solutions and determined the number of utilized transistors (see the last two columns). Similarly to the previous findings, the usage of implicit inverters as well as the unrestricted possibilities of S/D terminal connections improved the performance of the evolutionary approach in all cases. Another parameter which can have a great impact on the success rate is the size of array. Too small array on the one hand and too large array on the other hand have a negative impact on the success rate. While the small array may prevent to find a valid solution because there is not a space to represent a target circuit, large array increases substantially the search space. Fortunately, it seems that increasing of the number of available nodes does not increase the size of the evolved circuit.

The discovered circuits were verified and characterized using a SPICE simulator with an accurate transistor model. Except of a single evolved implementation of 'b1' circuit, all the circuits were valid and operated correctly. Thus we can conclude that the proposed discrete abstraction is successful.

Table 3 summarizes the basic parameters of the evolved solutions and the conventional implementations. Apart from the number of utilized transistors, delay and maximum operating frequency is given. If we compare the maximum operating frequency of the evolved circuits with the conventional circuits, we can see a significant improvement in all cases except the circuit 'c17'. This result is

Table 2. Success rate for the benchmark circuits for various array sizes, connection possibilities and availability of inverted primary inputs.

	$n_r \times n_c$	S/D←N+I		S/D←N		# transistors	
		with inv.	w/o inv.	with inv.	w/o inv.	min	max
xnor	1×10	100%	65%	0%	0%	6	8
	1×15	100%	100%	100%	5%	6	12
	2×15	100%	100%	100%	45%	6	12
xor	1×10	100%	75%	0%	0%	6	8
	1×15	100%	100%	100%	5%	6	12
	2×20	100%	100%	100%	5%	6	12
majority	1×20	100%	25%	0%	5%	10	14
	1×30	100%	30%	45%	10%	10	16
	2×30	80%	35%	60%	15%	10	17
adder-1	1×30	30 %	5%	0%	0%	14	20
	1×40	65%	0%	0%	0%	18	20
	2×40	50%	0%	5%	0%	18	25
b1	1×40	100%	15%	40%	0%	12	19
	1×60	100%	20%	60%	0%	12	20
	2×60	75%	5%	25%	0%	12	23
c17	1×40	5%	0%	0%	0%	22	24
	1×60	5%	0%	0%	0%	25	26
	2×60	0%	0%	5%	0%	25	28

very encouraging, because the delay was not optimized explicitly. We analyzed the circuits and determined that this improvement was achieved by replacing traditional gates implemented as CMOS logic with much effective implementation which utilized so-called transmission-gates. The usage of transmission-gates increases the speed but simultaneously reduces the number of utilized transistor.

A lot of different implementations were discovered. Example of an evolved circuit of one bit adder is shown in Figure 5. The discovered circuit is similar to low-power full adder consisting of 14 transistors which was introduced in [5]. The evolution was able to discover an implementation which belongs to the family of pass-transistor logic. The evolved solution utilizes three transmission gates to provide fast and compact solution and exhibits approx. 27% reduction in power consumption compared to the common CMOS implementation. Carry is represented by output labeled as out_0 and sum is available at out_1. Input in_2 corresponds to the input carry.

5 Conclusion

A new approach suitable to the evolutionary design of digital circuits conducted directly at transistor level was introduced in this paper. A discrete event-driven

Table 3. Parameters of the conventional as well as evolved digital circuits. The first part of the table contains the number of inputs, number of outputs and time and number of generations required to evolve the solution. Then, the parameters of conventional implementation are given. (a) Contains parameters of the fastest discovered solution, while (b) contains parameters of the most compact evolved solution.

		xor	xnor	majority	adder-1	b1	c17
	Inputs	2	2	3	3	3	5
	Outputs	1	1	1	2	2	2
	Time of evolution (min)	10	10	10	120	60	480
	Max. # generations	$14 \cdot 10^6$	$14 \cdot 10^6$	$5 \cdot 10^6$	$45 \cdot 10^6$	$30 \cdot 10^6$	$80 \cdot 10^6$
	Delay (ps)	208.3	180.9	335.2	422.7	360.1	324.0
	Frequency (GHz)	4.80	5.53	2.98	2.37	2.78	3.09
	Transistors	8	8	22	48	30	28
	Delay (ps)	87.5	87.8	271.4	291.4	173.2	355.4
(a)	Frequency (GHz)	11.43	11.39	3.68	3.43	5.77	2.81
	Transistors	6	8	16	14	16	24
	Delay (ps)	87.5	142.4	599.3	291.4	401.5	573.8
(b)	Frequency (GHz)	11.43	7.02	1.67	3.43	2.49	1.74
	Transistors	6	6	10	14	12	22

simulator operating on multiple logic levels was utilized to achieve reasonable trade-off between performance and precision. The proposed method was evaluated on a set of benchmark circuits.In order to improve the success rate, implicit inverters were introduced to the encoding.

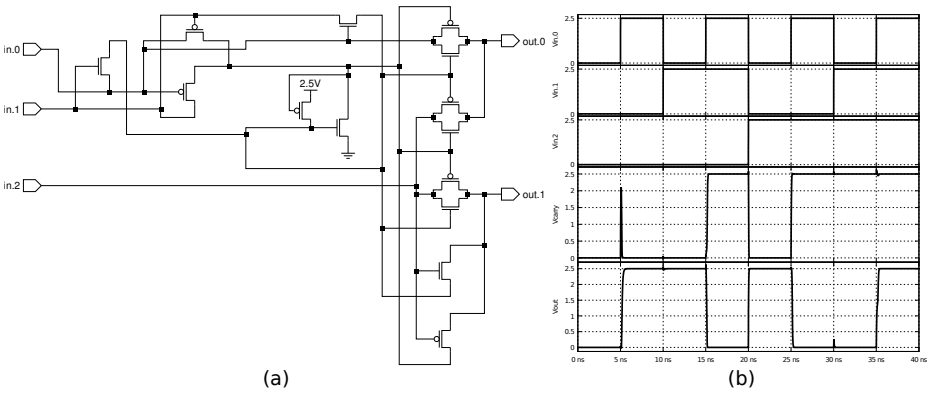

Fig. 5. (a) The most compact and simultaneously the fastest circuit consisting of 14 transistors implementing one-bit full adder. (b) Output waveform obtained using a SPICE simulator.

It was demonstrated that the proposed method is able to produce valid solutions despite the fact that a relative simple discrete model of MOS transistors (compared to the complex models used in SPICE-based simulators) was utilized. According to the analysis of the obtained results, we can confirm, that the evolution was able to discover solutions that are based not only on complementary logic but also on pass-transistor logic.

However, future work has to be conducted to improve the scalability of the proposed method. One of the possible directions is to introduce more complex building blocks such as transmission gate.

Acknowlegement This work was supported by the Czech science foundation project 14-04197S.

References

1. Kapre, N., DeHon, A.: Accelerating spice model-evaluation using fpgas. In: Field Programmable Custom Computing Machines, 2009. FCCM '09. 17th IEEE Symposium on. pp. 37–44 (April 2009)
2. Miller, J.F. (ed.): Cartesian genetic programming. Natural Computing Series, Springer, Berlin, 22. edn. (2011)
3. Miller, J.F., Job, D., Vassilev, V.K.: Principles in the Evolutionary Design of Digital Circuits – Part I. Genetic Programming and Evolvable Machines 1(1), 8–35 (2000)
4. Miller, J.F., Job, D., Vassilev, V.K.: Principles in the Evolutionary Design of Digital Circuits – Part II. Genetic Programming and Evolvable Machines 1(3), 259–288 (2000)
5. Shams, A., Bayoumi, M.: A novel high-performance cmos 1-bit full-adder cell. Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on 47(5), 478–481 (May 2000)
6. Trefzer, M.: Evolution of Transistor Circuits. Ph.D. thesis, Ruprecht-Karls-Universitt Heidelberg (2006)
7. Vassilev, V., Job, D., Miller, J.: Towards the Automatic Design of More Efficient Digital Circuits. In: Proc. of the 2nd NASA/DoD Workshop on Evolvable Hardware. pp. 151–160. IEEE Computer Society, Los Alamitos, CA, USA (2000)
8. Walker, J., Hilder, J., Tyrrell, A.: Evolving variability-tolerant cmos designs. In: Hornby, G., Sekanina, L., Haddow, P. (eds.) Evolvable Systems: From Biology to Hardware. LNCS, vol. 5216, pp. 308–319. Springer Berlin Heidelberg (2008)
9. Walker, M., Edwards, H., Messom, C.H.: Success effort and other statistics for performance comparisons in genetic programming. In: IEEE Congress on Evolutionary Computation. pp. 4631–4638 (2007)
10. Weste, N.H., Harris, D.: CMOS VLSI design: a circuits and systemes perspective. Addison-Wesley, Boston, USA, 3. edn. (2005)
11. Zaloudek, L., Sekanina, L.: Transistor-level evolution of digital circuits using a special circuit simulator. In: Evolvable Systems: From Biology to Hardware. LNCS, vol. 5216, pp. 320–331. Springer Verlag (2008)