Research and development of methods for technology-independent power optimization of integrated circuits

Alexander Kamkin
Plekhanov RUE
ISP RAS
MSU, MIPT, HSE
Moscow, Russia
kamkin@ispras.ru

Sergey Smolov
Plekhanov RUE
ISP RAS
Moscow, Russia
smolov@ispras.ru

Anastasiya Kurganskaya
ISP RAS
MIEM, HSE
Moscow, Russia
askurganskaya_1@edu.hse.ru

Alexey Yagzhov
ISP RAS
MIEM, HSE
Moscow, Russia
aayagzhov@edu.hse.ru

***Abstract* — Increasing power consumption has become a major issue in the integrated circuit industry. Solving this problem will lead to a reduction in packaging costs. The main challenge in this problem is to generate a netlist that minimizes power dissipation. This paper proposes a method for technology-independent power optimization of combinational circuits. The proposed algorithm is based on a probabilistic estimate of power consumption. The method includes 3 consecutive stages: rewriting, refactoring and lazy refactoring. The main idea of each stage is the local resynthesize of sub-circuits of the initial circuit to minimize the power consumption. Experimental results demonstrated reducing the switching activity by 20.5 % with developed algorithm.**

Keywords — integrated circuit combinational circuits, switching activity, power consumption, technology-independent optimization

I. Introduction

In the era of advanced technology and increasing demands for high-performance electronic devices, the design and manufacturing of integrated circuits (IC) play an increasingly important role in digital systems. IC is a group of electronic circuits placed on a metal plate designed with semiconductor materials. An IC is the fundamental building block for all modern electronic devices.

The complexity and functionality of ICs continue to increase, so does their power consumption. The increase in power will lead to an increase in heat dissipation cost and packaging cost [1, 2]. Reducing power consumption in control systems, especially digital circuits, is an urgent scientific and technical issue that has recently attracted increasing attention from scientists and engineers [3]. It is clear that estimating and minimizing power during the design phase is crucial to avoid costly redesigns during manufacturing.

The IC design process contains several steps: 1) system specification, 2) architectural design step, 3) RTL model design, 4) logic synthesis, 5) physical synthesis [4]. Logic synthesis transforms a cycle-level functional design into a gate-level representation [5]. It makes up the gap between the technology-independent and technology-dependent stages. The result of logic synthesis is an optimal network composed of standard cells in a given technology library. In most design systems, the logical synthesis process contains at least two important stages: technologically independent optimization and technological mapping [6]. During the technology-independent optimization stage, the circuit is represented as a Boolean function, and this function is transformed into an equivalent one with optimal quality metrics.

The technology-independent power optimization is very important, because power optimization has a cumulative effect [7].

This paper is aimed to explore the problem of technology-independent power optimization of combinational circuits. This paper offers an initial overview of some key definitions in section 2. Then the power estimation model on a technology-independent phase is considered in section 3. After that the optimization approach is proposed in section 4. The current results are described in section 5. Section 6 concludes the paper.

II. Background

Circuits and nodes: a Boolean circuit is a *directed acyclic graph* (DAG). Logic gates are nodes, with wires connecting these gates representing edges. The *primary inputs* (PIs) are the nodes of the Boolean circuit without incoming edges. The *primary outputs* (POs) are a subset of the nodes that connect the Boolean circuit with its environment. A Boolean circuit may be represented in different basis and represented using various graphs. An *And-Inverter Graph* (AIG) is a directed acyclic graph (DAG) in which there are PI, PO and two-input AND gate with two incoming edges, respectively. It is possible that there may be certain markings on the edges. A marked edge is indicative of a signal inversion.

Cuts: a *cut* C of node v is a set of nodes, called *leaves*, through which all paths from the primary inputs of the original circuit to node v lead. Node v is the *root* of the cut. A cut is *K-feasible* if the number of leaves does not exceed K. The *cut function* is the function of node v in terms of the cut leaves.

NPN-equivalent: two Boolean functions, f and g, are considered to be *NPN-equivalent* if f can be derived from g by negating and permuting the inputs and negating the output.

Switching activity: the average number of switches per cycle at the output of the gate is the *switching activity*.

III. Estimation Model

The power dissipation of CMOS digital circuits includes two components: dynamic one and static one. Reverse currents of p-n junctions, resistive load, and leakage currents are the cause of static power. Static power is dissipated when the logic element is in a fixed logic state ("0" or "1"). The reasons for the dynamic power are processes of charging and discharging of the circuit node capacitance. This energy is dissipated when the signals at the outputs of the circuit nodes are switched. And this component of power dissipation is dominant [8] in comparison with static power. This assertion is not applicable to VLSI (Very-large-scale integration) ICs. But this paper will not present such circuits.

The average level of dynamic power dissipation for a single gate is estimated by the following approximation [9]:

$$P_{avg} = 0.5 \cdot \frac{C * V^2}{T} * E_{sw} \qquad (1)$$

where C - average load capacitance of the gate; V - supply voltage; T - clock cycle time; E_{sw} - switching activity.

The only multiplier in the formula that can be changed in the technology independent stage is E_{sw} (rest of them are constants for the optimized circuit). As a result, to minimize power consumption, the switching activity should be minimized.

B. Power estimation model

Power estimation methods are classified [10, 11]:

- methods based on modeling
- statistical methods
- probabilistic methods

The modeling is the simplest method and the most accurate method. The specified sets are fed to the input of the circuit. Then the behavior of the circuit is simulated and as a result, a power value is obtained. This method is costly.

The idea of statistical estimation is to repeatedly simulate the operation of the circuit. In this case, random sets are fed to the input of the circuit. The simulation takes place until the average power value has not been obtained. This method is less accurate than the previous one. A stopping criterion based on statistics is required for this method.

The probabilistic method is the fastest. This method is based on calculating the probability of switching each node of the circuit [8].

For estimation of the switching activity of combinational circuits a probabilistic model was chosen. In this idea, a signal probability is used to evaluate the switching activity. Signal probability can be calculated by the signal's input probabilities depending on the logical function of the cell. Figure 1 illustrates the formulas for calculating signal probabilities for several Boolean functions: inversion, logical AND, logical OR, logical XOR. Signal probabilities for three-input, four-input functions etc. are calculated using a similar idea.

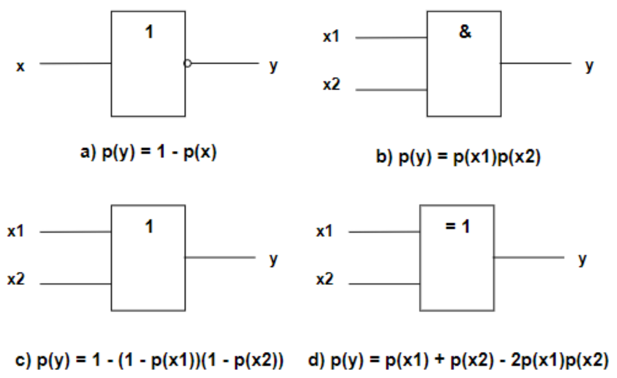
1
x
y
a) p(y) = 1 - p(x)
&
x1
x2
y
b) p(y) = p(x1)p(x2)
1
x1
x2
y
c) p(y) = 1 - (1 - p(x1))(1 - p(x2))
= 1
x1
x2
y
d) p(y) = p(x1) + p(x2) - 2p(x1)p(x2)

Figure 1. Formulas for calculating signal probabilities for a) inverter; b) two-input AND; c) two-input OR; d) two-input XOR.

The switching activity for gate i is the sum of the probabilities of switching from 0 to 1 and from 1 to 0. The probability of switching from 1 to 0 is equal to the product of the probabilities that the gate is in state 1 in this clock cycle and in state 0 in the next clock cycle. The formula for the probability of switching from 1 to 0:

$$p_{sw}^{1\to0} = p(1) * p(0) \qquad (2)$$

Similarly, the formula for the probability of switching from 0 to 1:

$$p_{sw}^{0\to1} = p(0) * p(1) \qquad (3)$$

Consequently, the formula for the switching activity:

$$E_{sw} = p_{sw}^{1\to0} + p_{sw}^{0\to1} = 2p(0) * p(1) \qquad (4)$$

Finally, $p(0) = 1 - p(1)$. The formula for the switching activity for gate i can be calculated as follows:

$$E_{sw} = 2p(1) * \left(1 - p(1)\right) \qquad (5)$$

where $p(1)$ - the output signal probability that the gate is in state 1 .

IV. Proposed Method

According to the selected power estimation method, switching activity should be minimized for power optimization. The proposed method is based on three approaches which are applied consecutively for input circuit: rewriting, refactoring and lazy refactoring.

A. Rewriting

B. Refactoring

It represents a modification of rewriting. The basic idea is that a single large cut is computed for each node. Optimization is based on the replacement the cut with sub-circuit that implement the same Boolean function, but the replacement has to lead to a decrease in the switching activity of the circuit as a whole. The change in switching activity can be calculated as the difference in switching activity of the removed nodes and the added nodes. To reduce the error, the minimum threshold for reducing the switching activity should be set. Methods such as Akers [12], Bi-Decomposition [13], Minato-Morreale [14], DSD-decomposition [15] and others can be used as synthesis algorithms to obtain equivalent Boolean functions.

The implemented method employs the Minato-Morreale algorithm without algebraic simplification of sum-of-product forms (SOPs) for synthesis of sub-circuits. Reconvergence-driven cuts of size 10 were extracted for this approache (reconvergence refers to the situation in which paths starting at the output of one of the nodes meet again before reaching the primary output of the design) [16].

C. Lazy refactoring

The algorithm is a modification of basic algorithm of refactoring with delayed replacement. The main idea is to resynthesize equivalent sub-circuits with reduced switching activity and substitute the initial sub-circuits with those exhibiting optimal switching activity. In this algorithm two-level sub-graphs are optimized. However, that not all sub-graphs are replaced; only those that are non-intersecting are.

This approach consists of two stages: 1) select and optimization of each two-level sub-circuit; 2) select from the received set of intersecting sub-graphs a subset of non-intersecting ones.

First stage: The main purpose of this stage is to identify all sub-circuits that can be optimized. The optimization process is based on the local resynthesize of two-level sub-graphs, which involves reordering inputs. In the initial stage, a two-level sub-graph is constructed for each node. Each sub-graph is a Boolean function. In case the Boolean function under consideration permits the interchange of inputs, then all possible permutations of the inputs are considered. For each variant the switching activity can be calculated. The sub-graph with the minimum value of the switching activity is selected. And in case the value of the switching activity is strictly less than that of the initial sub-graph, the latter is added into the desired set.

The possibility of permutation of inputs is determined by the properties of the Boolean function. If a function is both associative and commutative, it is possible to swap the inputs. For example, in Figure 2 two-level sub-graph is created for the 6th node. In this example, the inputs may be rearranged in any desired order.

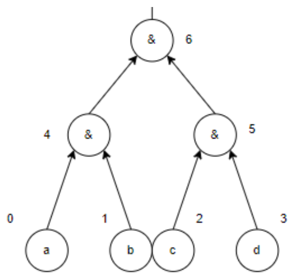

Figure 2. Sub-circuit

Figure 3 shows three unique sub-graphs in terms of switching activity for the Boolean function presented in Figure 2. All of these sub-graphs are equivalent functions, but the overall switching activity of these three sub-graphs may vary depending on the distribution of input probabilities.

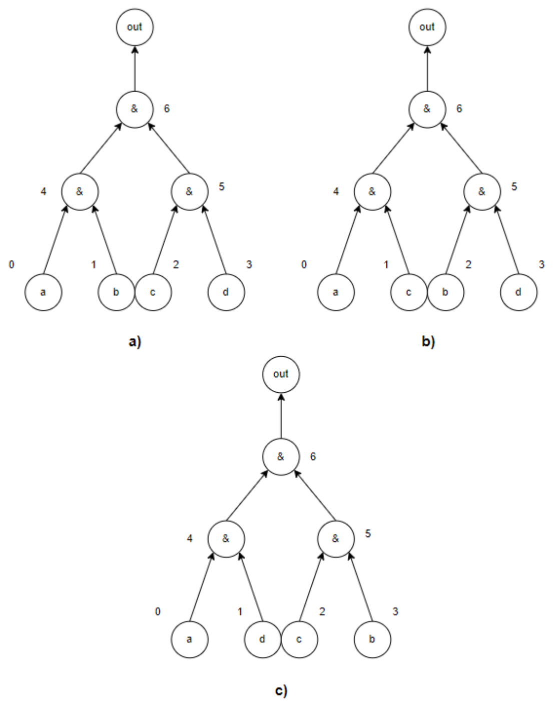

Figure 3. Example

Suppose the probability that the inputs gate in state 1 is given and they are as follows: $p_a(1) = 0{,}3$; $p_b(1) = 0{,}7$; $p_c(1) = 0{,}1$; $p_d(1) = 0{,}5$,

where $p_i(1)$ – the probability that the input gate i in state 1.

The overall switching activity for each sub-graph in Figure 3 is detailed below:

$$E_{sw} = E_{sw}^{inputs} + E_{sw}^{internal} + E_{sw}^{outputs} = \sum_{i=0}^{3} E_{sw}^{i} + \\ + E_{sw}^{internal} + E_{sw}^{6} \quad (6)$$

The overall switching activity of the inputs (nodes 1, 2, 3, and 4) and outputs (node 6) for the various sub-circuits remains unchanged, but the switching activity of the internal nodes (nodes 4 and 5) differs for sub-graphs a, b, and c.

This example shows how the overall switching activity of the internal gates of sub-graph can change when the inputs are rearranged. Switching activity of internal gates for sub-graph a:

$$E_{sw}^{3} = 2p_a(1)p_c(1)\big(1 - p_a(1)p_c(1)\big) = 0{,}3318$$

$$E_{sw}^{4} = 2p_c(1)p_d(1)\big(1 - p_c(1)p_d(1)\big) = 0{,}095$$

$$E_{sw}^{internal} = 0{,}4268$$

for sub-graph b:

$$E_{sw}^{3} = 2p_a(1)p_b(1)\big(1 - p_a(1)p_b(1)\big) = 0{,}0582$$

$$E_{sw}^{4} = 2p_b(1)p_d(1)\big(1 - p_b(1)p_d(1)\big) = 0{,}455$$

$$E_{sw}^{internal} = 0{,}5132$$

for sub-graph c:

$$E_{sw}^{3} = 2p_a(1)p_d(1)\big(1 - p_a(1)p_d(1)\big) = 0{,}255$$

$$E_{sw}^{4} = 2p_b(1)p_c(1)\big(1 - p_b(1)p_c(1)\big) = 0{,}1302$$

$$E_{sw}^{internal} = 0{,}3852$$

In this example, the inputs are swapped in the sub-graph and the overall switching activity has changed.

Second stage: The final task is to select a set of non-intersecting sub-graphs from the set of intersecting ones resulted from the first stage. Additionally, the solution to such a task should provide the greatest possible gain in optimizing the total switching activity of the circuit.

The problem is depicted as a graph. The graph represents sub-graphs that need to be replaced as nodes, and edges indicate intersections between sub-graphs. Each node has a weight. It is optimization gain that can be obtained by replacing. This problem is equivalent to the maximum weighted independent set problem. To solve it, a greedy algorithm is used.

When the non-intersecting set is selected, sub-circuits are replaced. The algorithm can be applied multiple times to the same circuit to achieve the best result.

In this method the preliminary estimation of optimization can be obtained before replacement. And this algorithm can be applied to circuits on an arbitrary basis.

V. Results

The considered method was implemented using the C++ language and was applied to circuits from OpenABC-D [17] dataset, which were represented in AIG.

Although the logical optimization process used a probabilistic approach to evaluate the switching activity, simulation (under a zero-delay model) of the source and resulting circuits was used to evaluate the results.

Table 1 presents the results of the optimization of area (Area) which is calculated as the number of logic gates in the circuit and switching activity (SA) for 47 circuits from the dataset. Each of first three columns illustrates the results of optimization for each stage individually, relative to the previous step: rw – rewriting; rf – refactor; lr – lazy refactor. The fourth column presents the results of optimization for the proposed method relative to the source circuit. Optimization was calculated using the following expression:

$$Delta = \frac{E_{sw}^{old} - E_{sw}^{new}}{E_{sw}^{old}} * 100\% \qquad (7)$$

where E_{sw}^{old} - the initial switching activity, E_{sw}^{new} - the switching activity after optimization.

The similar expression was used for area.

Additionally, we compared our method with the resyn2 pass from ABC [18] and resyn2, which consist from our approaches for optimization switching activity (modified resyn2):

$$b; rw; rf; b; rw; rwz; b; rf; rwz; 5 * lr \qquad (8)$$

where b – balance; rw – rewriter; rwz – zero-cost rewriter, rf – refactor for switching activity optimization; lr – lazy refactor.

The final two columns show the result of the optimization of area and switching activity by ABC resyn2 and our modified resyn2 (m-resyn2).

The table 2 shows the average results for area and switching activity optimization which was calculated as the arithmetic mean of the optimization values for all considered circuits.

Consequently, using our method for optimizations and modified resyn2 pass yields comparable results in terms of area optimization, as well as better results in regard to switching activity optimization, when compared to the resyn2 pass from ABC.

VI. Conclusion

Power optimization of logic circuits is a crucial concern in computer engineering. This paper presents a technology-independent approach to minimize the power consumption of integrated circuits. The optimization method comprises a number of stages. Firstly, the circuit is reduced in size through rewriting. Then we use refactoring with synthesis algorithms to reduce the switching activity of cones. Finally, a heuristic method is proposed to further reduce switching activity without significant circuit changes.

For future work, the power minimization method will be subjected to further development. The refactoring stage can be improved by using Boolean function synthesis algorithms that take into account the switching activities of the input gates, in order to reduce the power consumption of the synthesized circuits.

Additionally, further research will be conducted to assess the impact of the proposed method on the circuit after the technology-dependent stage.

ACKNOWLEDGEMENTS

This research was carried out at Plekhanov Russian University of Economics at the expense of a grant from the Russian Science Foundation, № 23-21-00313, https://rscf.ru/project/23-21-00313.

TABLE I. EXPERIMENT RESULTS OF OPTIMIZATION CIRCUITS IN AIG

Design	rw		rf		lr		all 3 stages		ABC resyn2		m-resyn2	
	Ar-ea	SA	Area	SA	Area	SA	Area	SA	Area	SA	Area	SA
ac97_ctrl	3,6	3,5	1,9	4,0	0,0	0,1	5,5	7,5	3,4	3,1	5,7	7,6
aes	9,7	23,6	0,0	0,0	0,0	0,1	9,7	23,6	9,9	-2,9	9,9	24,4
aes_secworks	26,3	34,0	-1,4	0,4	0,1	0,0	25,3	34,2	19,1	22,0	26,8	34,2
aes_xcrypt	8,2	15,6	-1,4	0,5	0,1	0,0	6,9	16,1	6,6	1,4	7,2	15,7
apex1	6,8	3,3	0,6	4,5	0,5	1,1	7,8	8,7	19,8	13,9	9,7	7,4
bc0	18,6	18,8	-26,7	28,1	0,5	2,3	-2,6	43,0	40,5	38,3	0,9	50,6
bp_be	2,4	2,0	-0,3	2,3	0,0	0,1	2,1	4,5	3,7	1,9	2,1	5,0
c1355	56,4	49,1	0,0	0,0	0,0	0,0	56,4	49,1	20,2	12,2	55,7	47,3
c5315	22,9	21,3	-0,5	0,2	0,0	0,7	22,5	21,9	16,2	11,9	24,0	23,0
c6288	38,2	31,9	-2,2	1,0	0,0	0,3	36,9	32,8	19,5	21,4	37,4	33,0
c7552	37,6	34,6	0,0	0,0	0,0	0,4	37,6	34,8	29,1	20,0	40,0	35,4
dalu	32,2	38,4	-5,1	6,8	0,0	0,1	28,8	42,6	36,9	32,8	1,3	42,9
des3_area	8,4	13,4	0,0	0,0	0,0	0,0	8,4	13,4	7,1	12,8	8,4	13,4
dft	4,5	4,3	-14,3	13,1	0,0	0,1	-9,2	16,9	2,0	1,3	-1,8	19,3
div	40,9	39,5	-0,2	0,0	0,0	0,0	40,7	39,5	28,7	28,1	41,2	40,5
dynamic_node	3,5	2,7	-2,1	5,5	0,0	0,2	1,5	8,2	3,5	1,8	4,8	8,1
ethernet	2,4	2,2	-18,8	14,8	0,0	0,1	-16,0	16,7	2,4	2,4	-7,3	20,5
fir	27,0	24,7	0,0	0,0	0,0	1,9	27,0	26,2	10,7	5,2	27,7	26,5
fpu	26,7	28,3	-1,3	0,4	0,0	1,3	25,8	29,5	17,6	15,7	26,8	30,7
hyp	22,9	22,8	0,0	0,0	0,0	0,0	22,9	22,8	1,4	0,3	22,9	22,8
i2c	9,2	8,9	-7,2	2,7	0,0	0,7	2,6	12,0	16,1	14,1	1,9	11,9
i10	18,0	16,9	-0,8	1,0	0,1	1,4	17,4	18,9	19,6	13,1	16,3	20,0
idft	4,1	4,0	-14,4	13,2	0,0	0,1	-9,7	16,7	1,9	1,3	-2,2	19,2
iir	28,1	26,0	0,0	0,0	0,0	2,0	28,1	27,5	19,7	12,6	29,2	28,2
jpeg	25,3	26,3	0,7	1,9	0,0	0,4	25,9	28,0	8,3	4,3	26,2	28,0
k2	7,6	3,8	0,4	1,4	0,5	0,5	8,4	5,6	38,4	-6,6	18,4	-30,5
log2	22,7	26,0	0,0	0,0	0,0	0,1	22,7	26,1	8,4	17,1	24,1	35,3
mainpla	20,2	20,2	-46,3	31,0	0,4	1,4	-16,3	45,7	30,7	28,3	-19,4	55,0
max	0,0	0,0	-1,6	0,2	0,0	0,4	-1,6	0,6	0,9	-0,1	-1,3	0,8
mem_ctrl	9,5	8,8	-23,5	6,5	0,0	0,3	-11,7	15,0	40,9	20,7	-10,8	17,0
multiplier	29,3	29,6	0,0	0,0	0,0	0,1	29,3	29,7	9,2	7,8	31,1	30,8

pci	7,5	7,9	-0,3	0,1	0,0	0,1	7,2	8,1	6,7	5,8	8,1	8,2
picosoc	4,2	3,9	-19,8	13,6	0,0	0,2	-14,8	17,1	5,6	5,0	-6,6	23,3
sasc	6,3	5,9	-1,2	5,2	0,0	0,0	5,2	10,9	2,4	3,1	5,2	11,0
sha256	14,8	15,1	0,4	1,4	0,0	0,5	15,1	16,7	13,1	14,2	15,6	17,0
simple_spi	10,9	10,0	-2,7	8,5	0,0	0,5	8,5	18,0	8,9	6,9	2,4	20,0
sin	18,0	20,9	0,0	0,0	0,0	0,5	18,0	21,3	6,9	16,5	20,3	33,0
spi	19,2	22,5	0,1	0,2	0,0	0,1	19,3	22,7	21,1	21,4	22,4	24,3
sqrt	32,4	30,0	0,0	0,0	0,0	0,0	32,4	30,0	20,9	21,7	33,3	30,6
square	19,6	22,1	0,1	1,4	0,0	0,4	19,7	23,5	10,0	3,8	23,0	26,4
ss_pcm	1,8	1,8	-1,0	0,4	0,0	0,0	0,8	2,2	1,3	2,2	0,8	2,3
tinyRocket	18,1	16,6	-9,5	0,7	0,0	0,4	10,3	17,4	17,9	15,2	12,2	18,4
tv80	13,0	15,4	-4,3	1,6	0,1	0,9	9,3	17,5	16,7	17,9	8,8	19,3
usb_phy	9,3	8,9	-1,1	3,2	0,0	0,1	8,3	12,0	8,2	7,0	10,0	12,3
vga_lcd	0,9	0,8	-21,3	15,6	0,0	0,0	-20,3	16,3	1,1	1,4	-10,2	20,2
wb_conmax	2,2	2,8	-2,8	0,9	0,1	0,8	-0,4	4,6	5,6	9,1	0,8	5,7
wb_dma	12,7	5,6	-0,1	0,8	0,0	0,0	12,6	6,4	16,8	7,3	14,0	6,5

TABLE II. AVERAGE VALUES OF THE EXPERIMENT RESULTS

rw		**rf**		**lr**		**all 3 stage**		**ABC resyn2**		**m-resyn2**	
Area	**SA**	**Area**	**SA**	**Area**	**SA**	**Area**	**SA**	**Area**	**SA**	**Area**	**SA**
16,3	16,5	-4,9	4,1	0,1	0,4	12,0	20,5	14,0	10,9	13,1	21,3

REFERENCES

[1] M. Yaseen, S. Abd, and I. Mansoor, 'Critical factors affecting the adoption of open source software in public organizations', ijcsm, pp. 29–37, Jul. 2020, doi: 10.52866/ijcsm.2020.01.02.005.

[2] K. Kaur and A. Noor, 'CMOS Low Power Cell Library for Digital Design', VLSICS, vol. 4, no. 3, pp. 43–51, Jun. 2013, doi: 10.5121/vlsic.2013.4305

[3] W. Nebel and J. P. Mermet, Low power design in deep submicron electronics: proceedings of the NATO advanced study institute on low power design in deep submicron electronics, Il Ciocco, Lucca, Italy, 20-30 August, 1996. in NATO ASI series, no. 337. Dordrecht Boston London: Kluwer academic publ. in cooperation with NATO scientific affairs division, 1997.

[4] A. B. Kahng, J. Hu, J. Lienig, and I. L. Markov, VLSI Physical Design: From Graph Partitioning to Timing Closure. Dordrecht: Springer Science+Business Media B.V, 2011.

[5] S. Zou, J. Zhang, B. Shi, and G. Luo, 'PowerSyn: A Logic Synthesis Framework With Early Power Optimization', IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 43, no. 1, pp. 203–216, Jan. 2024, doi: 10.1109/TCAD.2023.3297069.

[6] R. K. Brayton, G. D. Hachtel, and A. L. Sangiovanni-Vincentelli, 'Multilevel logic synthesis', Proc. IEEE, vol. 78, no. 2, pp. 264–300, Feb. 1990, doi: 10.1109/5.52213

[7] J. Lamoureux and S. J. Wilton, "On the interaction between power-aware FPGA CAD algorithms," in Proc. Int. Conf. Comput.-Aided Design (ICCAD), 2003, pp. 701–708.

[8] I. A. Murashko, 'МЕТОДЫ ОЦЕНКИ РАССЕИВАЕМОЙ МОЩНОСТИ В ЦИФРОВЫХ КМОП СХЕМАХ', Доклады БГУИР, no. 1(17), pp. 100–108, 2007.

[9] Chi-Ying Tsui, M. Pedram, and A. M. Despain, 'Technology Decomposition and Mapping Targeting Low Power Dissipation', 30th ACM/IEEE Design Automation Conference, Dallas, TX, USA, pp. 68–73, 1993, doi: 10.1109/DAC.1993.203921.

[10] M. Pedram, 'ACM Transactions on Design Automation of Electronic Systems', Power minimization in {IC} design: principles and applications, vol. 1, no. 1, pp. 3–56, Jan. 1996.

[11] F. N. Najm, 'A survey of power estimation techniques in VLSI circuits', IEEE Trans. VLSI Syst., vol. 2, no. 4, pp. 446–455, Dec. 1994, doi: 10.1109/92.335013.

[12] B. Akers, "Synthesis of combinational logic using three-input majority gates," in Proc. 3rd Annu. Symp. Switch. Circuit Theory Logical Design, 1962, pp. 149–157.

[13] M. Choudhury and K. Mohanram, "Bi-decomposition of large Boolean functions using blocking edge graphs," in Proc. of the International Conference on Computer-Aided Design., 2010, pp. 586–591.

[14] S. Minato: "Fast generation of prime-irredundant covers from binary decision diagrams," IEICE Trans. Fundamentals, vol. E76-A, no. 6, pp. 967-973, June 1993.S.

[15] V. Bertacco, "Scalable Hardware Verification with Symbolic Simulation," Springer, 2005.

[16] A. Mishchenko and R. K. Brayton, "Scalable logic synthesis using a simple circuit structure," in Proc. IWLS, Vail, CO, USA, 2006, pp. 15–22.

[17] OpenABC-D. [Online]. https://github.com/NYU-MLDA/OpenABC.

[18] ABC. [Online]. https://github.com/berkeley-abc/abc.