A Bipartition-Codec Architecture to Reduce Power in Pipelined Circuits

Shanq-Jang Ruan, Rung-Ji Shang, Feipei Lai, Shyh-Jong Chen and Xian-Jun Huang
Dept. of Electrical Engineering and Dept. of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan
stj@orchid.ee.ntu.edu.tw shang@bulls.csie.ntu.edu.tw flai@cc.ee.ntu.edu.tw

Abstract

This paper proposes a new bipatition-codec architecture that may reduce power consumption of pipelined circuits. We treat each output value of a pipelined circuit as one state of a FSM. If the output of a pipelined circuit transit mainly among few states, we could partition the combinational portion of a pipelined circuit into two blocks: one that contains the few states of high activity is small and the other that contains the remainder of low activity is big. Consequently, the state transitions will be confined to the small block in most of the time. Then we replace the small block with a codec circuit, which consists of an encoder and a decoder, to reduce the internal switching activity of the block. The encoder minimizes the number of bit changes during state transitions thus the switching which propagates into decoder is reduced considerably. We present experimental results on several MCNC benchmarks and get up to 63.7% power savings by using our new architecture.

1. Introduction

Portable devices have made our work more convenient and efficient in recent years. While they require low power to prolong the operating time. Thus low power consideration has been one of the new treads for circuits design.

For low power circuits design, CMOS technology is a natural choice because of almost zero DC power dissipation. Unlike bipolar technology, the main component of power dissipation in CMOS technologies is the dynamic parts. The power dissipation in digital CMOS circuits can be written as follows: [1]

$$P_{total} = P_i(C_L \cdot V \cdot V_{dd} \cdot f_{clk}) + I_{sc} \cdot V_{dd} + I_{leakage} \cdot V_{dd} \qquad (1)$$

The first term in the right hand side represents the power dissipation in switching components, where C_L is the loading capacitance, f_{clk} is the clock frequency, V is the voltage swing, V_{dd} .is the supply voltage. The factor P_i is the probability of circuit switching. If no DC-to-DC converter is used as a power supply within the circuits, V should be equivalent to V_{dd} . The second term "$I_{sc} \cdot V_{dd}$" is due to the direct-path short circuit current when both PMOS and NMOS turn on at the same time. The last term "$I_{leakage} \cdot V_{dd}$" which arises from substrate injection and subthreshold effect, is primarily determined by the fabrication technology considerations. The dominant term of power dissipation in a well-designed circuit is the switching component, it almost accounts for over 90% of the total power dissipation. Therefore, lots of researches focus on reducing the $P_i, C_L, V_{dd}, f_{clk}$ term to obtain power reduction.

Many power optimization techniques at various levels of abstractions are proposed in recent years. On architectural level and logic level, the probability of switching activity is directly proportional to the dominant power dissipation term, thus there have been many researches on reducing switching activity for saving power. [2][3][4] proposed new sate encoding algorithms to reduce the Hamming distances between the binary representations of frequent transition pairs of states. Thus few switching will propagate into the combinational logic of FSM such that the switching activity in the internal nodes of combinational logic can be reduced. Although the complexity of combinational logic is increased after state assignment, these algorithms could reduce power consumption by 20% comparing with the originals that do not encode states with low power algorithm. Alidina, etc. proposed two precomputation architectures for low power sequential logic [5]. They selectively precompute the output values of the circuit one clock cycle before the values being required, and use the precomputed logic function to isolate part of input signals with registers from the combinational logic. If the output values can be precomputed, the original logic circuit can be "turned off" in the next clock cycle and will not have any switching activity. This approach will have much power saving effect, if few input pins can determine the correct output values. However, the precomputation architecture is not suitable for the circuits which output pins must be determined by most input pins. Luca and Giovanni used gated-clock method for low power FSMs [6]. They exploits the concept of self-loop, an idle condition for a Moore machine. When the machine is in a self-loop condition, the next state is the same as the present state, and the output do not change in the next cycle. Therefore clocking the machine only wastes power. The technique can obtain power saving by stopping the clock of the circuit during the self-loop period. FSMs with a lot of self-loops can perform well using gated-clock approach, but some FSMs transit mainly among some states can not benefit from these methods. [7] decomposed a FSM into a number of coupled submachines so that there is a high probability and state transitions will be confined to the smaller submachines most of the time.

This paper will proposes a bipartion-codec architecture (bipartitioned coding-decoding architecture), a novel

0-7803-5832-5/99/ $10.00 © 1999 IEEE

architecture for low power pipelined circuits. We treat each different output as a corresponding state of FSM in our architecture and a FSM implicitly specifies a state transition graph (STG). Then the bipartition algorithm is exploited to measure the probabilistic behavior and partition the STG. After performing our bipartition algorithm, the original STG is partitioned into two STGs: one contains fewer states of high activity and the other contains more states of low activity. Obviously, the former will be active at most of the time and dominate the power dissipation. We use codec structure to implement the high active portion for minimizing the circuit switching.

The presence of bipartition-codec architecture has two important cosiderations. First, when we have a FSM with n different states, it will have 2^n-2 different bipartitions. It is impossible to synthesize and estimate the power of all bipartitions for choosing the best case. Second, detecting clustering conditions requires some computation to be performed by additional circuitry. The additional circuitry also dissipates power. In this paper, we propose a heuristic and fast algorithm to partition circuit for our bipartition-codec architecture, then combine the encoder circuit with the clustering detecting circuitry for decreasing circuit area.

The remainder of this paper is organized as follows. Section 2 proposes a biparition-codec architecture and its working principle. Bipartition algorithm is described in section 3. In section 4, we discuss the codec structure and state encoding for low power. In this section we also discuss our architecture without using codec structure for comparison. In section 5, experimental results are presented, that confirm the effectiveness of our new architecture. Finally we summarize the conclusion of the paper in section 6.

2. Bipartition-Codec Architecture

A. Preliminary

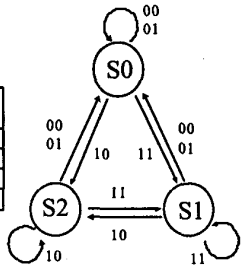
Before describing the bipartition-codec architecture in detail, let us consider the relationship between the FSM and combinational circuit. By paying attention to the output end of combinational circuit, each output value can be regarded as one state of a Moore FSM. However, the combinational circuits do not need information of the past cycles to determine output value in the present cycle. The transition of the states only depends on input patterns. From all different states, they will arrive at the same state if the same input patterns are used to trigger the machine.

Definition 1. *A combinational circuit can be modeled by a four-tuple (I, O, S, F) where I is the set of inputs, O is the set of outputs, S is the set of states. Every state S_i has a unique code e_i, The output function is defined as $O_t = F(I) = e_i$, where $i=1, ..., |S|$.*

Example 1. *In Fig. 1 (a), there are three different output values 10, 00, 11 in the truth table, and the corresponding states are: S0, S1, S2. Since the output values can change to any valid output values at the next cycle, certainly including itself, the state transition graph (STG) is a complete connective graph.*

Input		Output		State
a	b	f_1	f_2	
0	0	1	0	S0
0	1	1	0	S0
1	0	1	1	S2
1	1	0	0	S1

(a)



(b)

Fig. 1.(a) the truth table of a combinational circuit (b) the state transition graph

The state transition probability provides a measure of the most frequently traversed transitions. Markov chain is usually used to model the transition behavior of FSM. It is not necessary for combinational circuit, because the output values of combinational circuit completely depend on input patterns. Thus we can only consider the magnitude of state probability to figure out the probabilistic behavior.

Definition 2. *The state probability of combinational circuits is the number of different input patterns of a particular state divided by the number of total different input patterns.*

Example 2. *In Example 1, we assumed the probability distribution of output patterns is uniform. The probability of each state is: Prob(S0) = 2/4, Prob(S1) = 1/4, Prob(S2) = 1/4.*

B. Bipartition-Codec Architecture

Consider the circuit of Fig. 2. A pipelined stage is a combinational logic block separated by distinct registers.

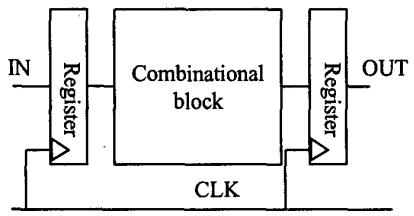


Fig. 2. A combinational pipelined circuit

The bipartition architecture without codec structure is shown in Fig. 3. The combinational logic portion in Fig. 2 is partitioned into two groups. One that includes some states of high activity is small called $Group_1$ and the other that includes the remainder of low activity is big called $Group_2$. The two groups work in turns. The GCB is a precomputation block for only one block is working at the same time. Then we replace the $Group_1$ and GCB with a codec structure, which consists of an encoder and a decoder. The encoder not only encodes the

output state but also issues a signal SEL to choose which group to compute. The SEL function also controls the MUX to select correct outputs. The bipartition-codec architecture is shown in Fig. 4.

In Fig. 4, the input IN feeds into the Encoder and the registers of $Group_2$. Encoder issues a SEL function to select which group to compute. When SEL=1, the gated clock CLK1 will be active, and CLK2 stopped. The output values of encoder pass through the register word and propagate into the decoder. Finally the output values of the decoder directly connect to the output ports by multiplexer. At this moment, the input register word of $Group_2$ is disabled by the complementary value of SEL, it prevents input signal switching from entering into the $Group_2$. So there is no power consumption in the $Group_2$.

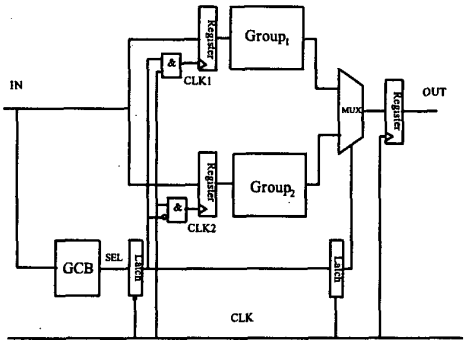


Fig 3. A bipartition architecture without code structure

The low enable latch in our architecture is needed for a correct behavior, because SEL may have glitches that must not propagate to the AND gate when global clock is high. The other latch, which behind low-enable one is transparent when clock is high. These two latches work as a master-slave flip-flop. They save the SEL function during a period of the global clock so that the multiplexers can select the correct outputs.

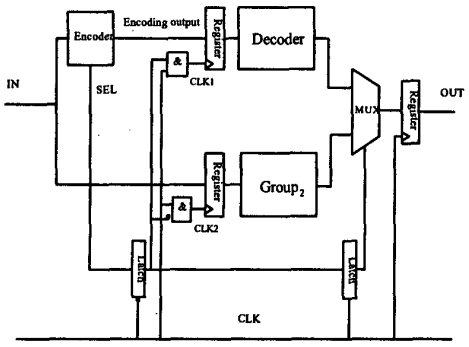


Fig 4. A new bipartition-codec architecture

We can obtain power reduction when most of the input conditions corresponding to SEL=1. In other words, if the smaller combinational block $Group_1$, which we replace by an encoder and a decoder, is active of the most time, the effect of our architecture on power saving is better. In the following section, we will describe the detail of our low power architecture.

Example 3. *The given example illustrates the cluster characteristic for bipartiton architecture. Consider the benchmark sao2 of MCNC which has 10 input pins and 4 output pins. The state list of sao2 is shown below:*

State no	State pattern	count
S_1	0000	513
S_2	0010	257
S_3	0011	219
S_4	0100	8
S_5	1000	7
S_6	1100	6
S_7	0001	5
S_8	0101	4
S_9	1001	3
S_{10}	1101	2

State pattern is the original output pattern of the benchmark sao2. The column of count stands for the number of different input patterns corresponding to a particular state. In this example, if we cluster S_1, S_2 and S_3 into $Group_1$, the probability of $Group_1$ is 989/1024. For our architecture, the meaning is that most transitions will occur in the small $Group_1$.

3. Bipartition Algorithm

The objective of the bipartition algorithm is to partition a combinational circuit into $Group_1$ and $Group_2$. The $Group_1$ is small but with high active probability and the $Group_2$ is big with low active probability. Unfortunately, we can not estimate power and area of circuits before synthesis, proposing an optimal bipartition algorithm for achieving the maximum power saving in our architecture become very difficult, so a heuristic algorithm must be used. In this section, we formulate the bipartition problem and propose a heuristic algorithm for our low power architecture.

A. Problem Formulation

The problem of bipartition can be regarded as to bipartite the circuit which implementation dissipates the minimum power. The $Group_1$ dominates the most power consumption of the circuit since most of the time the $Group_1$ is at work in our architecture. Observe that minimizing the gate count of $Group_1$ for reducing the circuit switching plays a significant role in lowering the power dissipation. Thus the problem can be simplified as follows: First we hope the sum of the state probabilities in $Group_1$ as large as possible; second the gate counts of $Group_1$ after synthesis as small as possible. The problem of finding a bipartition can be formalized as follows:

$$Max \frac{prob(Group_1)}{G\_count(Group_1)} \qquad (2)$$

In (2), $\boldsymbol{Prob(Group_1)}$ is the summary of state probabilities in $Group_1$ and $\boldsymbol{G\_count(Group_1)}$ represents the gate count of $Group_1$. Since the gate count of $Group_1$ is hard to estimate before synthesis. We pay our attention to the relation between state number and gate count. The less state number we select, the more don't care conditions we have. The size of a cover is the number of its implicants. Don't care conditions can be effectively used to reduce the size of a cover of an

incompletely specified function [9]. Thus we assume the gate count is proportional to the numbers of states. The formulation (2) can be rewritten as:

$$Max \frac{prob(Group_1)}{State\_No(Group_1)} \qquad (3)$$

Example 4. *From example 1, the truth table can be determined by a two-input, two-output function, where f_1 = a'+b'; f_2 = ab'. We need an OR gate and an AND gate to implement the two functions. If we partition the STG into {S0} and {S1, S2}, input patterns 10, 11 become don't care conditions for {S0}. The function of group {S0} will become f_1 = 1; f_2 = 0. Obviously, the gate counts will reduce to 0 due to the reduction of implicants.*

B. Heuristic Algorithm

The bipatition algorithm is used to select the states of $Group_1$, and the remainders are put in $Group_2$. The input of our algorithm is a set of individual state, S1, S2, .., Sn. In line 1, MinCount is equal to the half number of input patterns since Count(S) is a function that calculates the number of input patterns that generate all output values in S. We determine the AvgCount by dividing Count(S) by state number *n*. Clearly, AvgCount is an average number of the input patterns of one state and serves as a threshold value while selecting states in this algorithm. Ceil function is used to exclude the states which are greater than the average state input count. Then, we initialize the selected to null and the Count of selected to zero. From lines 4 to 9, the **foreach** loop expands the Selected set by adding the states of which the state count are greater than AvgCount. From lines 10 to 14, the **while** loop continues selecting states until the constraint on MinCount is satisfied. Otherwise, it could not obtain better power saving effect due to its small group probability. In most cases, the while loop will skip since most benchmarks have cluster and selfloop characteristic.

```
Bipartition ( S = {S1, S2, ...,Sn})
{
1    MinCount = Count(S)/2;
2    AvgCount = ⌈Count(S)/n⌉;
3    selected = NULL; Count = 0;
4    foreach ( Si∈S ){
5     if (Count(Si) > AvgCount) {
6     S = S-Si;
7     selected = selected ∪ Si;
8     Count = Count(Si);
9    }
10   while( Count ≤ MinCount){
11    NewState=sel_max_Count(S);
12    selected=selected ∪ NewState;
13    Count=Count+Count(NewState);
14   }
15   return selected;
}
```

Fig 5. bipatition algorithm for bipartition-codec architecture

4. Codec Structure and State Encoding

A. Codec structure

According to our bipartition algorithm, $Group_1$ includes few highly active states. That means $Group_1$ generates fewer different output patterns than the original circuit. Consequently, we can decrease the output pins of encoder to $\log\lceil No\_of\_States \rceil$ for scaling down the circuit complexity. In order to select which group is active, SEL function plays a significantly role in our architecture. The Encoder also generates the SEL function in our design. Because that the function shared most of the product terms of the Encoding portion. Namely, just few additional logic will be added to the Encoder for performing the SEL computation. Decoder is used to maintain functional equivalence. We insert registers between the Encoder and the Decoder for synchronization and reducing input toggles of the Decoder. If the value of the output of the encoder is the same as the previous value, there is no toggle in the Decoder.

Example 5. *Consider the sao2 and misex3 in PLA MCNC. We cluster the top two probabilities of states together by our bipartition algorithm. In the following table we show the contrast between the Group1 without codec and with codec structure in area.*

Benchmark	Bipartition		Bipartition-codec	
	SEL	$Group_1$	Encoder	Decoder
sao2	50	12	62	0
misex3	759	97	799	3

The table shows that if we combine the SEL function into the Encoder, the area will just increase little due to the sharing of most product terms. Notice that the smaller Decoder, comparing to $Group_1$, imply less switching activity.

B. State encoding

At the gate level of abstraction, average power dissipation is proportional to the average switching activity. A good state assignment algorithm can significantly reduce switching activity of $Group_1$. Many state assignment algorithms such like [2], [3] and [4], minimize the Hamming distance between the codes of the states with high transition probability.

As we discussed in section 2, STG (state transition graph) for combinational logic circuit is a complete graph. Thus, transition probability information for each edge in the STG can then be determined by modeling the STG as a Markov chain [4]. For a transition from state S_i to state S_j, weight $p_{i,j}$ on the corresponding probabilities depends on the probability distribution of the inputs, that is initially known. Assume for simplicity that all inputs are equiprobable. In order to find the probability of a transition without any condition, we need to know the state probability q_i that represents the probability of the machine being in a given state S_i. Namely, the total transistion probabilities we are looking for are

$$r_{i,j} = p_{i,j} \cdot q_i \qquad (4)$$

Calculating state probability for pipelined circuits is simple. In a pipelined circuit, $p_{i,j}$ can be simplified to p_j because no matter which state the edge starts, the input conditions are the same. It means if we model the probabilistic behavior of a combinational logic using a Markov chain, each row in the state transition matrix will be the same. In fact, each row stands for the stationary vector of the transition matrix. We then directly assign conditional probability p_i to state probability q_i. The total transition probabilities can be simplified to

$$r_{i,j} = p_j \cdot p_i \qquad (5)$$

After calculating the transition probabilities of each pair of states, we transform the STG into a weighted undirected graph. We sum up the total probabilities between any two states into an undirected edge $w_{i,j}$. Note that self-loops can be eliminated because we only need the information of each pair of states for state assignment.

Example 6. *In Fig. 1, the probability of each state is calculated in example 2. We can calculate the total probability of each edge by equation (5). The* $r_{0,1} = r_{1,0} = 2/4 \cdot 1/4 = 2/16$, *then we obtain* $W_{0,1} = 4/16$ *by summing up* $r_{0,1}$ *and* $r_{1,0}$. *The total probabilities and weighted graph of example 1 is illustrated in Fig. 6.*

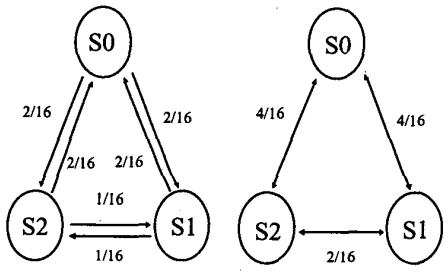


Fig 6. (a) total transition probabilities (b) weighted graph

The problem of finding a state encoding that results in minimum switching activity can then be formalized as: Find a set of Boolean row vectors $(e_i^1, \ldots, e_i^{n_{var}})$, $i=1, 2, \ldots, n_s$, n_s is the number of states and n_{var} is the number of state variables used ($\lceil \log_2 n_s \rceil$), that are solutions to the problem:

$$Min\left(\sum_i \sum_j w_{i,j} \sum_{l=1}^{n_{var}} e_i^l \oplus e_j^l \right) \text{ such that} \qquad (6)$$

$$\sum_{l=1}^{n_{var}} e_i^l \oplus e_j^l \geq 1 \quad \forall s_i, s_j, s_i \neq s_j, \qquad (7)$$

where $\oplus$ represents the *XOR* operation, $w_{i,j}$ is the weight on the edge between states S_i and S_j. Because the weighted graph is a complete graph, there will be $C(n_s,2)$ edges. The cost function evaluates how well to assign adjacent codes to state of high-probability transitions. Equation (7) expresses that no two states can be allowed to have the same code. We selected the heuristic algorithm in [4] for encoding the states in $Group_1$.

5. Experimental Results

The bipartition and codec algorithm have been implemented in C++ on a SUN Sparc station. We used SIS [9] to synthesize our partition results and estimate the power by PowerMill. The random logic circuits taken from MCNC PLAs are used to demonstrate our algorithm. In the experiment, 5v supply voltage and a clock frequency of 20MHz was assumed. The rugged script of SIS was used to optimize most of the benchmarks. Due to the size of the circuits, only few examples of the simple script were used. These circuits were marked by *.

Circuits	Input No.	Output No.	$Group_1$		$Group_2$	
			State No.	P_{G1}	State No.	P_{G2}
sao2	10	4	3	0.966	7	0.034
misex3	14	14	2	0.808	1039	0.192
table3*	14	14	4	0.925	66	0.075
misex1	8	7	2	0.625	9	0.375
table5*	17	15	4	0.906	56	0.094
sqrt8	8	4	8	0.75	8	0.25
con1	7	2	2	0.688	2	0.312
rd84	8	4	3	0.711	6	0.289
bw	5	8	7	0.5	16	0.5

Table 1. The probabilities of bipartition

Table 1 presents the performance of our bipartition algorithm on a subset of the MCNC PLAs, it includes the state numbers in $Group_1$ and $Group_2$, and their group probabilities. In all of the 9 benchmarks in Table 1, the state number of $Group_1$ is less than that of $Group_2$, but probabilities of $Group_1$ are far bigger than $Group_2$'s except for the last benchmark 'bw'. Because only a few states exist in $Group_1$, the encoder uses a small number of output pins to encode those few states. The output number of the encoder can be determined by the function $\log\lceil No\_of\_States \rceil$, where '*No_of_states*' is the 'State No.' of $Group_1$ in table 1. Obviously, which is much smaller than the number of the original output. Therefore, the 9 benchmarks are suitable for our architecture because there is a small group dominating the circuit behavior in most of the time.

In Table 2, we record the area and power dissipation of the 9 benchmarks implemented by conventional architecture and our bipartition-codec architecture. 'Original' column shows the area and the power dissipation of the conventional architecture. The 'Bipartition-Codec' consists of the two subcolumns, 'Area' and 'Power'. There are four columns in the subcolumn 'Area'. The 'E & D' shows the area of codec structure. The area $Group_2$ is put in the column 'G2'. The total area of the overhead is in the column 'O' which contains latches, registers, multiplexer and two additional gates shown in Fig. 4. T_A is the total area synthesized in bipartition-codec architecture. There are 5 subcolumns in column 'Power'. The 'E' and 'D' record the power dissipation in the encoder and decoder, respectively. The 'G2' is the power dissipation in the

$Group_2$. The 'O' stands for the power dissipation of the overhead in bipartition-Codec architecture. 'T_P' and 's%' represent the total power dissipation and the power saving percentage. Observing the 'Bipartition-Codec' part, in most cases the power consumption in the encoder is bigger than that of the decoder and the $Group_2$. The reason is that the additional pin 'SEL' in the output of encoder. Whenever the encoder must be active to ensure whether the small group or the big one is active. Because of self-loop effect and small area in the decoder, the decoder hardly dissipates power. The big group consumes small power dissipation because it is often at idle. Because the small group 'E & D' is active most of the time and the area is smaller than the original combinational block, the power dissipation can be considerably reduced. The highest power-saving percentage is 63.7%, and the average power saving approximates 31.2% (area is in 128 μm^2, power is in μW) .

6. Conclusion

In this paper, a new bipartition-codec architecture for pipelined circuits with low power consideration is proposed. The experimental results show that our architecture can effectively save the power in a large class of random logic circuits.

We treat each output value of a combinational block in a pipelined circuit as a state in an FSM. When the output values of some benchmarks cluster around only a few special output values, our algorithm could effectively extract the small group that include these few special output values. We apply codec structure and state encoding techniques to reduce the switching activity in the high active $Group_1$. Although the encoder must be active at any time, it dissipates little power. The $Group_2$ is often at idle and dissipates little power in average. Therefore, large power reduction is obtained for large pipelined circuits, where the probability of being in $Group_1$ is high. The only disadvantage of this bipartition method is that the critical path delay is increased by the extra Encoder block.

Circuits	Original		Bipartition-Codec								
			Area				Power				
	Area	Power	E &D	G_2	O	T_A	E	D	G_2	O	T_P (s%)
sao2	571	3361	137	253	335	725	342	14	20	845	1255 (63.7)
misex3	2557	9941	802	1015	488	2305	2800	4	744	1527	5075 (43.9)
table3*	2842	6564	995	2580	509	4084	3207	28	663	1229	5127 (40.5)
misex1	340	2238	70	72	299	441	193	0	132	1230	1555 (31.7)
table5*	3203	7946	1195	2536	581	4312	4117	22	344	1265	5748 (27.6)
sqrt8	371	2541	144	79	319	542	526	5	78	1333	1942 (23.6)
con1	209	1506	88	21	233	342	278	0	16	888	1182 (21.5)
rd84	531	3176	273	208	293	774	969	15	225	1437	2646 (16.7)
bw	694	4443	244	249	287	780	238	191	880	2599	2908 (12)

Table 2. The simulation results

References

[1] Chandrakassan, S. Sheng, and R. W. Brodersen, "Low-Power CMOS Digital Design," *IEEE J. Solid-State Circuits, vol. 27, No. 4, pp. 473-483, Apr. 1993.*

[2] E. Olson and S. M. Kang, "Low-Power State Assignment for Finite State Machine," *1994 International Workshop on Low Power Design, pp. 63-68.*

[3] Hachtel, Mariano Hermida De La Rica, Abelardo Pardo, Massimo Poncino, and Fabio Somenzi, "Re-encoding Sequential Circuits to Reduce Power Dissipation," *1994 International Workshop on Low Power Design, pp. 69-74.*

[4] Luca Benini and Giovanni De Micheli, "State Assign for Low Power Dissipation," *IEEE J. Solid-State Circuits, vol. 30, pp. 258-266, Mar. 1995.*

[5] Alidina, Jose Monteiro, Srinivas Devadas, Abhijit Ghosh and Marios Papaefthymiou, "Precomputation-Based Sequential Logic Optimization for Low Power," *IEEE Tran. on VLSI, vol. 2, No. 4, Dec. 1994.*

[6] Luca Benini and Giovanni De Micheli, "Automatic synthesis of Low-Powe Finite-state Machines," *IEEE Tran. on* Computer-*Aided Design of Integrated Circuits and Systems, vol. 15, No. 6, Jun. 1996.*

[7] Sue-Hong Chow, Yi-Cheng Ho, TingTing Hwang, "Low Power Realization of Finite State Machines – A Decomposition Approach" *ACM Tran. on Design Automation & Electronic Systems, vol. 1, No .3, pp. 315-340 July 1996.*

[8] G. De Micheli, "Synthesis and Optimization of Digital Circuits," *McGraw-Hill, New York, 1994.*

[9] SIS: A System for Sequential Circuit synthesis is implementd by Electronics Research Laboratory in Department of EE and CS, University of California, Berkley, 4 May 1992.