第26卷 第7期 半 导 体 学 报 Vol. 26 No. 7
2005年7月 CHINESE JOURNAL OF SEMICONDUCTORS July, 2005

Energy Recovery Capacitance Coupling Logic and Its Synthesis Methodology*

Yang Qian and Zhou Runde

(Institute of Microelectronics, Tsinghua University, Beijing 100084, China)

Abstract: A novel energy recovery logic style ERCCL (energy recovery capacitance coupling logic), which has good energy performance compared to the conventional CMOS logic and other advanced energy recovery logic, is proposed. ERCCL uses capacitance coupling to perform a logic function, so it can energy-efficiently implement a high fan-in complex logic in a single gate. ERCCL is also a type of threshold logic. The gate count of a system based on ERCCL can be significantly reduced, which, in turn, will decrease the energy loss. A threshold logic synthesis methodology for ERCCL is also presented. MCNC benchmarks are run through the proposed synthesis methodology. The results indicate that about an 80% reduction in gate count can be obtained when compared with the synthesis results of SIS.

Key words: energy recovery; threshold logic; logic synthesis; capacitance coupling; CMOS circuits
EEACC: 1265A; 1130; 2570D
CLC number: TN432 **Document code:** A **Article ID:** 0253-4177(2005)07-1334-06

1 Introduction

Power consumption has been a critical parameter in digital design since the early 1990s. Methods for power reduction based on energy recovery techniques have been reported recently[1]. Numerous energy recovery logic styles have been proposed[2~4], assuming that an energy recovery system can be designed by replacing conventional logic gates with their energy recovery counterparts. The disadvantage of this approach is that it results in an extremely large pipeline depth, requiring a large number of delay matching latches that increase non-adiabatic dissipation[5]. In this paper, a novel energy recovery system design approach is presented, which firstly transforms a conventional logic network into a threshold logic network then replaces each threshold gate with a novel energy recovery logic gate ERCCL[6] (energy recovery capacitance coupling logic). Threshold logic is attractive as it has both reduced logic depth and gate count when compared to conventional logic[7]. ERCCL uses capacitance coupling, rather than a CMOS switches network, to perform a logic function. Thus ERCCL can energy-efficiently implement a complex logic with a high fan-in in a single gate. The previously proposed energy recovery logics cannot energy-efficiently implement high complex logic in a single gate because the energy of the internal node in complex switched network cannot be efficiently recovered[5]. ERCCL implements threshold logic by capacitance coupling used. The proposed design approach based on ERCCL can mini-

* Project supported by the National Natural Science Foundation of China (No. 59995550-1)

Yang Qian male, was born in 1977, PhD candidate. His research interest is in low power CMOS circuits design.

Zhou Runde male, was born in 1945, professor, advisor of PhD candidates. His research interests are in low power IC design and embedded system structures.

Received 23 July 2004, revised manuscript received 17 February 2005

mize the total non-adiabatic loss by minimizing the number of logic gates. For maximally reducing gate count of a logic network, a threshold logic synthesis methodology for ERCCL is developed, which reduce gates count significantly compared to the conventional logic synthesis methodology.

2 ERCCL

Figure 1 shows a basic ERCCL gate and its four-phase of power-clock. The ideal power clocks and auxiliary clocks are shown in Fig. 2. Power clocks ϕ_i have four phases, i. e. evaluate, hold, (energy) recovery, and wait. The auxiliary clock CK_i is held high during the wait phase of ϕ_i to avoid the nodes L, R, L_1 and R_1 floating all the time.

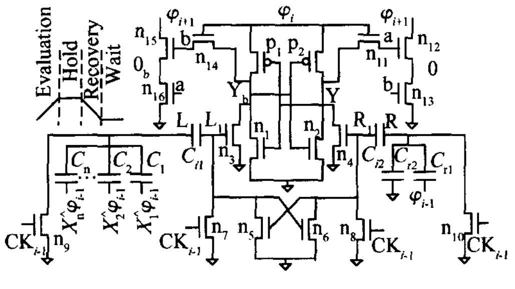

Fig. 1　ERCCL gate and its four-phase power-clock

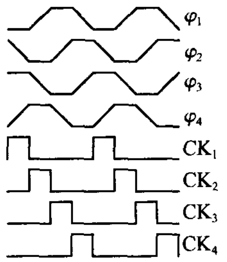

Fig. 2　Power-clocks and auxiliary clocks

A threshold logic gate has n binary inputs X_1, X_2, ⋯, X_n and a binary output y. The gate is specified by the threshold T and a group of weights w_1, w_2, ⋯, w_n. If $\sum_{i=1}^{n} w_i X_i \geq T, y = 1$; otherwise $y = 0$. ERCCL implements threshold logic function using the capacitance coupling technique.

The operation of ERCCL is described in Ref. [6]. The non-adiabatic loss of an ERCCL gate in the worst case is $1/2 C_{L_1(R_1)} V_{tn}^2 + C_{Y(Y_b)} V_{tp}^2 + 1/2 C_{a(b)} (V_{dd} - V_{tn})^2 + 4 C_g V_{CK}^2$[6]. $C_{L_1(R_1)}$ is the capacitance of node L_1 (R_1), $C_{Y(Y_b)}$ is the capacitance of node Y(Y_b), $C_{a(b)}$ is the capacitance of node a(b), and C_g is the gate capacitance of transistors n_7 ~ n_{10}. V_{tn} and V_{tp} are the threshold voltage of nMOS and pMOS, respectively. V_{CK} is the voltage swing of the auxiliary clocks. Thus the non-adiabatic loss of an ERCCL gate is independent of the logic complexity and load, therefore depends only on the total gate count.

Additionally ERCCL reduces adiabatic loss in the circuit level as it implements logic through capacitance coupling rather than a switches network, which leads to less turn-on resistance[6], ERCCL can greatly reduce dissipation in the architecture level with less gate count by implementing high fan-in complex threshold logic energy-efficiently in a single gate.

To demonstrate the energy efficiency of ERCCL, a 4bit adder of ERCCL, a conventional CMOS and a ECRL[4], and a NERL[3] are designed and compared. The conventional CMOS adder, ECRL adder, NERL adder all take CLA (carry-lookahead adder) architecture[4], while ERCCL takes threshold logic adder architecture[6]. HSPICE simulation based on CSMC 0.6μm DPDM technology parameters are performed. The supply voltage is 5V, and the voltage of the ERCCL auxiliary clocks is 1V. Each primary output is connected to a 100fF load. The simulation results are listed in Table 1.

Table 1　Power loss and area costs for various 4-bit adders

	CMOS	ERCCL	ECRL	NERL
20MHz	237μW	63μW	130μW	104μW
50MHz	595μW	247μW	407μW	330μW
100MHz	1183μW	712μW	1090μW	913μW
Gate count	27	27	47	47
Transistor count	140	372	242	490

Compared to the ECRL, NERL, and conventional CMOS, ERCCL has lower power loss. At 20 ~ 100MHz, the ERCCL achieves a power savings of 74% ~ 40% compared to the conventional

CMOS, 52%～35% compared to the ECRL, and 40%～23% compared to the NERL. However, ERCCL pays more area cost. Including the area cost of coupling capacitors, the ERCCL adder costs about 3 times more area than the conventional CMOS and costs about 60% more than the ECRL. The area cost of the ERCCL is less than that of the NERL. The transistor counts of various adders is listed in Table 1.

ERCCL, ECRL, and NERL all use the same four-phase power clock. Power clock loss is not considered in simulation. The energy efficiency of the four-phase power clock can reach about 80%[8]. If power clock loss is added, the ERCCL can achieve a power saving of 68%～25% compared to the conventional CMOS at 20～100MHz.

Although the ERCCL adder needs less gate count and less circuit level, its latency is a little more than the NERL adder and ECRL adder, since its gate needs 3 power clocks to work. The latency of the ERCCL adder is 1.5 cycles while that of NERL and ECRL is 1.25 cycles. In addition, as coupling capacitors can filter some noise and glitch, the robustness of ERCCL is better than that of ECRL and NERL.

3 Threshold logic synthesis for ERCCL

Because maximally reducing gate count in an ERCCL circuit can greatly reduce power loss, a high efficiency threshold logic synthesis methodology is important for ERCCL. In this section, a threshold logic synthesis methodology is developed.

3.1 Background

In this section, a concept and a theorem, which are important for threshold logic synthesis, are presented.

Binate function: If there exists a disjunctive or conjunctive expression of $f(x_1, x_2, \cdots, x_m)$ in which x_i appears only in uncomplemented (complemented) form, f is said to be unate in x_i. A function, which is unate in all variables, is called a unate function; otherwise, it is called a binate function. Binate functions are important for threshold logic as every binate function is not a threshold function[7].

Theorem 1: Given an irredundant sum-of-products (SOP) expression for function $f(x_1, x_2, \cdots, x_n)$ (in which no cube includes any other cube), if there exists two cubes a and b which both are composed of more than two variables and have no common variable, f is not a threshold function.

Proof: Assume $a = x_{i1} x_{i2} \cdots x_{im}$, $(x_{i1}, x_{i2}, \cdots, x_{im} \in \{x_1, x_2, \cdots, x_n\})$, $b = x_{j1} x_{j2} \cdots x_{jk}$, $(x_{j1}, x_{j2}, \cdots, x_{jk} \in \{x_1, x_2, \cdots, x_n\})$, $\{x_{i1}, x_{i2}, \cdots, x_{im}\} \cap \{x_{j1}, x_{j2}, \cdots, x_{jk}\} = \Phi$. If f is a threshold function, there exists a set $\{w_1, w_2, \cdots, w_n, T\}$ satisfying Eqs. (1) and (2).

$$\sum_{i=1}^{n} w_i x_i \geq T, \quad f = 1 \tag{1}$$

$$\sum_{i=1}^{n} w_i x_i < T, \quad f = 0 \tag{2}$$

Assume $x_{i1} = 1$, $x_{j1} = 1$, other variables of f are 0, in this case $f = 0$, we have

$$w_{i1} + w_{j1} < T \tag{3}$$

Assume $x_{i2} = \cdots = x_{im} = 1$, $x_{j2} = \cdots = x_{jk} = 1$, other variables of f are 0, in this case $f = 0$, we have

$$w_{i2} + \cdots + w_{im} + w_{j2} + \cdots w_{jk} < T \tag{4}$$

From Eqs. (3) and (4), we have

$$w_{i1} + w_{i2} + \cdots + w_{im} + w_{j1} + w_{j2} + \cdots w_{jk} < 2T \tag{5}$$

Assume $x_{i1} = x_{i2} = \cdots = x_{im} = 1$, other variables of f are 0, in this case, $f = 1$, we have

$$w_{i1} + w_{i2} + \cdots + w_{im} \geq T \tag{6}$$

Similarly, we have

$$w_{j1} + w_{j2} + \cdots + w_{jk} \geq T \tag{7}$$

From Eqs. (6) and (7), we obtain

$$w_{i1} + w_{i2} + \cdots + w_{im} + w_{j1} + w_{j2} + \cdots w_{jk} \geq 2T \tag{8}$$

Equations (5) and (8) are contradictory. Therefore f is not a threshold function.

3.2 Algorithms

Figure 3 gives a high-level overview of the main steps of the proposed methodology. The input to the proposed methodology is a multi-output combinational logic function; the output is a functionally equivalent threshold network. The weighted-sum restriction can be specified during threshold network synthesis.

For clarification of description, some definitions are defined as follows. A threshold node is a node which is a threshold function and also meets the weighted-sum restriction. A non-threshold node is a node which is not threshold node. A theorem 1 node is any node satisfying theorem 1. A binate node is a node which is a binate function. A node, which is neither a binate node nor a theorem 1 node, is called an ordinary node. A threshold kernel is a kernel which is a threshold function.

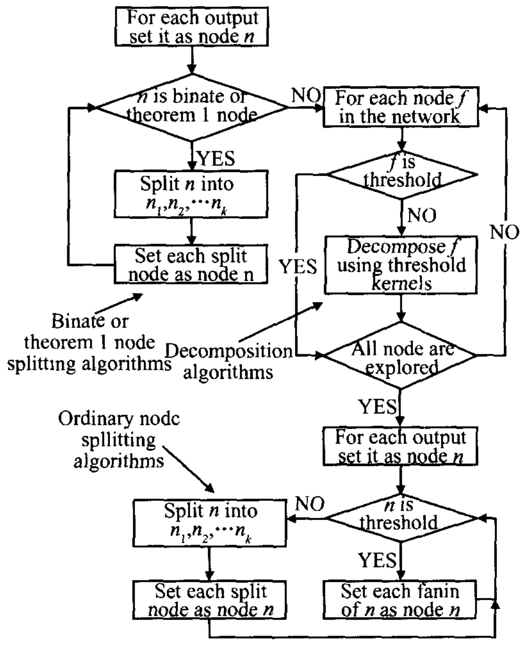
For each output set it as node n
n is binate or theorem 1 node
NO
YES
Split n into $n_1, n_2, \cdots n_k$
Set each split node as node n
For each node f in the network
f is threshold
NO
YES
Decompose f using threshold kernels
NO
All node are explored
YES
Binate or theorem 1 node splitting algorithms
Decomposition algorithms
Ordinary nodc spllitting algorithms
For each output set it as node n
n is threshold
NO
YES
Split n into $n_1, n_2, \cdots n_k$
Set each split node as node n
Set each fanin of n as node n

Fig. 3　High-level overview of the proposed methodology

The key idea of the methodology is firstly to split each binate node or theorem 1 node n into multiple small nodes $\{n_1, n_2, \cdots n_k\}$ using efficient heuristics, because binate nodes and theorem 1 nodes must be non-threshold nodes. Secondly, form an OR gate with these split nodes ($n = n_1 + n_2 + \cdots n_k$). After the split, decompose each non-threshold node with threshold kernels (if it exists). Then split the rest of the non-threshold nodes into small nodes and form an OR gate with the split nodes until all nodes of the network are threshold nodes. After threshold logic synthesis, every threshold gate is mapped to a ERCCL gate and latches are added to accommodate gate level pipelining introduced by the use of energy recovery logic gates. We describe each node split algorithm and decomposition algorithm in detail (in pseudo C-code) in the following. Variable W, which is used in the following algorithms, is the weighted-sum restriction on a threshold gate.

```
Decomposition algorithm:
$f$ is one non-threshold node
Decomposition ($f$) {
$g \leftarrow$ find_best_kernel($f$)
if $g = \Phi$ then
    return $f$
else
    $(q, r)$ = weak_div($f$, $g$)
    return (decomposition ($q$) $g$ + decomposition ($r$))
}
```

Decomposition, which relies on kernels, is the process of deriving an optimal factored form of a given logic function[9]. The proposed algorithm is described in recursive form. Firstly, we compute all of the threshold kernels of f utilizing rectangle covering algorithms, which is both fast and effective[9]. Procedure find_best_kernel chooses one threshold kernel with the most variables as the divisor g. If there does not exist any threshold kernels of f, f is not decomposed. Otherwise, quotient q and remainder r are generated by a fast weak division algorithm (implemented by procedure weak_div)[9]. q and r are recursively decomposed with the same decomposition algorithms. Finally, we obtain an optimal factored form of f.

```
Binate node splitting algorithm:
$k \leftarrow 1$
$n$ is one binate node
while $k < W$ and $n$ is binate node do
    $\{L, R\}$ = binate_split_two($n$)
    $n_k \leftarrow L$
    $k \leftarrow k + 1$
    $n_k \leftarrow R$
    $n \leftarrow n_k$
```

Procedure binate _ split _ two splits n into two nodes, L and R. L is a unate node, which does not satisfy theorem 1 and has as many cubes of n as possible. R includes all the rest cubes of n excluding the cubes of L. The process stops until all of the split nodes are unate nodes or the number of the split nodes exceeds W.

Theorem 1 node splitting algorithm:

```
$k \leftarrow 1$
$n$ is one theorem 1 node
while $k < W$ and $n$ is theorem 1 node do
    $\{L, R\}$ = theorem1 _ split _ two ($n$)
    $n_k \leftarrow L$
    $k \leftarrow k + 1$
    $n_k \leftarrow R$
    $n \leftarrow n_k$
```

Procedure theorem1 _ split _ two splits n into two nodes, L and R. L does not satisfy theorem 1 and has as many cubes of n as possible. R includes all the rest cubes of n, excluding the cubes of L. The process stops until all of the split nodes are not theorem 1 nodes or the number of the split nodes exceeds W.

Ordinary node splitting algorithm:

```
$n$ is one non-threshold node
$k \leftarrow 1$
$n_k \leftarrow n$
while $k < W$ and $n_1, n_2, \cdots, n_k$ are not all threshold node do
    for $i = 1$ to $k$ do
        if $n_i$ is not threshold node then
            $\{L, R\}$ = threshold _ split _ two ($n_i$)
            $n_i \leftarrow L$
            $k \leftarrow k + 1$
            $n_k \leftarrow R$
```

Procedure threshold _ split _ two splits n_i into two nodes L and R, using the most frequently appearing variable (excluding the variables appearing in all the cubes). For example, given $n_i = x_1 x_2 x_3 + x_1 x_2 x_4 + x_1 x_5 x_6$, we split on x_2 to get $L = x_1 x_2 x_3 + x_1 x_2 x_4$, $R = x_1 x_5 x_6$. We do not split on x_1 in the above example, because x_1 appears in all the cubes. This split method is based on the following: the likelihood of a function being threshold is greater with fewer variables[10].

3.3 ILP problem and threshold function

One key issue of the above methodology is to determine whether a function is threshold or not. This problem is solved by casting it in an ILP (integer linear programming) formulation. The method of Ref. [10] is used to formulate the ILP problem. After formulation, an existing linear programming tool called LP _ SOLVE[10] is used to solve the problem.

3.4 Experiment results

We ran the benchmarks in the MCNC benchmark suite with the proposed threshold synthesis methodology. Some of the results are listed in Table 2. Conventional synthesis results are obtained by: firstly replacing each gate in the optimized Boolean network with an ERCCL gate, then adding the latches to accommodate gate level pipelining. The optimized Boolean network is obtained by running the script. boolean script in SIS[11] (an existing boolean logic synthesis tool).

It can be seen from the results that up to 85 % reduction in gate count is possible, with average reduction being 78 %.

Table 2 Threshold synthesis results compared with conventional synthesis

Benchmark	Conventional synthesis		Threshold synthesis	
	Gate	Level	Gate	Level
cm85a	165	8	37	2
cmb	100	7	36	3
term1	1790	12	378	4
pm1	117	5	17	2
x1	1053	10	298	4
cm152a	49	4	9	2

4 Conclusion

A novel energy recovery logic style ERCCL is presented in this paper. Compared to the conventional CMOS logic and other competing energy recovery logics, ERCCL has energy performance. ERCCL can energy-efficiently implement a complex logic with high fan-in in a single gate and a threshold logic. A novel energy recovery system design approach based on ERCCL, which minimizes the total non-adiabatic loss by minimizing the number of logic gates, is proposed. A threshold logic synthesis methodology for ERCCL, which reduces gate count significantly compared to the synthesis results of SIS, is also presented.

References

[1] Denker J S. A review of adiabatic computing. Proceedings of the Symp in Low Power Electronics, San Diego, 1994:94

[2] Liu F, Lau K T. Pass-transistor adiabatic logic with NMOS pull-down configuration. Electron Lett, 1998, 34(8):739

[3] Kim C, Yoo S M, Kang S M. Low-power adiabatic computing with nMOS energy recovery logic. Electron Lett, 2000, 36(16):1349

[4] Moon Y, Jeong D K. An efficient charge recovery logic circuit. IEEE J Solid-State Circuits, 1996:514

[5] Suvakovic D. Application of energy recovery in low-power DSP system design. PhD Thesis, 2002

[6] Yang Q, Zhou R D. ERCCL: energy recovery capacitance coupling logic. ICSICT', 2004:2090

[7] Kohavi Z. Switching and finite automata theory. McGraw-Hill, 1978

[8] Yang Q, Zhou R D. Energy recovery threshold logic and power clock generation circuits. Chinese Journal of Semiconductors, 2004, 25(11):1403

[9] Brayton R K, Hachtel G D, Sangiovanni-vincentelli A L. Multilevel logic synthesis. Proc IEEE, 1990, 78(10):264

[10] Zhang R, Gupta P, Zhong L, Jha N K. Synthesis and optimization of threshold logic networks with application to nanotechnologies. Proc Design, Automation and Test in Europe Conference and Exhibition, 2004:904

[11] Sentovich E M, Singh K J, Moon C, et al. Sequential circuit design using synthesis and optimization. Proc Int Conf Computer Design, 1992:328

能量回收电容耦合逻辑及其综合方法*

杨　骞　周润德

(清华大学微电子学研究所, 北京　100084)

摘要: 提出了一种新型能量回收电路 ERCCL (能量回收电容耦合逻辑), 该电路的能耗低于传统 CMOS 电路及其他能量回收电路. ERCCL 利用电容耦合进行逻辑求值, 因此可以在一个门中低能耗地实现高扇入、高复杂度的逻辑. 同时 ERCCL 是一种阈值逻辑. 所以一个基于 ERCCL 的系统可以大大减少逻辑门数, 从而降低系统能耗. 针对 ERCCL 提出了一种阈值逻辑综合方法. 用基准电路集 MCNC 做了相应的实验. 与 SIS 的综合结果相比, 该方法大约减少 80 %的逻辑门.

关键词: 能量回收; 阈值逻辑; 逻辑综合; 电容耦合; CMOS 电路

EEACC: 1265A; 1130; 2570D

中图分类号: TN432　　**文献标识码**: A　　**文章编号**: 0253-4177(2005)07-1334-06

*国家自然科学基金资助项目(批准号:59995550-1)

杨　骞　男, 1977 年出生, 博士研究生, 从事 CMOS 低功耗集成电路研究.

周润德　男, 1945 年出生, 教授, 博士生导师, 从事低功耗集成电路与嵌入式系统研究.

2004-07-23 收到, 2005-02-17 定稿