부울함수의 XOR 표현을 기초로 한 저전력 논리합성

황 민, 이귀상
전남대학교 전산통계학과
전화 : 062-530-0147 / 핸드폰 : 019-613-1142

Low Power Logic Synthesis based on XOR Representation of Boolean Functions

Min Hwang, Guee-Sang Lee
Dept. of Computer Science, Chonnam National University
E-mail : hwang@cs.chonnam.ac.kr

Abstract

In this paper, we put forth a procedure that target low power logic synthesis based on XOR representation of Boolean functions, and the results of synthesis procedure are a multi-level XOR form with minimum switching activity. Specially, this paper show a method to extract the common cubes or kernels by Boolean matrix and rectangle covering, and to estimate the power consumption in terms of the extracted common sub-functions.

I. 서론

정보화에 대한 인식과 활용이 보편화되면서 발생되는 휴대용 컴퓨터를 비롯하여 이동 통신이나 개인 휴대통신의 폭발적인 수요를 고려할 때, 각종 시스템에 있는 회로에서 전력소모를 고려한 집적회로(IC) 설계 방법은 회로 설계시 동작속도를 줄인다거나 회로의 면적을 최소화하는 것만큼이나 필수적인 고려사항으로 인식되고 있다. 부가적으로, 높은 전력소모(power dissipation)는 장치의 열을 분산시키는데 따른 비용을 증가시킬 뿐만 아니라 더욱 높은 트랜지스터 카운트와 빠른 클럭 속도를 가진 오늘날의 복잡한 회로의 신뢰도를 떨어뜨릴 수 있으므로 저전력(low power) 논리합성(logic synthesis)에 대한 연구는 필수적이다.

지금까지의 저전력 논리합성에 관한 대부분의 연구는 AND/OR 혹은 NAND/NOR를 기반으로 하는 논리합성에 초점이 맞추어져 있었다. 전형적인 접근방법은 우선 초기에 다단계(multi-level) AND/OR나 NAND/NOR 표현의 부울함수(Boolean function)를 생성한 후 이 표현을 여러 가지 잘 알려진 기법들을 통한 스위칭 동작에 대해 최적화 시킨다[3]. 가령, 과거의 연구는 AND/OR 게이트를 기반으로 한 저전력 기술분할(technology decomposition)이나 기술매핑(technology mapping)에 초점이 맞추어져 있었다[1,4]. 그렇지만, 산술함수(arithmetic functions) 즉, 덧셈기(adders), 곱셈기(multipliers), 그리고 에러수정코드(error correcting codes) 등과 같은 함수들은 AND/OR 표현보다는 AND/XOR 표현으로 더욱 간결한 함수를 얻을 수 있다[5,6,7]. 최근에 잘 알려진 고정극성 리드뮬러(FPRM: Fixed Polarity Reed-Muller) 형식에서 초기 AND/XOR 표현을 목적으로 한 최적화 기법들을 사용하여 회로의 면적과 전력 감소를 이루는 사례도 있었다[7,8].

본 논문에서는 XOR 표현의 부울함수를 기반으로 하는 저전력 논리합성을 위한 방법으로, 임의의 ESOP (EXOR Sum-of-Products) 표현에서 factoring에 의한 최소의 전력을 소모하는 다단계 리드뮬러 형식을 합성하는 방법을 제시하고, 이를 구현해 보고자 한다. 특히, 본 논문에서는 부울행렬(Boolean matrix)과 사각형커버링(rectangle covering)에 의하여 논리함수의 커널(kernels)을 추출하고, 이를 전력소모 예측함수를 기반으로 효과적인 factoring을 유도함으로써 저전력 논리합성을 이루도록 하였다. 본 논문의 접근방법은 zero delay model을 가정하였다. 구성은, 2장에서 회로의 스위칭 동작에 관하여 기술하고, 3장과 4장에서는 사각형커버링에 의한 AND/XOR 회로에서의 공통큐브(cube)를 추출하는 방법과 전력소모 예측함수를 이용한 factoring 방법을 각각 논의한다. 5장에서는 결론을 제시한다.

II. 관련연구

일반적으로 VLSI 회로들은 CMOS로 구현되고 있으

며, 여기에서의 주된 전력소모는 다음과 같다[1,2,3].

(1) 이상적인 MOS 트랜지스터가 아닌 데에 따른 전력 누수(leakage current)에 의한 정적(static) 전력소모
(2) 출력전이시에 발생하는 단락회로(short circuit)에 흐르는 전류에 의한 전력소모
(3) 회로 노드의 charge/discharge에 의한 전력소모

이 중 (2)-(3)은 CMOS 게이트의 출력이 0→1 또는 1→0으로 전이(transition)가 일어날 때 발생하는 출력전이율(number output transitions per clock cycle)로 결정되며, 이를 동적인 전력소모(dynamic power consumption)라고도 한다. 대부분의 저전력 논리합성 방법들은 회로에서 전력소모를 줄이기 위하여 회로의 총 전력소모의 90%를 점유하고 있는 (3)에 의한 전력소모를 최소화하는데 중점을 두고 있다[1]. CMOS 회로기술에서 칩에서의 동적인 전력소모로 인해 발생되는 전력소모량은 다음과 같이 계산된다[1,3].

$$\sum_{i=1}^{N} \frac{1}{2} \cdot V_{dd}^2 \cdot C_i \cdot f \cdot E_i \qquad (2.1)$$

식 (2.1)에서 V_{dd}는 공급전압(supply voltage)이며, C_i는 i번째 게이트 출력의 로드(load) 커패시턴스(capacitance)이고, f는 주파수변이, E_i는 클럭사이클(clock cycle) 동안 게이트 g에서 발생하는 신호변이 또는 스위칭 동작의 기대치(the expected number of signal transition or switching activity), 그리고 N은 칩에 있는 전체 게이트 수이다. 특히, E_i와 f의 곱(product)은 초당 발생하는 전이의 수로, 전이밀도(transition density)로 참조되며, 이러한 E_i와 f에서 전력감소는 회로의 총 전력소모 감소를 유도해낼 수 있다. 또한, E_i와 C_i의 곱은 스위칭 이후의 게이트 출력의 로드 캐패시턴스로 참조된다[1,3].

회로에서 각 노드의 스위칭 동작을 계산하는 방법으로는 시뮬레이션을 방법(simulation based techniques), 확률적인 방법(probabilistic techniques), OBDDs(Ordered Binary Decision Diagrams)을 구성하여 계산하는 방법 등 다양하며[1,4], 이들 중 본 논문에서는 확률적인 접근방법을 이용한다.

일반적으로, 논리합성 도구에서는 입력 화일에 대하여 여러 개의 노드들이 연결된 네트워크가 구성되며, 이러한 각 노드에서의 전력소모 예측함수의 설정이 선행되어져야 한다. 즉, 임의의 함수 f의 전력소모를 정확히 예측하기 위해서는 함수 f를 구성하는 게이트 출력에서의 스위칭 동작과 로드 캐패시턴스를 계산하여야 하는데, 이 때 노드의 스위칭 동작의 계산 방법으로는, 네트워크에서 임의의 내부노드 n의 출력이 high(on)가 되는 신호확률(signal probability)을 $p(n)$이라 할 때, 이 노드 n의 스위칭 동작(switching activity) $E(n)$은 식(2.2)와 같다.

$$E(n) = 2 * p(n) * (1 - p(n)) \qquad (2.2)$$

여기서, 회로의 네트워크가 간단한 게이트들로만 구성되고, re-convergent fanout 노드들을 갖지 않는다고 가정하면, 네트워크에서 각 게이트의 신호확률은 NOT gate: $p(o) = 1 - p(i)$, AND gate: $p(o) = \prod_{i \in inputs} p(i)$, OR gate: $p(o) = 1 - \prod_{i \in inputs} (1 - p(i))$, 그리고 XOR 게이트를 포함한 다른 게이트들은 이 세 가지를 이용하여 계산된다[1].

III. AND/XOR 표현의 공통큐브추출

본 장에서는 면적(area)과 전력(power)의 최소화를 위한 다단계 리드뮬러 형식의 표현을 얻기 위하여 임의의 이단계 AND/XOR 표현에서 공통부분표현(common sub-expression)을 추출하고, factoring 방법을 적용하여 함수의 크기를 줄이면서 동시에 전력소모를 줄이는 방법을 기술한다. 이를 위해 먼저 논리함수에서 공통부분표현을 추출하기 위한 방법들을 소개한다.

3.1 kernel과 kernel의 추출

논리함수를 다루는데 중요한 논리연산은 어떤 주어진 표현에서 공통부분표현을 찾는 일이다. 이 공통표현은 대개 divisor로 표현되며, 다음과 같이 설명할 수 있다. 즉, 임의의 함수 f와 논리표현 p, q, r에 대하여 $f = pq + r$ 이나 $f = pq \oplus r$과 같이 표현할 수 있다면, 이를 "f를 p로 나누었다."고 하며, q를 몫, r을 나머지라 한다. 그리고 적절한 divisor를 구하여 공통부분을 추출함으로써 논리식이 간소화[2]된다. 이러한 divisor 중의 하나가 "커널(kernel)"이며, 이는 두 개 이상의 논리표현에서 공통표현을 찾는데 효과적인 수단으로 사용된다. 여기서 간단히 커널을 정의하자.

임의의 함수 f를 어떤 큐브로도 균등하게 그 표현을 나눌 수 없으면 "cube-free"라 한다. 예를 들어, $f = ab + c$는 cube-free이지만 $f = ab + ac$나 $f = abc$는 cube-free가 아니다. 또한 임의의 논리함수 f의 "primary divisor" $D(f)$는 표현(함수)들의 집합으로 다음과 같이 정의된다.

$$D(f) = \{f/c \mid c \text{ is a cube}\}$$

그리고 함수 f의 kernel $K(f)$는 f의 cube-free인 primary divisor들의 집합으로, 다음과 같은 표현들의 집합이다.

$$K(f) = \{g \mid g \in D(f) \text{ and } g \text{ is cube} - \text{free}\}$$

kernel $k = f/c$를 구하기 위해 사용된 큐브(cube) c는 k의 "co-kernel"이라 한다. 예를 들어 다음과 같은 논리함수에서 이에 대응하는 추출된 kernel들은 다음 표에 나타나 있다. 여기서 kernel 추출과정은 생략한다.

$$\begin{aligned} f &= x_1x_4x_6 \oplus x_1x_5x_6 \oplus x_2x_4x_6 \oplus x_2x_5x_6 \\ &= \oplus x_3x_4x_6 \oplus x_3x_5x_6 \oplus x_7 \\ &= (x_1 \oplus x_2 \oplus x_3)(x_4 \oplus x_5)x_6 \oplus x_7 \end{aligned}$$

Kernel	Co-kernel	level
$(x_1 \oplus x_2 \oplus x_3)$	$x_4x_6,\ x_5x_6$	0
$(x_4 \oplus x_5)$	x_1x_6, x_2x_6, x_3x_6	0
$(x_1 \oplus x_2 \oplus x_3)(x_4 \oplus x_5)$	x_6	1
$(x_1 \oplus x_2 \oplus x_3)(x_4 \oplus x_5)x_6$	1	2

Figure 2.1 Kernels and Co-Kernels of function f

2.2 rectangle과 rectangle covering

부울대수 연산에서 매우 중요한 문제는 divisor를 알아내는 것으로, kernel은 factoring, 분할(decomposition), 그리고 공통부분추출(extraction)에 필요한 divisor들의 집합이므로, kernel을 찾는 문제는 하나 또는 다수의 공통큐브 divisor를 구하는 것과 함께 공통표현을 찾아 이를 추출해 내는 수단이다. 그런데 공통표현을 찾는 일은 수학적인 문제로 처리될 수 있으며[2], 여기서는 사각형(rectangle)과 사각형 커버링(rectangle covering) 문제에 대해서 알아본다. 이 후, 이러한 rectangle이 어떻게 공통표현을 찾는데 이용되는지 설명하고, 다음절에서는 이 방법이 저전력 논리합성에 어떻게 이용될 수 있는지 알아보기로 한다.

Definition 3.1: 행렬 $B, B_{ij}=\{0,1,*\}^2$의 rectangle (R, C)은 모든 $i \in R, j \in C$에 대하여 $B_{ij} \in \{1,*\}$인 행들의 부분집합 R과 열들의 부분집합 C로 이루어진다.

$R_2 \subseteq R_1$이고 $C_2 \subset C_1$이거나 $R_2 \subset R_1$이고 $C_2 \subseteq C_1$이면 rectangle (R_1, C_1)은 rectangle (R_2, C_2)을 엄격히 포함한다고 한다. 행렬 B의 "prime rectangle" (R, C)은 다른 어떤 B의 rectangle에도 엄격히 포함되지 않는 rectangle이다. rectangle (R, C)의 "co-rectangle"은 (R, C')을 말하며, 이 때 C'는 C에 존재하지 않은 열들을 말한다. rectangle의 집합 $\{(R^k, C^k)\}$은, 만일 $B_{ij}=1$이 $i \in R^k, j \in C^k$을 의미한다면 행렬 B의 "rectangle cover"라 한다. 그러면 B의 각 1은 적어도 하나 이상의 rectangle에 의하여 포함되어야 한다. Covering은 반드시 disjoint 할 필요는 없으며, 따라서 B의 각 1은 하나 이상의 rectangle에 의하여 커버될 수 있다. B의 항 중 *는 don't care이다.

일반적으로 임의의 rectangle (R^k, C^k)가 무게함수 $w(R^k, C^k)$에 의해 정의된 weight(or cost) 함수를 갖고 있다면 rectangle cover $\{(R^k, C^k)\}$의 무게(weight)는 다음과 같이 나타낼 수 있다.

$$\sum_k w(R^k, C^k) \tag{3.1}$$

그리고 최소의 weighted-rectangle covering 문제는 주어진 행렬 B에서 최소의 총 weight 합을 갖는 rectangle cover를 찾는 것이다.

이러한 rectangle을 사용하여 논리함수의 kernel을 계산해 낼 수 있다. 가령, 임의의 논리함수 $f = x_1x_2x_5 \oplus x_1x_3x_4 \oplus x_2x_3x_4$을 부울행렬(cube-literal 행렬)로 표시하면 각 행에는 각각의 큐브를, 각 열에는 각각의 리터럴들을 배치하여 다음과 같이 표시할 수 있다.

cubes \ literals	x_1	x_2	x_3	x_4	x_5
$x_1x_2x_5$	1	1	0	0	1
$x_1x_3x_4$	1	0	1	1	0
$x_2x_3x_4$	0	1	1	1	0

Figure 3.1 Cube-literal Matrix

위 행렬에서 rectangle $\{R, C\}=\{(2,3),(3,4)\}$에 대한 co-rectangle $\{R, C'\}=\{(2,3),(1,2,5)\}$이고, 이의 대응 표현은 $x_1 \oplus x_2$이다. 즉, 이 예에서 rectangle $\{R, C\}=\{(2,3),(3,4)\}$은 큐브 x_3x_4에 해당하며, 따라서 rectangle은 co-kernel에 해당하며 kernel을 얻기 위해 필요한 큐브 divisor이다. 즉, 큐브 divisor는 열 C에 있는 리터럴들의 집합 x_3x_4을 말한다.

Ⅳ. 저전력을 위한 kernel 추출

논리함수의 간소화와 전력소모를 줄이는 과정에서, rectangle covering을 적용함에 있어 네트워크의 모든 공통큐브들 중에서 어떤 큐브를 공통부분표현으로 선택할 것인가 하는 문제에 대하여 알아보자. 이를 위해서는 임의의 함수 f를 구현하는 게이트들의 출력에서 스위칭 동작과 출력로드 캐패시턴스를 계산하여 회로의 전력소모를 예측하여야 하며, 네트워크에서 임의의 내부노드 n의 출력이 high가 되는 신호확률을 $p(n)$이라 할 때, 이 노드 n의 스위칭 동작은 식 (2.2)로 계산되고, 이단계 AND/XOR 표현의 출력로드의 캐패시턴스를 위해서는 다음의 정의를 사용한다[1,9].

Definition 4.1: 임의의 노드 n_i와 fanin 노드 n_j가 주어질 때, 노드 n_i에 대한 n_j의 cube load $CL(n_j, n_i)$은 변수 n_j가 노드 n_i의 AND/XOR 표현에서 사용된 횟수로 정의된다.

위의 정의를 이용하여 임의의 노드의 전력소모함수, 즉 그 노드가 전체 네트워크의 전력소모에서 차지하는 부분을 다음과 같이 정의한다. 즉, AND/XOR 형태로 구현된 노드에 의해 소비된 전력소모 계산으로, 출력 함수 f와 cubes $(c_1, \ldots, c_N)$로 구성된 노드 n_i가 주어질 때, 그 노드 n_i의 전력소모 비용함수[1]는 다음 식(4.1)과 같다.

$$P_{n_i} = E(f) \cdot \sum_{n_k \in fanouts(n_i)} CL(n_i, n_k) + \sum_{c_m \in cubes(n_i)} E(c_m) + \sum_{n_j \in fanins(n_i)} E(n_j) * CL(n_j, n_k) \tag{4.1}$$

이 식의 첫 번째 항은 노드 출력에서 전력소모를, 둘째 항은 함수의 각 곱항을 구현하는 게이트들의 출력에서의 전력소모를, 셋째 항은 노드의 입력에서의 전력소모를 의미한다.

그리고 $F=(f_1,\cdots,f_L)$는 다중출력(multi-output) 부울함수라 하자. 여기서, F의 각 함수들은 cubes $(c_1,\cdots,c_N)$과 입력집합 $(v_1,\cdots,v_M)$으로 구성된 함수이다. 그리고 $D=d_1\oplus\cdots\oplus d_P$는 함수 F의 kernel들을 나타내고, $Q=\{q_1,\cdots,q_R\}$는 함수 F에서 kernel D의 추출을 위한 co-kernel들의 집합을 나타낸다고 할 때, kernel D를 추출할 때의 줄어드는 리터럴의 면적값(area value)은 다음과 같은 식으로 생각할 수 있다.

$$\left((R-1)\cdot\sum_{i=1}^{P}lit(d_i)\right)+\left((P-1)\cdot\sum_{i=1}^{R}lit(q_i)\right)-R \quad (4.2)$$

여기서 첫째 항은 kernel을 반복하지 않아 절약된 리터럴들의 수, 두 번째 항은 co-kernel을 반복하지 않음으로써 절약된 리터럴들의 수, 그리고 세 번째 항은 kernel D를 추출함으로써 유도된 리터럴들의 수이다. 또한, kernel D를 추출하기 위한 전력값(power value)은 다음 식으로 계산된다.

$$\begin{aligned}&\left((R-1)\cdot\sum_{i=1}^{M}E(v_i)\cdot CL(v_i,D)\right)+\\&\left((P-1)\cdot\sum_{i=1}^{P}\sum_{j=1}^{M}E(v_j)\cdot CL(v_j,q_i)\right)+\\&\left(\sum_{i=1}^{P}\sum_{j=1}^{R}E(d_i\,q_j)\right)-(R\cdot E(D))-\\&\left(\sum_{i=1}^{P}E(d_i)\right)-\left(\sum_{j=1}^{R}E(Dq_j)\right)\end{aligned} \quad (4.3)$$

위 식의 첫 번째 항은 kernel의 각 입력에서 load의 감소를 나타내고, 두 번째 항은 co-kernel의 각 입력에서 load의 감소를, 세 번째 항은 함수 F의 원래의 AND/XOR 표현에서 제거된 함수의 cube들의 수, 네 번째 항과 다섯 번째 항은 네트워크에 삽입된 새로운 노드의 출력과 큐브들의 출력에서의 각 전력소모, 그리고 여섯 째 항은 AND/XOR 표현에 삽입된 새로운 큐브들의 출력에서의 전력소모를 각각 나타낸다. area value와 power value의 계산을 위하여 다음 예를 보자.

Example 4.1: 다음과 같이 임의의 논리함수를 가정하자.

$$f = x_1x_2x_3\oplus x_2x_3x_4\oplus x_4x_5x_6$$

그리고 이 함수의 주입력(primary inputs)에서의 신호확률들을 $p(x_1)=0.92, p(x_2)=0.47, p(x_3)=0.75, p(x_4)=0.05, p(x_5)=0.85, p(x_6)=0.37$로 가정하자.

그러면 위 함수 f는 공통부분표현 kernel과 co-kernel을 포함한 다음과 같은 두 개의 부함수(sub-function)로 표현할 수 있다. 즉, 만일

f로부터 kernel D_1의 추출 :

$f_1=D_1x_2x_3\oplus x_4x_5x_6,\ D_1=x_1\oplus x_4 \Rightarrow$ co-kernel $q_1=x_2x_3$, 그리고 $R_1=1,\ P_1=2$

f로부터 kernel D_2의 추출 :

$f_2=D_2x_4\oplus x_1x_2x_3,\ D_2=x_2x_3\oplus x_5x_6\Rightarrow$ co-kernel $q_2=x_4$, 그리고 $R_2=1,\ P_2=2$

그러므로 논리함수에서 저전력을 위한 공통표현을 추출하는데 있어서 f로부터 kernel D_1과 D_2 중 최적의 kernel을 추출하기 위해서는, 주입력의 신호확률과 식(4.3)와 식(4.4)을 기반으로 함수의 스위칭 동작을 계산하면, kernel D_1 (D_2)를 각각 추출할 때, 함수 f_1 (f_2)의 area value는 1(0)으로. 이는 kernel 추출 후에 함수 f_1 (f_2)의 리터럴이 1(0)개만큼 감소되었음을 의미한다. 또한 이 때의 power value는 각각 0.462와 -1.2648이다. 이 결과는 kernel D_1이 추출된 이후의 회로의 전력소모는 D_2가 추출된 이후의 회로의 전력소모보다 감소한다는 것을 의미한다. 또한 kernel D_2가 추출된다면 회로에서의 전력소모는 오히려 증가함을 의미한다.

V. 결론

본 논문에서는 XOR 표현을 기반으로 한 논리회로에서 rectangle covering에 의한 kernel과 공통큐브를 추출하고, 이에 따른 소모전력을 예측하기 위하여 전력소모 예측함수를 적용하여 power value가 최대 값을 갖는 부분표현을 선택하는 방법으로 회로에서 소모되는 전력을 최소화하였다. 본 논문에서 제시된 방법은 최소의 전력을 소모하도록 XOR-트리, AND-트리, 그리고 OR-트리로 구성된 전력 최적화 다단계 표현의 회로를 합성할 수도 있을 것이다.

참고문헌

[1] S. Iman and M. Pedram. Logic Synthesis for Low Power VLSI Designs. Kluwer Academic Publishers, 1998.

[2] R. K. Brayton, G. D. Hachtel, and A. L. Sangiovanni-Vincentelli, "Multilevel Logic Synthesis," In Proceedings of IEE, vol.78, no.2, pp.264-300, 1990.

[3] U. Narayanan and C. L. Liu. "Low Power Logic Synthesis for XOR based circuits," In International Conference on Computer-Aided Design. 1997.

[4] C. Y. Tsui, M. Pedram, and A. M. Despain. "Technology decomposition and mapping targeting low power dissipation." In Design Automation Conference, pp. 68-73, June 1993.

[5] Tsutomu Sasao, editor. Logic Synthesis and Optimization. Kluwer Academic Publishers, 1993.

[6] Tsutomu Sasao and Masahiro Fujita, editor. Representations of Discrete Functions. Kluwer Academic Publishers, 1996.

[7] C. C. Tsai and M. Marek-Sadowska. "Multilevel Logic Synthesis for Arithmetic Functions," In Design Automation Conference, pp. 242-247, 1996.

[8] R. Drechsler, M. Theobald, and B. Becker. "Fast OFDD based Minimization of Fixed Polarity Reed-Muller Expressions." In IEEE European Design Automation Conference, pp.2-7, Sep. 1994.

[9] S. Iman and M. Pedram. "Logic Extraction and Factorization for Low Power," In Design Automation Conference, pp.248-253, 1995.