Low Power Driven Pass Transistor Logic Synthesis by Binary Decision Diagrams

Student : Yi-Yu Liu
Advisor : TingTing Hwang

Department of Computer Science
National Tsing Hua University
HsinChu, Taiwan 30043

Contents

List of Figures

List of Tables

Abstract

In this paper, we present methods to synthesize low power Pass Transistor Logic (PTL) cell. Given that the power consumption of a PTL cell includes power consumed at the gate terminal and source/drain capacitance when current flows from power/ground to output, we will translate finding a low power PTL cell to finding an OBDD with some defined cost function. This cost function includes the minimization of occurrence of variables with high transition probability and the minimization of the expected path length of an OBDD. To compute the cost function efficiently, three methods will be proposed to calculate the expected path length of Ordered Binary Decision Diagrams (OBDD).

Chapter 1

Introduction

Pass Transistor Logic (PTL) offers an alternative to static CMOS design styles. Unlike static CMOS where all paths from V_{dd} to output are connected via p-MOS (a pull-up network) and all paths from GND to output are connected via n-MOS (a pull-down network), Pass Transistor Logic is not strictly restricted. As a result, most functions can be implemented by Pass Transistor Logic with less transistors than by static CMOS. With less transistors, PTL designs as compared to traditional CMOS design offers not only higher performance but also less die area and power dissipation.

To synthesize PTL networks, many researches [1] [2] [3] [4] have proposed direct mapping from Reduced Ordered Binary Decision Diagrams (ROBDD) to PTL circuits because the mapped circuits are always sneak-path-free, i.e., at one time, only one path connecting the ground/power to the output is active. Generally, these ROBDD-based approaches can be classified into two types: monolithic ROBDD-based [2] [3] and decomposed ROBDD-based [1] [4]. In monolithic ROBDD-based approach, an ROBDD for a function is constructed, its size is minimized and then a one-to-one direct mapping from ROBDD to PTL circuit is

performed. In decomposed ROBDD-based approach, a multi-level network is decomposed from bottom up. At each decomposition point, one or more ROBDD can be constructed, and then PTL circuits are synthesized from the ROBDD rooted at the decomposition point. In either approach, the resultant PTL circuit may have a long chain of transistors in series that will cause signal and speed degradation. To this problem, the solution is to properly insert buffers on the long path chain.

Low power has become key design issues in modern VLSI circuit design. Minimization of power dissipation can be conducted at algorithmic, architectural, logic and circuit levels [5]. Studies on low power design are abundant in the literature where various techniques have been proposed to synthesize static CMOS circuits. However, very few researches [1] consider power dissipation when synthesizing PTL circuit.

In an active path of MOS circuit, one of the factor of power dissipation is determined by parasitic capacitance. The parasitic capacitances consists of load capacitance and drain/source capacitance (diffusion capacitance). Load capacitance is the summation of metal capacitance and gate capacitance while diffusion capacitance is determined by the number of transistors on a path. Therefore, to synthesize a low power PTL cell, the number of transistors and the number of transistors on a path have to be taken into consideration.

As indicated that an ROBDD has a direct one-to-one mapping to PTL circuit, we intend to study in this paper the construction of an ROBDD that can result in PTL cells with low power consumption. Our proposition is that given signal probabilities of input variables, the occurrence of variable with high transition density and the path lengths with high probability

in an ROBDD can be reduced so that the load capacitance and drain/source capacitance of frequently on-paths are minimized and thus power consumption is reduced.

This thesis is organized as follows besides Chapter 1 as introduction. We will introduce some literature survey on BDD and PTL in Chapter 2. In Chapter 3, we will discuss the relations between a low power PTL cell and a minimum expected path length ROBDD. Then, we will propose three methods to compute the expected path length of an ROBDD and a heuristics will be proposed to find variable orderings so that the resultant ROBDD corresponds to a PLT cell with lower power consumption. Finally, experiment results will be drawn.

Chapter 2

Literature Survey

2.1 Binary Decision Diagram

Binary Decision Diagram (BDD) is a widely used representation in Boolean function. It is a directed, acyclic graph [6]. There are two types of nodes, terminal and non-terminal. The terminal node is a Boolean value of {0} or {1}, and the non-terminal node is a Boolean variable which is the *input variable* of the Boolean function. Each non-terminal node contains two outgoing edges, which point to other nodes, is the *parent-node*. For convenience, we distinguish the two as *then-edge* and *else-edge*. A node which is connected from a then-edge is the *then-child* of its parent-node and from an else-edge the *else-child*. The formula representation of an BDD is "if *parent-node* then *then-child* else *else-child*", which is called an *ITE* operator (IF-THEN-ELSE). By using ITE operator, Boolean operations can be easily performed on BDDs.

Reduced Ordered BDD (ROBDD) was introduced in [7]. It is a BDD with an ordered variable ordering in all paths from root node to terminal nodes and possesses the following properties: (1) if both the then-child and else-child of a non-terminal node are identical, the

non-terminal node will be removed in the path, and (2) if there are more than one subtree which are identical, only one subtree will be reserved and all parent-nodes which have the same subtree will share one subtree as their children. ROBDD possesses a fine structure in both internal binary operations and explicit representation. Figure 2.1 gives an ROBDD for function $F = A + B \cdot C'$ with input variable ordering $A < B < C$. Note that for the same function, different input ordering will result in different ROBDD.

Although there are many advantages in BDD, the size of BDD is growing rapidly as the *number of fanin*s increases. Besides, BDD size is also strongly affected by *fanin ordering*. To expand the limitation of BDD, several papers focus on the following topics: efficient memory management strategy [8], reducing computational overhead [9], choosing a good initial variable ordering [10], dynamic variable reordering [11].

2.2 Pass Transistor Logic

Pass Transistor Logic is a direct logical level implementation of transistors. An *AND* operation is implemented in serial while an *OR* operation is implemented in parallel. As a result, designers preserve a lot of logic-level features in transistor-level design. Take 4-to-1 MUX as an example. Both the static CMOS and PTL implementations for this example are drawn in Figure 2.2 and 2.3. It is clear from these Figures that the static CMOS version is much more complex than the PTL version.

Because of the switching characteristics of MOS transistors, it is quite simple to implement a MUX as a wired OR of transistors in PTL, a non-terminal node in an ROBDD

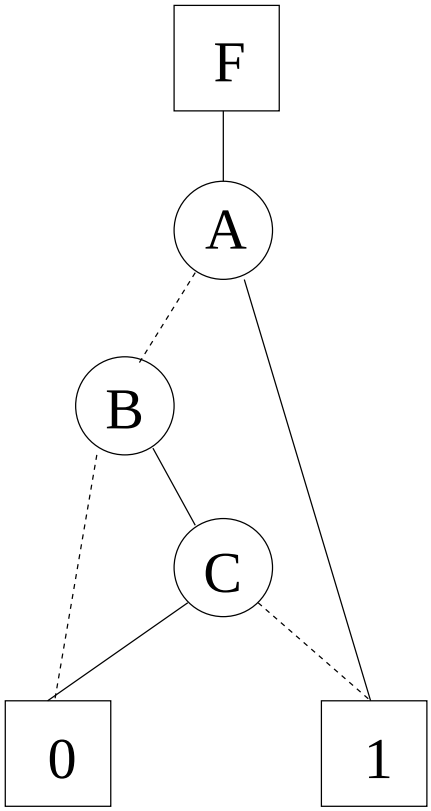

Figure 2.1: An ROBDD example.

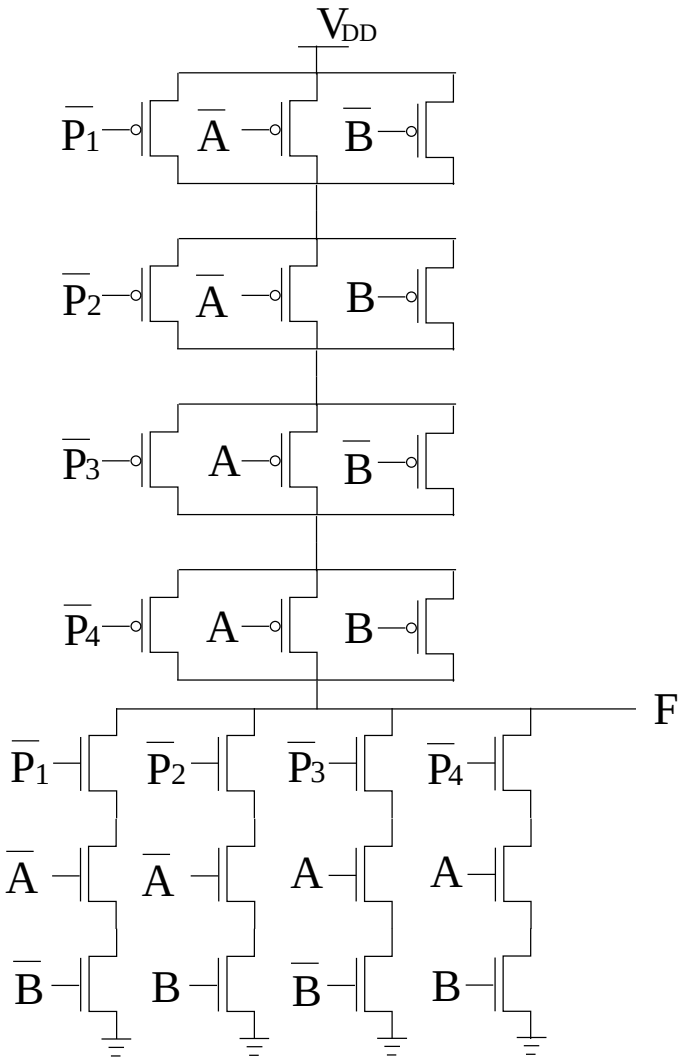

Figure 2.2: A CMOS version of 4-to-1 multiplexer.

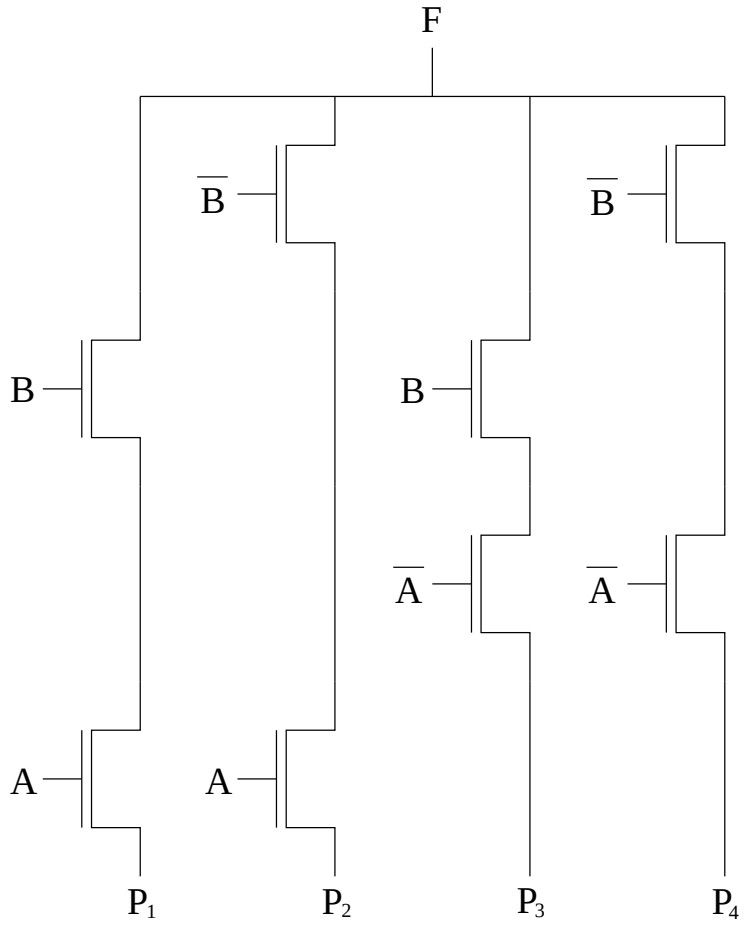

Figure 2.3: A NMOS version of 4-to-1 multiplexer.

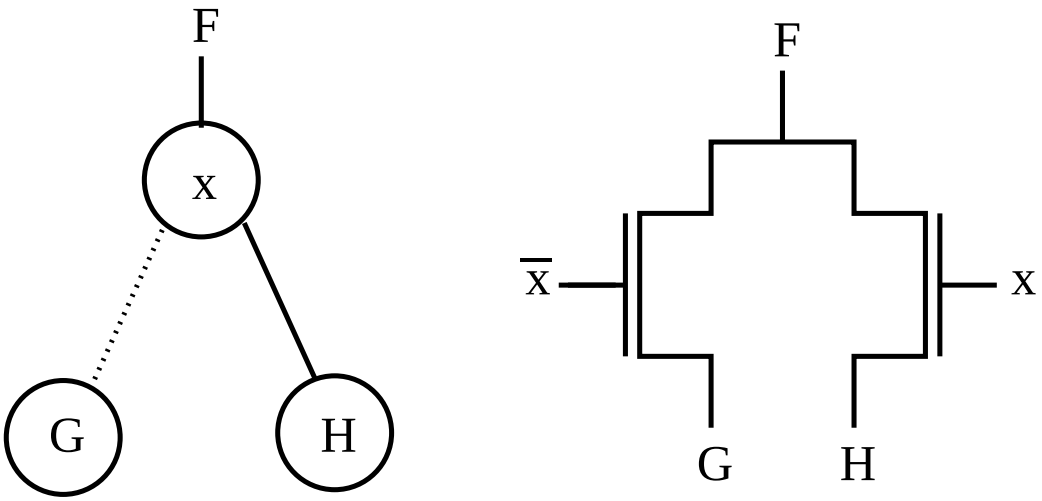

Figure 2.4: A BDD maps a PTL circuit.

can be directly mapped to PTL. For example, a node in an ROBDD representing Boolean function $F = ITE(x, G, H)$ can be easily implemented in a n-MOS PTL circuit as shown in Figure 2.4, where the input variable x corresponds to the controlling signal of the MUX. Moreover, the characteristic that all paths in an ROBDD are disjointed prevents the implementations of the mapped PTL from sneak path [1]. Figure 2.5 is a PTL circuit implemented by ROBDD in Figure 2.1.

There are several ways to implement a node of PTL circuits: transmission gate, n-MOS, and p-MOS. N-MOS implementation provides a faster propagation time and smaller die size than other implementations. Therefore, most of previous researches take n-MOS implementation. However, a n-MOS is good to conduct logic *low*, but poor to conduct logic *high*. Due to this characteristic, signals pass through long path length in PTL may cause distortion. To this problem, suitably dividing path length and inserting buffer are important design issues in PTL.

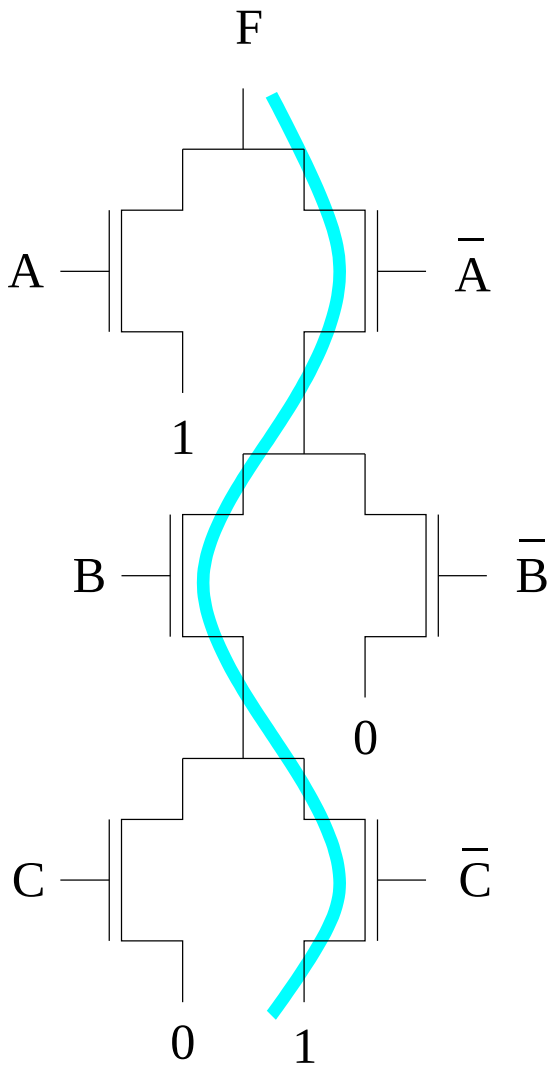

Figure 2.5: Implementing an ROBDD to a PTL.

Generally, ROBDD-based approaches can be classified into two types: *monolithic* ROBDD-based and *decomposed* ROBDD-based. In monolithic ROBDD-based approach [2] [3], iteratively division of ROBDD is proposed in [2]. ROBDD for the function is generated at the beginning. Then, the BDD is partitioned into smaller trees so that each of the tree height does not exceed the maximum tree height of PTL. These trees are delimited by buffers. After buffer insertion, these trees are mapped into PTL cells by technology mapper.

In decomposed ROBDD-based approach [1] [4], Boolean function is represented by its high level functional description (multi-level network) instead of its original ROBDD representation. It is decomposed from bottom up. At each decomposition point, one or more ROBDD can be constructed and then PTL circuits are synthesized from the ROBDD rooted at the decomposition point.

Chapter 3

Low Power PTL

3.1 Power Consumption for PTL

The power consumption model of a MOS circuit is determined by

$$W = \frac{1}{2} \cdot C \cdot V^2$$

where C is MOS capacitance and V is MOS operating voltage. The capacitance consists of two parts. One part is from the load capacitance which includes metal and gate capacitance. The other part is from the drain/source capacitance when current flows from transistors to output node. For the first part of capacitance, since a node in ROBDD corresponds to two transistors in PTL circuit and input variable at the node corresponds to the input signal at the gate terminal of the transistors, minimization of the power consumed due to load capacitance can be translated to minimization of the occurrences of input variable with high transition probability in an ROBDD.

For the second part of capacitance, since the number of drain/source capacitance in series on an active path corresponds to the path length of a path in an ROBDD, minimization

of the capacitance on an active path is shortening of the path length of a path with high probability to be active. For example, the path for $A = 0, B = 1$, and $C = 0$ on ROBDD in Figure 2.1 corresponds to the high-lighted path shown in Figure 2.5 where current flows through three transistors when the path is active. If the probability of $Prob(A = 0) \times Prob(B = 1) \times Prob(C = 0)$ is high, the probability of this long path being active is high and hence more power will be consumed. Therefore, minimization of the power when current flows from power/ground to output node can be translated to minimization of the expected path length of an ROBDD.

Based on the observations mentioned above, the power due to the input transition at the gate terminal is modeled as:

$$\sum_{i=1}^{n} T(X_i) \cdot Occ(X_i)$$

where $T(X_i)$ is the transition density of input variable X_i, $Occ(X_i)$ is the occurrences of input variable in an ROBDD, and n is the number of input variables. Moreover, the power due to current-flow from power/ground to output node is modeled as the expected path length of an ROBDD:

$$\sum_{i=1}^{m} P_i \cdot L_i$$

where P_i is the probability for the ith path to be active, L_i is the path length of the ith path in ROBDD, and m is the number of total paths in ROBDD. Therefore, power consumption of a PTL cell can be modeled in an ROBDD as follows.

$$\alpha \sum_{i=1}^{n} T(X_i) \cdot Occ(X_i) + (1 - \alpha) \cdot \sum_{i=1}^{m} P_i \cdot L_i$$

where α denotes the weight factor of the power consumption of the two sources. Hence, to synthesize a low power PTL cell, our goal is to find an ROBDD such that the objective function defined above is minimized.

3.2 Expected Path Length of ROBDD

Since the power consumption of PTL is affected by path length of ROBDD, we intend to shorten the path lengths with high probability, i.e., to minimize the expected path length of the ROBDD. It is necessary that we be able to compute the expected path length of a given ROBDD.

To begin with, we define the expected length of the ith path in ROBDD as $P_i \cdot L_i$, where P_i is the probability of the ith path and L_i is the path length of the ith path. The total Expected Path Length (EPL) of ROBDD rooted at node r is defined as

$$EPL(r) = \sum_{i=1}^{m} P_i \cdot L_i \tag{3.1}$$

where m is the number of total paths of ROBDD. Following this formula, to calculate the expected path length will need directly enumerate all paths. This approach is infeasible because the number of paths may grows exponentially. In the following subsections, we will propose three efficient methods to calculate expected length: bottom-up method, top-down method, and middle-way method.

3.2.1 Bottom-up Method

Bottom-up method is formulated by which the expected path length can be calculated from terminal nodes up to root node. Since at each non-terminal node, the expected path lengths of its two child-nodes are already computed, we can compute the expected path length of the non-terminal node based on the expected path lengths for its child-nodes. Before we proceed to develop the formula, we define the following notations as:

$BEPL(a)$ the expected path length of all paths from a to terminal nodes

BP_i^a the probability of the ith path from node a to a terminal node

BL_i^a the path length of the ith path from node a to a terminal node

Now, let a be a non-terminal node with its then-child a_1 and else-child a_0 as shown in Figure 3.1. The expected path lengths of all paths from terminal nodes up to a_1 and a_0 can supposely be computed as

$$BEPL(a_1) = \sum_{i=1}^{m_{a_1}} BP_i^{a_1} \cdot BL_i^{a_1} \tag{3.2}$$

$$BEPL(a_0) = \sum_{i=1}^{m_{a_0}} BP_i^{a_0} \cdot BL_i^{a_0} \tag{3.3}$$

where m_{a_1} and m_{a_0} are the total number of paths from terminal nodes up to node a_1 and a_0, respectively.

Let the probability from node a to a_1 be p_a. Then, the probability from node a to a_0 will be $1 - p_a$. The expected path length from terminal nodes up to nodes a is computed as

$$BEPL(a) = \sum_{i=1}^{m_{a_1}} p_a \cdot BP_i^{a_1} \cdot (BL_i^{a_1} + 1) + \sum_{i=1}^{m_{a_0}} (1 - p_a) \cdot BP_i^{a_0} \cdot (BL_i^{a_1} + 1)$$

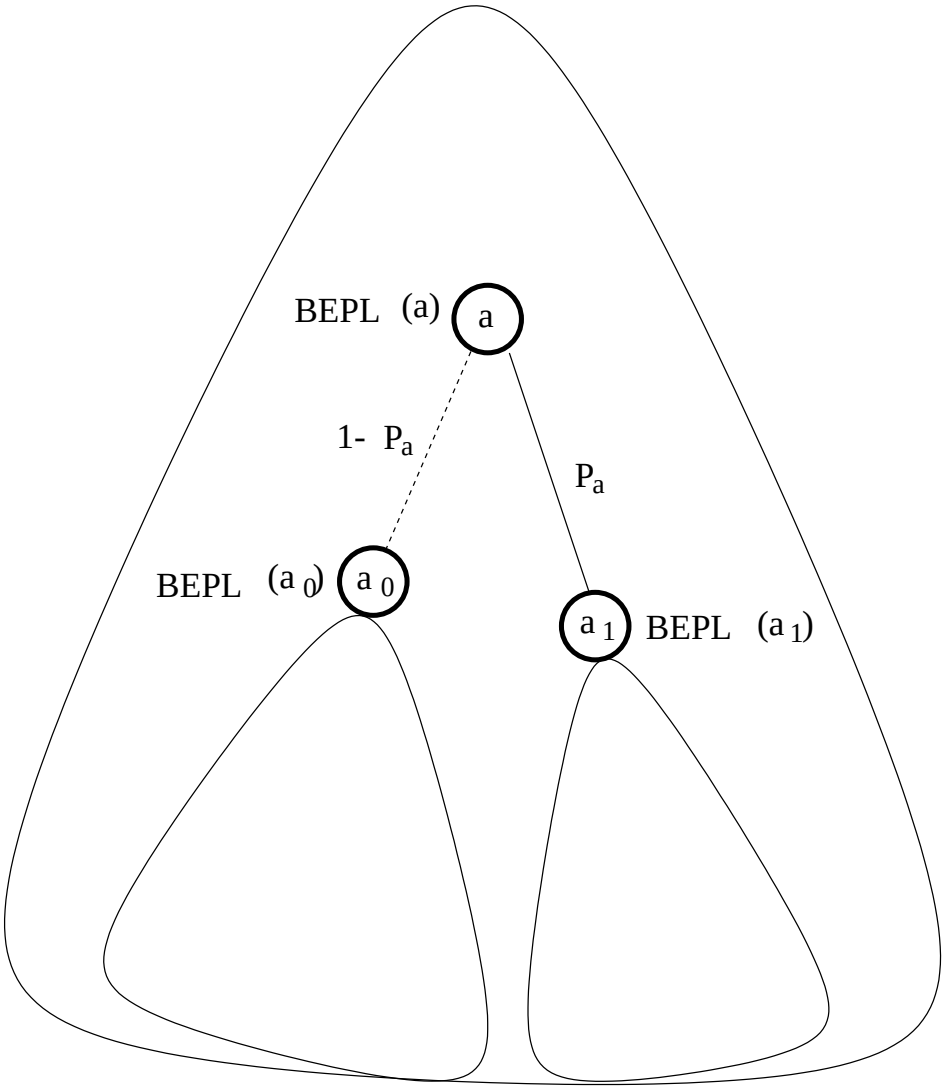

Figure 3.1: Bottom-up method for calculating the expected path length.

$$= \quad p_a \cdot (\sum_{i=1}^{m_{a_1}} BP_i^{a_1} \cdot BL_i^{a_1} + \sum_{i=1}^{m_{a_1}} BP_i^{a_1}) + (1 - p_a) \cdot (\sum_{i=1}^{m_{a_0}} BP_i^{a_0} \cdot BL_i^{a_0} + \sum_{i=1}^{m_{a_0}} BP_i^{a_0}) \tag{3.4}$$

Note that the total path probability from a non-terminal node to terminal nodes is 1. That is,

$$\sum_{i=1}^{m_{a_1}} BP_i^{a_1} = \sum_{i=1}^{m_{a_0}} BP_i^{a_0} = 1 \tag{3.5}$$

From the identity of equations (3.2) (3.3) (3.5), we can reduce equation (3.4) to

$$\begin{aligned} BEPL(a) &= p_a \cdot (BEPL(a_1) + 1) + (1 - p_a) \cdot (BEPL(a_0) + 1) \\ &= p_a \cdot BEPL(a_1) + (1 - p_a) \cdot BEPL(a_0) + 1 \end{aligned} \tag{3.6}$$

From equation (3.6), we follow that the expected path length of a non-terminal node at a can be computed by following steps :

step1 multiply the probability of the then-child by the expected path length of the then-child which is computed in the previous stage and is readily used.

step2 multiply the probability of the else-child by the expected path length of the else-child which is computed in the previous stage and is readily used.

step3 sum results of step1, step2, and constant 1

With boundary condition that both the path length of terminal nodes 0 and 1 are 0, we can recursively calculate the expected length of the root of an ROBDD by bottom-up method.

3.2.2 Top-down Method

Reversely, top-down method is used to calculate the expected path length from root node down to terminal nodes. Since at each non-terminal node, the expected path lengths of its parent nodes are already computed, we can compute the expected path length of an non-terminal node based on the expected path lengths for its parent nodes. Before we proceed to develop the formula, we define the following notations as:

$TEPL(b)$ the expected path length of all paths from root to non-terminal node b

TP_j^b the probability of the jth path from root to node b

STP^b the sum of probabilities of all paths from root to node b

TL_j^b the path length of the jth path from root to node b

Obviously, from the above definitions, equations (3.7) (3.8) hold. That is, the summation of probabilities of all paths from root to a node b_k is

$$STP^{b_k} = \sum_{j=1}^{m_{b_k}} TP_j^{b_k} \tag{3.7}$$

where m_{b_k} is the total number of paths from root to node b_k. Moreover, the expected path length from root to a node b_k is

$$TEPL(b_k) = \sum_{j=1}^{m_{b_k}} TP_j^{b_k} \cdot TL_j^{b_k} \tag{3.8}$$

Now, let b be a non-terminal node with its n parent-nodes labeled b_1 to b_n and the probabilities from parent nodes b_1 to b_n to node b be p_{b_1} to p_{b_n}, respectively, as shown in Figure 3.2. We are to compute the expected path length of all paths from root to node b.

First, the summation of probabilities of all paths from root node to node b is

$$STP^b = \sum_{i=1}^{n} p_{b_i} \cdot STP^{b_i} \tag{3.9}$$

where n is the number of parent-nodes of node b. Now, the expected path length from root to node b is

$$\begin{aligned} TEPL(b) &= \sum_{i=1}^{n} \sum_{j=1}^{m_{b_i}} p_{b_i} \cdot TP_j^{b_i} \cdot (TL_j^{b_i} + 1) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{m_{b_i}} (p_{b_i} \cdot TP_j^{b_i} \cdot TL_j^{b_i} + p_{b_i} \cdot TP_j^{b_i}) \end{aligned} \tag{3.10}$$

By equations (3.7) (3.8) and (3.9), equation (3.10) can be reduced to

$$\begin{aligned} TEPL(b) &= \sum_{i=1}^{n} (p_{b_i} \cdot TEPL(b_i) + p_{b_i} \cdot STP^{b_i}) \\ &= \sum_{i=1}^{n} p_{b_i} \cdot TEPL(b_i) + STP^b \end{aligned} \tag{3.11}$$

From equation (3.11), we follow that the expected path length of a non-terminal node at b can be computed by following steps:

step1 for all parent nodes, multiply the probability of parent-node to the node b by the expected path length from root to the parent node which is computed in the previous stage and is readily used.

step2 sum probabilities of all paths from root to node b by equation (9), where the sum of probabilities of all path from root to a parent-node is already computed and readily used.

step3 sum results of step1 and step2

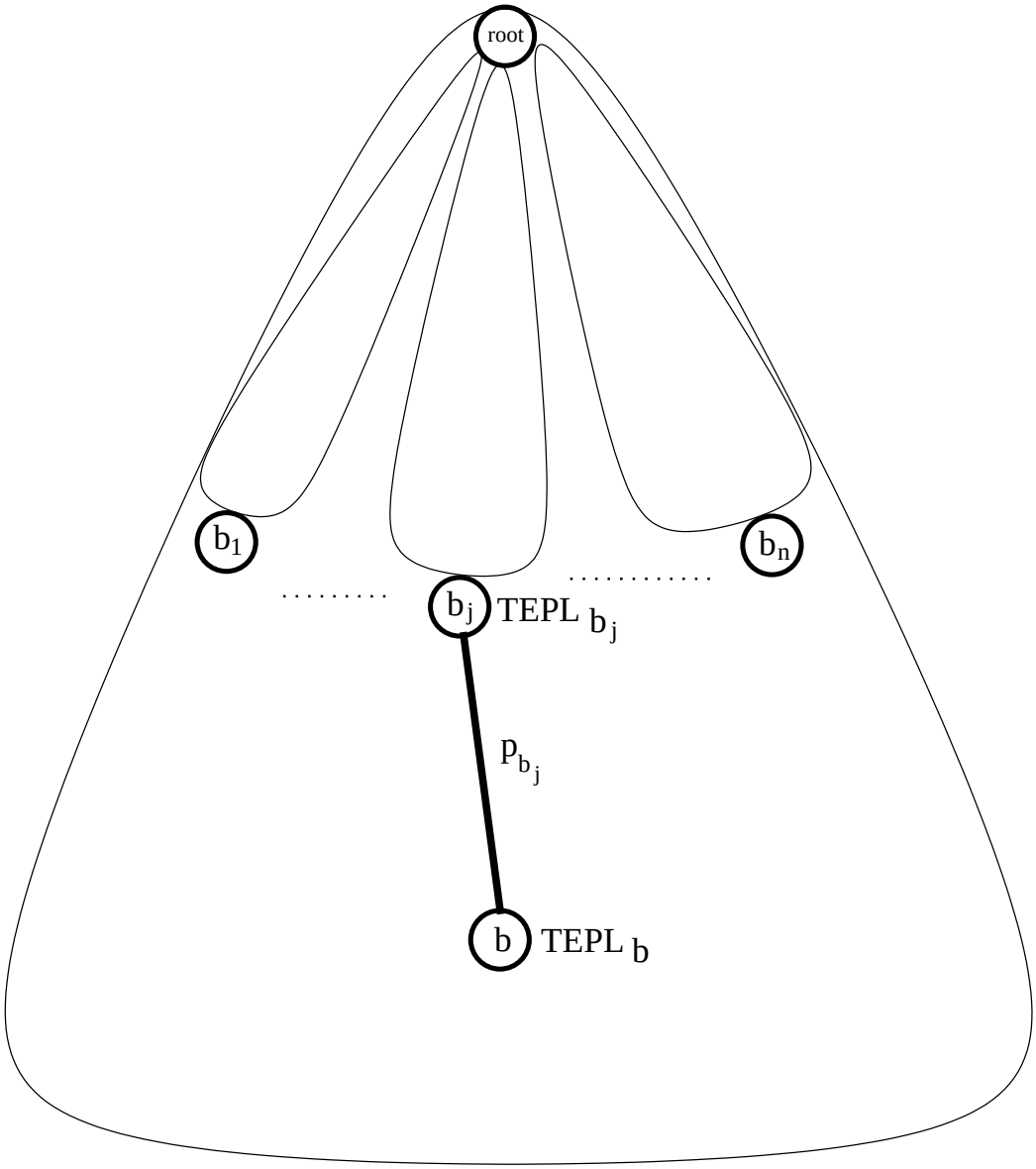

Figure 3.2: Top-down method for calculating the expected path length.

Based on the assumptions that the path length of root is 0 and the probabilities from all paths to root is 1, we can calculate the expected length of the terminal nodes of an ROBDD by top down method. Finally, the expected path length of the whole ROBDD is the summation of the expected path lengths of the two terminal nodes.

3.2.3 Middle-way Method

The third method is used to compute the expected path length of an ROBDD assuming that the expected path lengths of all non-terminal nodes at level $h-1$ have been computed by top-down method and the expected path lengths of all non-terminal nodes at level h have been computed by bottom-up method. Let C be the cut between levels $h-1$ and h and there are l edges crossing C that connect the nodes at levels $h-1$ and h as shown in Figure 3.3. To compute the expected path length of the whole ROBDD, we have to consider all paths connected by l edges. First, we consider how to compute the expected path length connected by a single crossing edge k. In this case, edge k connects the top end-node b_k and the bottom end-node a_k as shown in Figure 3.3. Suppose that the probability of edge k be p_k and there are n paths from root node to node b_k, and m paths from terminals nodes to a_k as shown in Figure 3.3. Then, the expected path length of all paths, $MEPL(k)$, which pass through edge k is

$$\begin{aligned} MEPL(k) &= \sum_{i=1}^{m}\sum_{j=1}^{n} p_k \cdot BP_i^{a_k} \cdot TP_j^{b_k} \cdot (BL_i^{a_k} + TL_j^{b_k} + 1) \\ &= p_k \cdot (\sum_{i=1}^{m}\sum_{j=1}^{n} BP_i^{a_k} \cdot TP_j^{b_k} \cdot BL_i^{a_k} + \sum_{i=1}^{m}\sum_{j=1}^{n} BP_i^{a_k} \cdot TP_j^{b_k} \cdot TL_j^{b_k} + \sum_{i=1}^{m}\sum_{j=1}^{n} BP_i^{a_k} \cdot TP_j^{b_k}) \end{aligned}$$

$$= \quad p_k \cdot (\sum_{j=1}^{n} TP_j^{b_k} \cdot \sum_{i=1}^{m} BP_i^{a_k} \cdot BL_i^{a_k} + \sum_{i=1}^{m} BP_i^{a_k} \cdot \sum_{j=1}^{n} TP_j^{b_k} \cdot TL_j^{b_k} + \sum_{i=i}^{m} BP_i^{a_k} \cdot \sum_{j=1}^{n} TP_j^{b_k}) \tag{3.12}$$

By equations (3.2)(3.5)(3.8)(3.9), equation (3.12) can be reduced to

$$MEPL(k) \quad = \quad p_k \cdot (STP^{b_k} \cdot BEPL(a_k) + TEPL(b_k) + STP^{b_k}) \tag{3.13}$$

From equation (3.13), we follow that the expected path length of all paths passing through edge k connecting node b_k and a_k can be computed by following steps:

step1 sum the expected path length from root of all paths to node b_k, the expected path length of all paths from node a_k to terminals and twice of the total path probability to node b_k. All of them are already computed and are readily used.

step2 multiply the result of step1 and the probability of edge k

Finally, the total expected length of the whole ROBDD is

$$EPL = \sum_{k=1}^{l} MEPL(k)$$

3.3 Algorithms for Finding Low Cost ROBDD

To synthesize a low power PTL cell, we need to find its corresponding ROBDD which minimizes the cost function defined in Section 3.1:

$$\alpha \sum_{i=1}^{n} T(X_i) \cdot Occ(X_i) + (1-\alpha) \sum_{i=1}^{m} \cdot P_i \cdot L_i$$

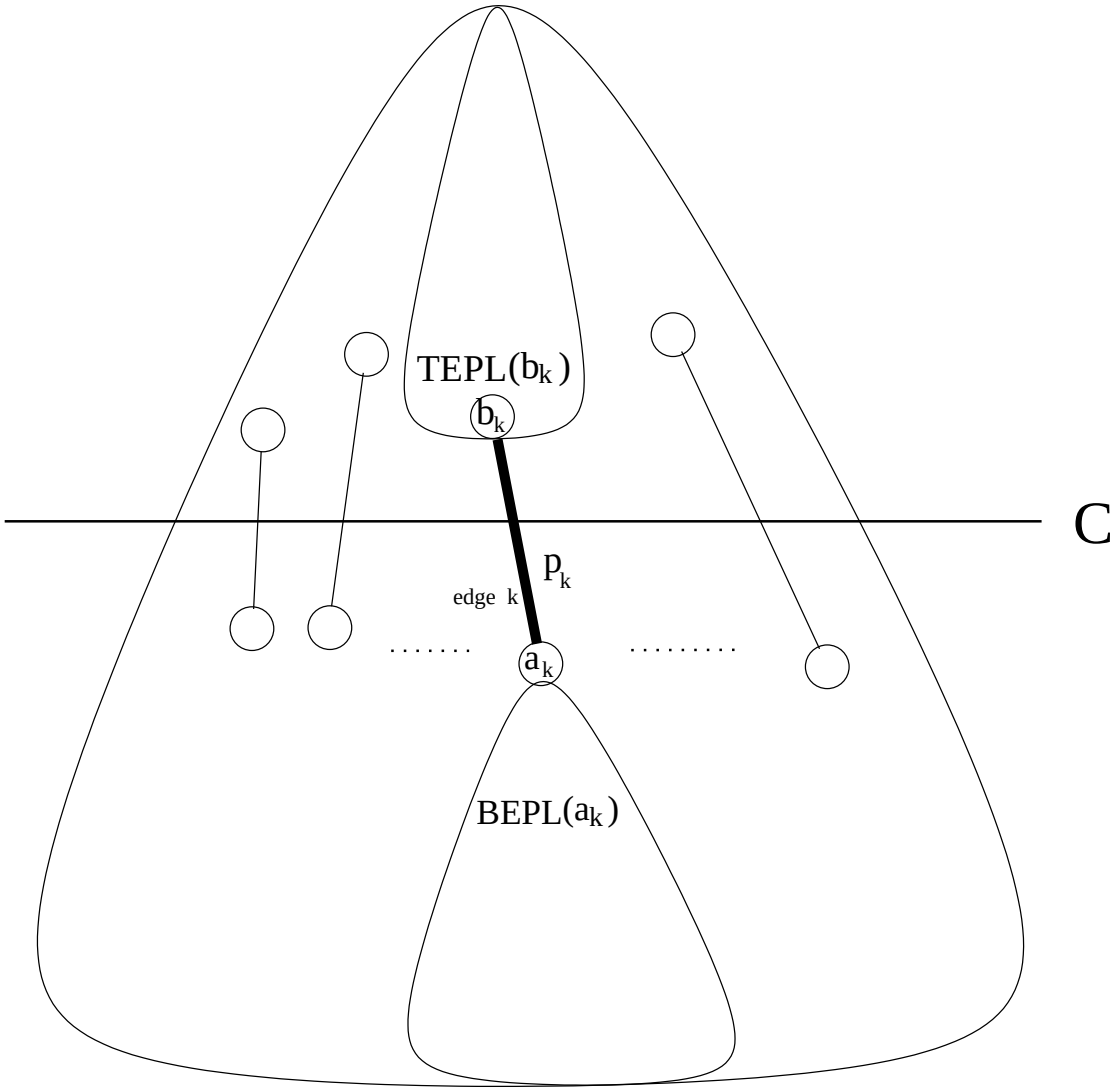

Figure 3.3: Middle-way method for calculating the expected path length.

For a given function, different input ordering will result in different ROBDD structure. Hence our goal is to find an input variable ordering that has an ROBDD with the least cost.

Our algorithm begins with an initial ordering. Then, an iterative loop is entered to improve the initial ordering. In each iteration, a new ordering is found and the old ROBDD is restructured based on the new variable ordering. Next, a new cost is computed for the new ROBDD. Finally, the cost function is updated. The loop continues till the criterion for stopping is met. The procedure is shown in Figure 3.4. The detailed description is explained as follows.

In step 1, an initial oredering needs to be determined to construct an initial ROBDD. It can be determined by constructing a minimum size ROBDD. In this case, algorithms proposed in [12] [13] can be used. Step 3 is to compute the cost of the initial ROBDD. For the computation of the expected path length of the ROBDD, bottom-up and top-down methods proposed in Section 3.2 are used and all computed data for each node is stored at the node. Step 5 finds a new input ordering. Window permutation algorithm [12] [13] which finds a local minimal in a window size k and sifting algorithm [11] which looks for a suitable position for one variable to move in each iteration can be used. With the new input variable ordering, step 6 restructures the old ROBDD to a new ROBDD. Transposition operator [14] is used which restructures an ROBDD by several simple ROBDD operations. For example, if a new input ordering is that the orders of x_i and x_j are swapped, the following formula can be used.

$$f_{x_i \leftrightarrow x_j} = ITE(x_i \odot x_j, f, f_{x_i \leftarrow \bar{x_i}, x_j \leftarrow \bar{x_j}})$$

Note that the swapping is not limited to adjacient variables.

In step 7, the new cost for the new ROBDD is computed. For the expected length of the new ROBDD, similar to step 6, it needs not be computed from scratch. Instead, we will use the information for the old ROBDD to calculate the new expected length of ROBDD. Suppose that we have a new ordering that variables at levels i and j $(i < j)$ are swapped as shown in Figure 3.5. Since the part of ROBDD above level i and the part of ROBDD below level j will not be changed as seen in Figure 3.5, we do not have to recompute them. We only have to compute the new expected path length for the nodes in the middle. On the one side, the bottom-up method calculates the expected path length from nodes at level $j + 1$ up. On the other side, the top-down method calculates the expected path length from nodes at level $i - 1$ down. When variables on the two sides meet in the middle, the middle-way method can be used.

```
     Algorithm find-least-cost(f)
     Input:f = Boolean function;
     Output: return cost and the corresponding ROBDD;
     Begin
(1)      find an initial variable ordering for ROBDD of f;
(2)      build an initial ROBDD;
(3)      compute the cost of the ROBDD;
(4)      while ( not stop ) do
(5)          find new variable ordering;
(6)          restructure the ROBDD to the new variable ordering;
(7)          compute the cost for the new ROBDD;
(8)          update cost function;
(9)          check if the stopping criterion needs to be set;
         endwhile
(10)     return cost and the corresponding ROBDD;
     End
```

Figure 3.4: The *find-least-cost* algorithm

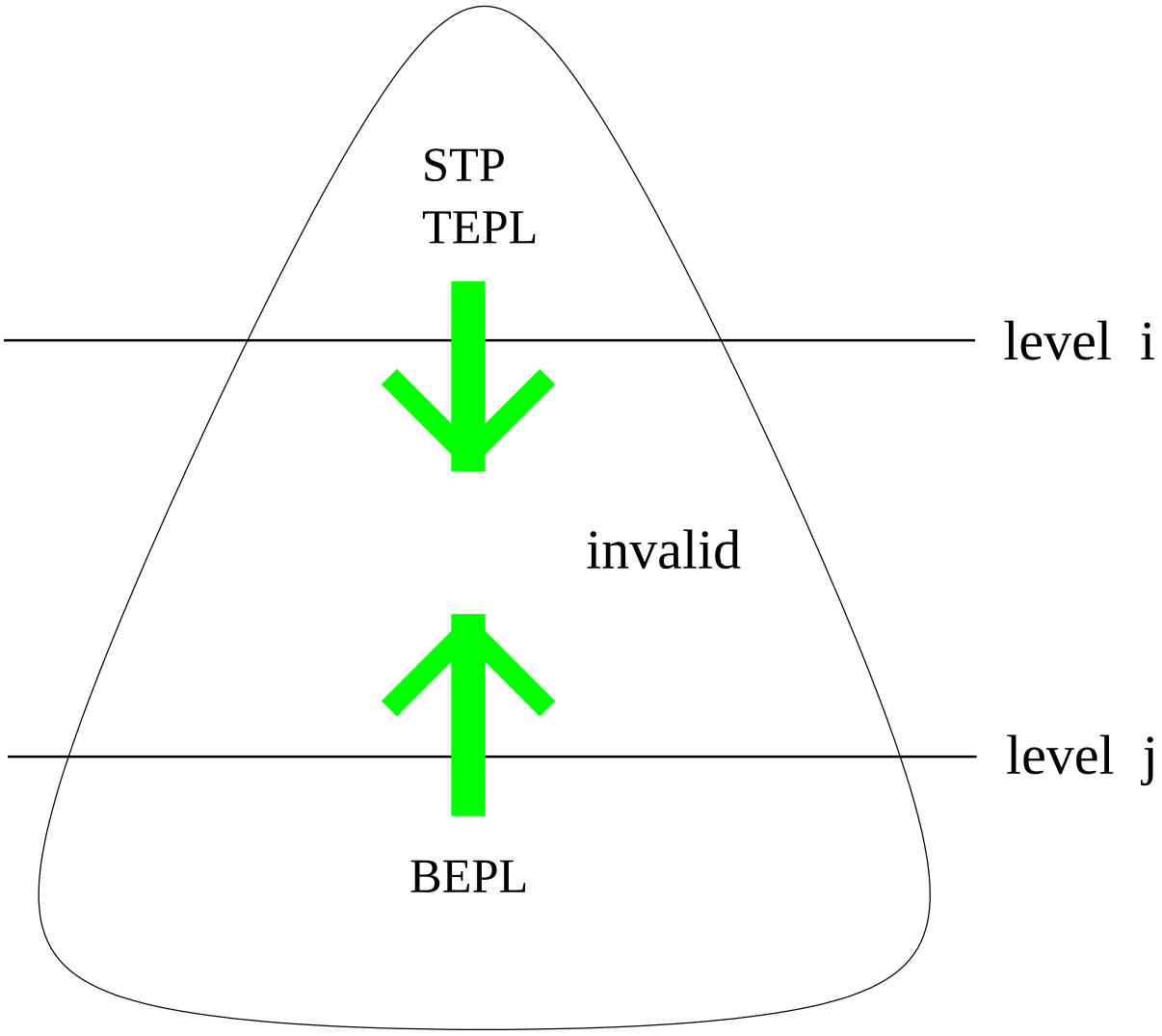

Figure 3.5: Computation for parts lying between the two swapped variables.

Chapter 4

Experimental Results

Our experiment is performed on SUN-Ultra Enterprise 150. Software platform is based on the BDD package in SIS [15]. *LGSynth93* benchmark suite is used in our experiment.

The first experiment is conducted to find a minimum expected path length ROBDD. First, sifting algorithm [11] is used to find a minimum-sized ROBDD which can be used as our initial ROBDD. The reason for this initial ordering heuristic is that a smaller size of ROBDD may gain a smaller expected length of ROBDD. In subsequent steps, window permutation is used to generate a new input ordering and the expected path length for the new ROBDD is computed. Finally, the probabilities of input variables are set to be equal.

Table 4.1 shows the results with our proposed expected path length model as the cost function. Column *min_node* gives the results of ROBDD with initial ordering whose objective is to find an ROBDD with minimum size. Column *min_exp_path* gives the results of ROBDD with expected path length as objective. Columns *path* and *size* give the expected path length and size of ROBDD, respectively. In column *ratio*, we compute the ratio of the results of minimum nodes to the results of minimum expected path length.

It is clear from Table 4.1 that in most cases, the expected path length can be reduced at the cost of small increase of the number of nodes except *cordic* and *f51m*. In some cases (*con1, vg2, cm150a, etc.*), we found that both the size and the expected path length of ROBDD are reduced. On the average, the expected path length is reduced by 25% while the the number of nodes is increased by 6%.

The second experiment is conducted using our proposed power consumption model as the cost function. In this cost function, weighting factor α needs to be set to reflect the weight of load capacitance and diffusion capacitance. Based on the example given in [16], where 1 μm CMOS technology is used, the ratio of load capacitance and diffusion capacitance in an cascaded inverter is about 1 : 1. Therefore, we set the weight factor $\alpha = 0.5$ in this experiment. We also assume that the probability $Prob(X_i = 0) = Prob(X_i = 1) = \frac{1}{2}$.

Table 4.2 shows the results. Column *min_node* gives the same results produced by the first experiment plus the defined cost function computed for each ROBDD. Column *min_cost* gives the results of ROBDD with our proposed power function as objective. The column *cost* gives the defined cost computed for each ROBDD. From this Table, it is clear that our proposed algorithm is superior to the minimum-sized algorithm not only in cost (12%) but also in size (4%) and path length (20%). This is because our power consumption model provides a more flexible mechanism for "jumping away" the local minimum.

The third experiment is conducted to observe the influence of variable probabilities. We use two types of probabilities setting, ascending and descending, in this experiment. For n variables appearing in ROBDD from top to bottom, the probabilities are set to

$\frac{1}{n+1}, \frac{2}{n+1}, ..., \frac{n}{n+1}$ by ascending probability ordering and to $\frac{n}{n+1}, \frac{n-1}{n+1}, ..., \frac{1}{n+1}$ by descending probability ordering. Table 4.3 and Table 4.4 shows the results for probabilities in descending order and in ascending order, respectively. From these Tables, we can see that for different input probabilities, the best ROBDD found are different.

Table 4.1: Results of expected path length with equal input probability

circuit	min_node		min_exp_path		ratio (%)	
	path	size	path	size	path	size
5xp1	37.19	78	31.63	90	85.04	115.38
alu4	62.33	749	48.65	906	78.05	120.96
b12	29.52	77	22.03	79	74.64	102.60
con1	6.44	17	6.06	16	94.17	94.12
cordic	17.76	109	11.15	131	62.80	120.18
ex1010	82.16	1482	80.47	1550	97.94	104.59
inc	32.42	106	27.08	111	83.52	104.72
sao2	21.99	108	10.69	129	48.62	119.44
vg2	46.84	238	31.49	273	67.24	114.71
misex1	24.09	64	22.22	70	92.22	109.38
cm150a	5.50	33	3.50	33	63.64	100.00
cm151a	9.00	34	6.50	36	72.22	105.88
cm162a	21.50	57	11.70	58	54.43	101.75
cm163a	18.09	46	11.70	43	64.68	93.48
cm85a	15.97	41	8.44	44	52.84	107.32
mux	5.50	33	3.59	35	65.34	106.06
z4ml	18.25	32	18.25	32	100.00	100.00
f51m	27.98	61	27.33	67	97.65	109.84
pcle	35.51	101	22.50	90	63.36	89.11
traffic	20.88	41	14.88	41	71.26	100.00
Average					74.48	105.98

Table 4.2: Results of new cost function of PTL with equal input probability

circuit	min_node			min_cost			ratio (%)		
	path	size	cost	path	size	cost	path	size	cost
5xp1	37.19	78	27.22	31.48	74	23.87	84.66	94.87	87.69
alu4	62.33	749	123.79	61.13	728	120.56	98.07	97.20	97.39
b12	29.52	77	23.26	24.31	73	20.16	82.37	94.81	86.66
con1	6.44	17	5.09	6.31	17	5.03	98.06	100.00	98.77
cordic	17.76	109	22.25	12.43	94	17.71	70.00	86.24	79.60
ex1010	82.16	1482	225.08	81.96	1477	224.36	99.76	99.66	99.68
inc	32.42	106	28.34	29.02	101	26.01	89.49	95.28	91.78
sao2	21.99	108	24.00	16.08	115	21.91	73.09	106.48	91.32
vg2	46.84	238	52.17	43.49	235	50.12	92.86	98.74	96.08
misex1	24.09	64	19.17	23.22	63	18.61	96.37	98.44	97.07
cm150a	5.50	33	6.75	3.50	33	5.75	63.64	100.00	85.19
cm151a	9.00	34	8.50	6.50	36	7.50	72.22	105.88	88.24
cm162a	21.50	57	17.25	13.77	50	12.51	64.03	87.72	72.51
cm163a	18.09	46	14.17	11.70	40	10.23	64.68	86.96	72.16
cm85a	15.97	41	12.73	9.48	41	9.49	59.39	100.00	74.54
mux	5.50	33	6.75	3.50	33	5.75	63.64	100.00	85.19
z4ml	18.25	32	12.63	18.25	32	12.63	100.00	100.00	100.00
f51m	27.98	61	20.62	27.67	61	20.46	98.88	100.00	99.24
pcle	35.51	101	29.26	22.50	88	21.13	63.36	87.13	72.21
traffic	20.88	41	14.44	16.13	38	11.69	77.25	92.68	80.95
Average							80.59	96.60	87.81

Table 4.3: Results of probabilities in descending order.

circuit	min_node			min_cost			ratio (%)		
	path	size	cost	path	size	cost	path	size	cost
5xp1	36.92	79	25.27	30.17	79	21.62	81.72	100.00	85.56
con1	6.84	17	4.98	6.42	18	4.8	93.91	105.88	96.29
inc	29.72	106	24.84	27.14	108	22.62	91.32	101.89	91.06
misex1	22.1	64	16.68	20.72	67	15.58	93.77	104.69	93.43
cm85a	19.08	41	13.24	9.61	44	8.55	50.39	107.32	64.55
Average							82.22	103.96	86.18

Table 4.4: Results of probabilities in ascending order.

circuit	min_node			min_cost			ratio (%)		
	path	size	cost	path	size	cost	path	size	cost
5xp1	36.76	78	24.74	33.86	76	22.87	92.10	97.44	92.43
con1	6.3	17	4.71	6.2	18	4.6	98.43	105.88	97.62
inc	33.48	106	26.72	24.47	103	20.78	73.10	97.17	77.77
misex1	25.39	64	18.33	21.2	67	15.71	83.50	104.69	85.74
cm85a	18.54	41	12.98	7.8	41	7.43	42.06	100.00	57.29
Average							77.84	101.04	82.17

Chapter 5

Conclusions

Given that the power dissipation of a PTL cell is affected by the capacitance which consists of load capacitance and diffusion capacitance, we have shown that we can translate the synthesis of low power PTL cell to minimization of the capacitance of an ROBDD. We have proposed three efficient methods to calculate the expected length of ROBDD during variable reordering. The experimental results indicate that our proposed power dissipation model provides a better cost function for PTL power consumption to produce small ROBDD both in node size and path length.

Bibliography

[1] P. Buch, A. Narayan A. R. Newton, and A. Sangiovanni-Vincentelli, "Logic Synthesis for Large Pass Transistor Circuits", *ICCAD*, Nov. 1997.

[2] K. Yano, Y. Saski, K. Rikino, and K. Seki, "Top-down Pass-transistor Logic Design" *IEEE JSSC*, Vol. 31, No. 6, June 1996.

[3] N. Zhuang, M. V. Scotti, and P. Y. K. Cheung, "PTM: Technology mapper for pass-transistor logic", *IEECDT*, Vol. 146, No 1, Jan. 1999.

[4] F. Ferrandi, A. Macii, E. Macii, M. Poncino, R. Scarsi, and F. Somenzi, "Symbolic Algorithms for Layout-Oriented Synthesis of Pass Transistor Logic Circuits", *ICCAD*, Nov. 1998.

[5] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-Power COMS Digital Design", *IEEE Journal of Solid-State Circuits*, Vol. 27, No. 4, pp. 473-484, April 1992.

[6] S. B. Akers, "Binary decision diagrams", *IEEE Trans. Comput.*, Vol. C-27, pp. 509-516, June 1978.

[7] R. E. Bryant, "Graph-based algorithms for boolean function manipulation", *IEEE Trans. on Comput.*, Vol. C-35, No. 8, August 1986.

[8] K. S. Brace, R. L. Rudell, and R. E. Bryant, "Efficient Implementation of a BDD Package", *DAC*, 1990.

[9] H. R. Andersen and H. Hulgaard, "Boolean Expression Diagrams", *LICS*, 1997.

[10] M. Fujita, H. Fujisawa, and N. Kawato, "Evaluations and Improvements of a Boolean Comparison Method Based on Binary Decision Diagrams", *ICCAD*, 1988.

[11] R. Rudell, "Dynamic variables ordering for ordered binary decision diagrams", *ICCAD*, 1993.

[12] M. Fujita, Y. Matsunaga, and T. Kakuda, "On Variable Ordering of Binary Decision Diagrams for the Application of Multi-level Logic Synthesis", *EDAC*, pp 50-54, Mar. 1991.

[13] N. Ishiura, H. Sawada, and S. Yajima, "Minimization of Binary Decision Diagrams Based on Exchanges of Variables" *ICCAD*, pp. 472-475, Nov. 1991.

[14] K. H. Wang, T. T. Hwang, and C. Chen, "Restructuring Binary Decision Diagrams Based on Functional Equivalence", *EDAC*, pp 261-265, Feb. 1993.

[15] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, A. Sangiovanni-Vincentelli, "SIS: A System for Sequen-

tial Circuit Synthesis", *Electronics Research Laboratory* Memorandum No. UCB/ERL M92/41, 4 May 1992.

[16] N. H. E. Weste, K. Eshraghian, *Principles of CMOS VLSI Design 2nd Edition*, Addison Wesley, 1993.