Low Power Technology Mapping by Hiding High-Transition Paths in Invisible Edges for LUT-Based FPGAs *

Chua-Chin Wang, & Cheng-Pin Kwan
Department of Electrical Engineering
National Sun Yat-Sen University
Kaohsiung, Taiwan 80424 China

***Abstract*— Considering that the connection switches possessing large resistance and capacitance in lookup-table-based (LUT-based) Field Programmable Gate Array (FPGA) routing channels consume a great portion of total power, a power-saving technology mapping algorithm is proposed tending to reduce the transition density on "visible" edges of the mapped logic circuits by hiding the paths with high transition activity in "invisible" edges. Meanwhile, the number of LUTs is also kept optimally small compared to prior technology mapping method. Finally, detailed simulation results of certain benchmark circuits are presented to verify the performance of the proposed algorithm.**

I. Introduction

Recently there has been a surge of interest in low-power devices and design techniques mainly due to the booming of personal wireless communication tool, while many papers have been published describing power-saving skills for use in digital systems [2], [3], including voltage scaling [3], and transition reduction to save dynamic power, [7]. Thus, this low power demand also affect the design methodology of FPGAs, [5], [6]. Tsui *et al.* [6] proposed an NAND tree decomposition approach in which postorder and preorder traversals of the tree are used so that the computation complexity is costly. Besides, their method gains 21% transition reduction on the penalty of 12.6% increase of the area. Tiwari *et al.* [5] also proposed their power-saving technology mapping which utilized DAG (directed acyclic graph) covering. Since tree matching is used in this method, the computation cost is also high. The transition reduction is about 10% while the penalty of increase of the chip area is about 12%. Besides, most important of all, these prior works did not discuss the real effect for LUT-based technology mapping. Though Brown *et al.* [1] proposed the bin-packing algorithm for LUT-based FPGAs to achieve the goal of the minimal number of LUTs used in technology mapping, the power factor was not considered.

Note that no matter which programming technology, e.g., SRAM-based cells or anti-fuses, is used for the interconnection in FPGAs, the connection switch can be deemed as a RC delay circuit. The range of R is from hundreds of ohms to 4 KΩ, while the C is from 10 to 20 ff according to [1]. These switches certainly consume a great portion of power. In contrast, the power resulted from one transition inside the LUTs is relatively much smaller due to that one transition inside the LUTs only causes one RAM output bit to change states. Hence, reducing power dissipation can be achieved by hiding the edges of the original circuit with high transition density in the LUTs while exposing the edges with low transition density, if necessary, outside the LUTs.

II. Low Power Technology Mapping

In a typical FPGA design flow, the input to a technology mapper is provided by a logic optimizer. The input is assumed to be a general AND-OR form, $z = f(a, b, c, \ldots)$. Each input signal of z is associated with an turn-on probability, $P(x)$ if $x = 1$, and a transition density function, $D(x)$. Each input signal to z, x, is assumed to be a **independent variable** (IV) which implies that $P(x)$ and $D(x)$ are also independent.

From the viewpoint of power consumption, a state transition on a net outside of a LUT, called a "visible edge", causes either charge or discharge of RC circuits where the range of R is from hundreds of ohms to 4 KΩ, while the C is from 10 to 20 ff according to [1]. In contrast, if one state transition occurs on a net inside of a LUT, called an "invisible edge", it only results in a memory read operation. The ratio is about 26 μW to 3.6 μW at 10 MHz empirically. Thus, in order to reach power-saving, it is required to hide the edges, or wires, with high transition density under the K constraint of the LUT.

The transition density of a net in a logic circuit can be calculated by the equation,

$$D(y) = \sum_i P(\frac{\partial y}{\partial x_i}) D(x_i), \qquad (1)$$

where y is the output, x_i's are inputs, and $\frac{\partial y}{\partial x_i} = y(x_i) \bigoplus y(\overline{x_i})$ [4].

A. Power-Saving Guidelines

The feature of the matrix-based FPGA technology mapping is the K-input constraint. (A terminology, "bin", is used to denote a LUT in prior works [1].) Hence,

*This research was partially supported by National Science Council under grant NSC 86-2215-E-110-013.

0-7803-3583-X/97 $10.00 ©1997 IEEE

traditional mappers focus on the minimization of the number of LUTs used to realize a circuit or the levels of LUTs which are critical to the speed of the mapped circuit. The Boolean expression in a two-level AND-OR format (or OR-AND format) can be deemed as a tree while the output gate of a Boolean expression, thus, is the root of the tree, as shown in Fig. 1. The inputs to the leaf nodes are presumed to be IVs. We can summarize a few principles to keep the optimality of power consumption and the number of LUTs.

(Rule 1). If the number of inputs of a node is K, then this node won't be either merged or combined later in the procedure. It forms a K-input bin. Its output is deemed as an independent variable associated with $P(\cdot)$ and $D(\cdot)$ computed according to the inputs' turn-on probabilities and density functions, as shown in Fig. 2.

(Rule 2). If the the number of IVs of an AND node exceeds K, then this node is decomposed into a series of bins. The IVs of the original node are sorted by their densities. Merge from the IVs with smallest transition density so that one new bin with K IVs is generated. Then, The output of this bin and the next $K-1$ smallest IVs are merged into another new bin. Repeat this procedure till every original IV is merged into one bin, as shown in Fig. 3. Note that the last a few IVs can be merged into one new bin which might not have K inputs.

(Rule 3). If the the number of IVs of an OR node exceeds K, then this node is decomposed into a forest of bins. The IVs of the original node are sorted by their densities. In contrast to the decomposition of an AND node, the OR node should be decomposed into a number of parallel bins in order to reduce the transition activity according to Eqn. (1). We firstly compute the maximum number of K-input bins required to contain the IVs of the original OR node. Assume there are Bin_num of bins are needed. The IVs are iteratively assigned to the bins according to their density ascendingly, as shown in Fig. 4. Note that the last a few IVs can be merged into one new bin which might not have K inputs.

(Rule 4). In the same level of the logic circuit, either AND level or the OR level, the bins are sorted according to their individual output transition density. Start from the bin with maximum transition density. Search the bin list descendingly to find another bin which can be merged with the bin with maximum transition density to be a LUT which has K inputs totally. If the merge is successful, then these two bins are deleted from the bin list and combined into one bin with K inputs. Then, this new bin with K inputs will be deemed as an IV according to Rule 1. An example is shown in Fig. 5.

B. Power-Saving Algorithm Pseudo-codes

Basing on the power-saving principles, we can formulate the lower power technology mapping algorithm. The main procedure, Adjust_Tree(), is shown as follows.

```
Adjust_Tree(z) {  ** z is the output node of the given logic function.
  Child_node_list ← z's inputs;
  for(Child_node ∈ Child_node_list) {
    if( Child_node is not an IV ) { ** This checks whether Child_node is a bin.
      if( Child_node's function == z's function ) {
        Merge this Child_node with z;
        Add inputs of Child_node into Child_node_list;
      }
      else
        Child_node ← Adjust_Tree(Child_node);
    }
  }
  if ( z's every input is an IV ) {
    if ( the number of inputs of z > K )
      if ( node z is an AND gate )
        z ← Cascade_Tree(z); ** Rule (2)
      else
        z ← Expand_Tree(z); ** Rule (3)
    }
  else {
    z ← Merge_Tree(z);
  }
    if ( the number of inputs of z = K )
      z degenerates into an IV;
} ** End of Adjust_Tree()
```

The following procedure is to implement the task of Rule (4) which not only merges bins and IVs into K-input bins, but also keeps the number of bins optimally small by employing a greedy approach.

```
Merge_Tree(z) {
  Old_IV_list ← ∅;
  Old_Bin_list ← ∅;
  New_Bin_list ← ∅;
  do {
    Old_IV_list ← the inputs of z which are IVs;
    Sort Old_IV_list on their density ascendingly;
    Old_Bin_list ← the inputs of z which are bins;
    Sort Old_Bin_list on their density descendingly;
    while ( Old_Bin_list is not ∅ ) {
      t ← the bin with the maximum density;
      k ← the number of inputs of t;
      if ( Search Old_Bin_list descendingly and find a bin able to be merged with t to be a K-input bin) {
        remove t and the searched bin from Old_Bin_list;
        decompose root z, and then merge t and the searched bin into a new bin which is marked as an IV;
** It is as shown in Fig. 5.
        add this new bin into New_Bin_list;
      }
      else if ( the number of IV left in Old_IV_list ≥ K−k) {
        merge the first K − k IVs of Old_IV_list with t into a new bin, t';
        remove the first K − k IVs from Old_IV_list;
        remove t from Old_Bin_list;
        add t' into New_Bin_list and mark it as an IV;
      else if( Old_Bin_list ≠ ∅) { ** t will be hidden by merging with the bin with smallest density.
        q ← the bin with minimum density in the Old_Bin_list;
```

```
        remove q from Old_Bin_list;
        if( number of q's inputs +k ≤ K)
          decompose root z and then merge t and q into
a new bin, t';
        else
          decompose root z and then merge t and q's out-
put into a new bin, t';
        add t' into New_Bin_list;
      }
    }
    else
      add t to New_Bin_list;
  }  ** End of while loop.

  if( Old_IV_list ≠ ∅ ) {
    Sort New_Bin_list descendingly according to the den-
sity of their output;
    Pt ← first bin of New_Bin_list;
    do {
      v ← the number of IV left in Old_IV_list;
      if ( Search New_Bin_list from Pt and find a bin, t,
not marked as an IV) {
        Pt ← t;
        f ← number of t's inputs;
        c ← minimum(v, K-f);
        remove c IV's from Old_IV_list, and then decom-
pose root z merging with the c IV's into a new bin, t';
        if( the number of t's inputs = K)
          mark t' as an IV;
      }
    } while( Old_IV_list is not ∅ and Pt is not the last
bin of New_Bin_list)
  }

  if (z can be merged with more than one bin in the
New_Bin_list or the IVs in the Old_IV_list) {
    z is merged with the available bin with largest den-
sity or IVs in the Old_IV_list into one bin;
  }
  if (z's inputs are all IVs and the number of IVs > K)
{
    if( node z is an AND gate)
      z ← Cascade_Tree(z);
    else
      z ← Exapnd_Tree(z);
  }
 } while ( the number of inputs of z > K)
 return(z);
} ** End of Merge_Tree()
```

The final result of the mapping is stored in the New_Bin_list of the root node z of the given circuit.

III. Simulation and Analysis

We use five MCNC LGSynth91 benchmark circuits as simulation examples to compare the proposed algorithm and the bin-packing method of [1] so that the power reduction rate can be demonstrated. The results of these circuits simulated by HSPICE and CADENCE are tabulated in the following.

circuit	bin-packing		low power	
	# LB	power	# LB	power
con1.pla	6	3.8352 mW	6	3.3759 mW
rd53.pla	18	4.7280 mW	18	4.2790 mW
5xp.pla	12	7.9593 mW	12	7.2653 mW
bw.pla	19	9.9325 mW	19	8.7214 mW
mis.pla	12	3.2932 mW	12	2.9717 mW

Table. 1 : The performance comparison of benchmark circuits mapped by the proposed algorithm and the bin-packing algorithm.

Fig. 8 illustrates the LUT-based LB used in our simulation, which is close to the 5-input CLB of Xilinx 3000. Fig. 9 shows the result of the bin-packing algorithm for con1.pla, while Fig. 10 is the technology mapping result of con1.pla obtained by the proposed algorithm. Basing upon the results of Table 1, the proposed low power technology mapping method provides averagely 10.38% of power reduction without any penalty of area increase.

IV. Conclusion

Our method can achieve the goal of lower power technology mapping while not pay the price of area. It performs better than prior works. In addition, we consider the worst case improvement of power saving in our simulation by not including the effect of programming switches into the mapped circuits. If the switches are also considered in the visible edges, the reduction ratio will be significantly increased.

References

[1] D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, *Field-Programmable Gate Arrays*, Kluwer Academic Publisers, 1992.

[2] A. P. Chandrakasan, S. Sheng, and R. W. Broderson, "Low-Power CMOS digital design," *IEEE J. Solid-State Circuits*, vol. 27, no. 4, pp. 473-484, April 1992.

[3] Z. Chen, J. Scott, J. Burr, and J. D. Plummer, "CMOS technology scaling for low voltage low power applications," *1994 IEEE Symp. on Low Power Electronics*, pp. 56-57, Oct. 1994.

[4] F. N. Najm, "Transition density: a new measure of activity in digital circuits," *IEEE Trans. Computer-Aided Design*, vol. 12, no. 2, pp. 310-323, Feb. 1993.

[5] V. Tiwari, P. Ashar, and S. Malik, "Technology mapping for low power," *30th ACM/IEEE Design Automation Conference*, pp. 74 - 79, 1993.

[6] C.-Y. Tsui, M. Pedram, and A. D. Despain, "Technology decomposition and mapping tageting low power dissipation," *30th ACM/IEEE Design Automation Conference*, pp. 68 - 73, 1993.

[7] C.-C. Wang, and M.-D. Jeng, "Power Estimation of Internal Nodes for Finite State Machine Using Gray Code Encoding in State Assignment," *National Computer Symposium 1995*, pp. 842 - 949, Dec. 1995.

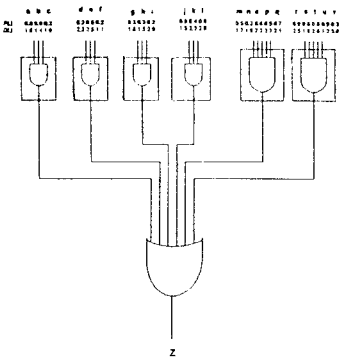


Figure 1: An example circuit in AND-OR form

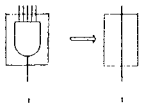
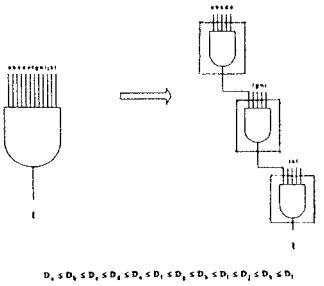
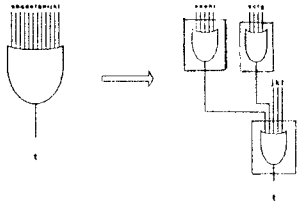


Figure 2: A K-input bin collapsed into an IV



$D_a \le D_b \le D_c \le D_d \le D_e \le D_f \le D_g \le D_h \le D_i \le D_j \le D_k \le D_l$

Figure 3: Cascade-tree() example



$D_a \le D_b \le D_c \le D_d \le D_e \le D_f \le D_g \le D_h \le D_i \le D_j \le D_k \le D_l$

Figure 4: Expand-tree() example

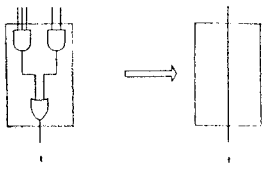


Figure 5: sucessful merge into a K-input bin

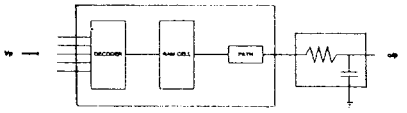
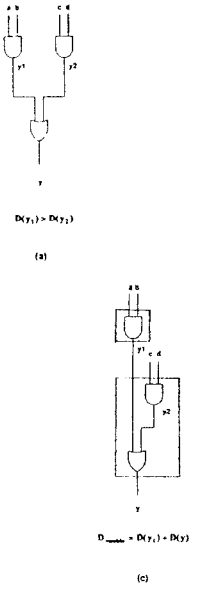


Figure 6: Equivalant circuit of 5-input CLB of Xilinx 3000



$D(y_1) > D(y_2)$

(a)

$D_{new} = D(y_2) + D(y)$

(b)

$D_{new} = D(y_1) + D(y)$

(c)

Figure 7: Power reduction by hiding edges with high transition activity

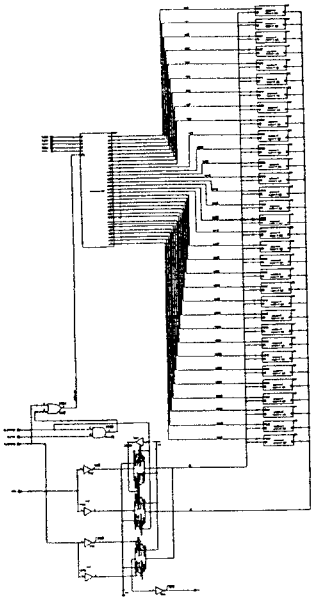
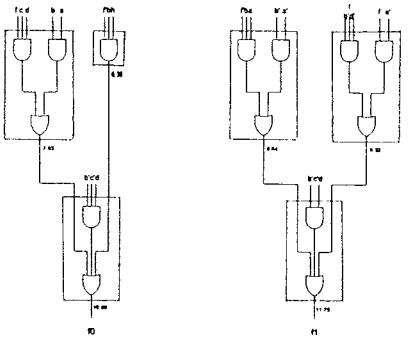
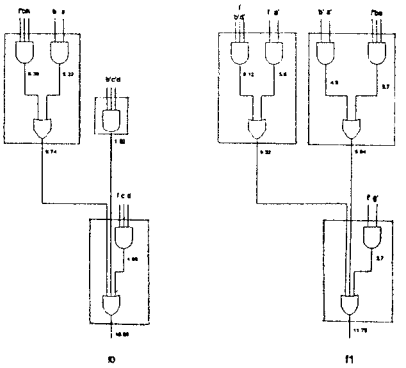


Figure 8: LUT-based LB schematic diagram



	f	b	c	d	a	h	g
P(+)	0.5	0.6	0.6	0.3	0.5	0.6	0.7
D(+)	9.0	8.0	6.0	4.0	2.0	2.5	2.0

Figure 9: con1.pla by the bin-packing algorithm



	f	b	c	d	a	h	g
P(+)	0.5	0.6	0.6	0.3	0.5	0.6	0.7
D(+)	9.0	8.0	6.0	4.0	2.0	2.5	2.0

Figure 10: con1.pla by the proposed low-power technology mapping algorithm