

# Logic Synthesis & Optimization

Spring 2016, Solutions to Homework # 5  
Prepared by Priyank

Solutions to the first 3 questions on HW 5.

- 1)  $F = ab + ad + cd$  and  $G = a + c$ . Conjunctive Bi-decomp is possible if  $F \subseteq G$ , and this is true because cube  $a$  in  $G$  contains cubes  $ab, ad$  in  $F$ ; and  $c$  contains  $cd$ . So the conjunctive decomposition is possible. However, with the same  $F, G$ , disjunctive decomposition is not possible because for OR-bi-decomp,  $F$  has to be larger than  $G$ . On the other hand XOR/XNOR decompositions are always possible:  $F = G \oplus (G \oplus F)$ .

$F = G \cdot H$  is easy, as you can compute  $H$  as  $F$  simplified with  $G'$  as DC. The answer is  $H = d + ab$ . OR decomp is not possible. Since XNOR decomp is always possible and  $G$  is given, compute  $H = G \oplus F$  (refer to Theorem 6) in Yang and Ciesielski's paper.

Some of you got confused by trying to identify  $x$ -dominators on the BDD. An  $x$ -dominator will always give you an XNOR (XOR) type decomposition. But the converse is not true. If you cannot find an  $x$  dominator - then it does not mean that you cannot perform XNOR decomposition of the function. It just means that you cannot identify a "trivial XNOR" structure on the BDD with that particular variable ordering. In such a case, you can ALWAYS use the ITE operator and compute  $H = G \oplus F$ .

- 2) For Ashenhurst-Curtis, you can draw the BDD with the given order and see that a simple disjunctive decomp is not possible. In order to perform the decomposition, we will have to encode the bound-set cubes corresponding to each cut-node. The BDD for  $f$  is shown below:

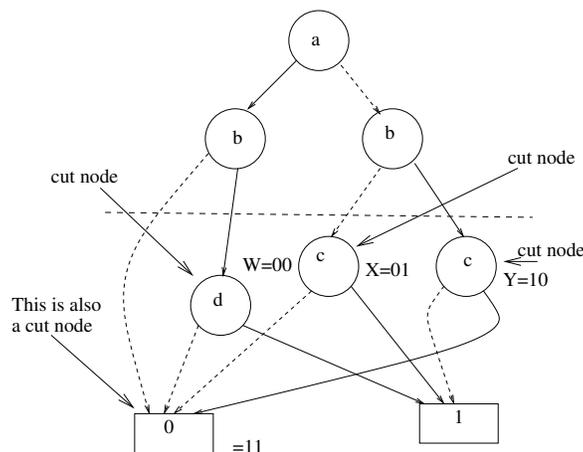


Fig. 1. Ashenhurst-Curtis Decomposition on BDD

Now comes the tricky part. You are supposed to encode ALL THE CUBES of the bound set. Please follow this basic procedure:

- Starting from the root, traverse ALL paths and “pierce the cut”. These paths correspond to the cubes in the bound set.
- The **first node that you hit across the cut** represents the subfunction that will be implemented in the free-set part.
- In the above BDD, you will find that the path corresponding to cube  $a \cdot b'$  hits terminal node **0** - this is the first node after the cut on this particular path ( $ab'$ ). Which means that corresponding to bound-set cubes  $ab'$ , the subfunction implemented in the free-set part will be  $a \cdot b' \cdot 0$ .
- This does not mean that you can ignore to encode the terminal node. If you ignore this terminal node, you will get an **incorrect** answer: *i.e.*,  $f$  will not compute zero when  $ab'$  is applied.

Let us encode the 4 cut-set nodes  $W, X, Y, Z$  with two bits  $g_0, g_1$ . Let  $Z = 11$  correspond to the encoding of the terminal **0** node.  $g_0 = 1$  for node  $Z$  and node  $Y$ .

**Again, note the following:** For node  $Z$ , only the incoming path  $ab'$  is to be considered for  $g_0$ . Why not the other ones? Because, the bound-set cubes corresponding to the other paths will be encoded by the remaining cut-set nodes. This is because the remaining paths hit cut-nodes  $W, X, Y$ .

$$g_0 = a \cdot b'(nodeZ) + a' \cdot b(nodeY) \quad (1)$$

$$g_1 = a \cdot b'(nodeZ) + a' \cdot b'(nodeX) \quad (2)$$

$$H = g'_0 g'_1 d + g'_0 g_1 c + g_0 g'_1 c' + g_0 g_1 0 \quad (3)$$

$$= g'_0 g'_1 d + g'_0 g_1 c + g_0 g'_1 c' \quad (4)$$

You may try any other encoding, but traverse the bound-set (paths) cubes properly! In the above, apply  $ab'$  as the input to the bound set function, and notice that the output of the overall function  $F$  also results in 0.

**Don't cares:** Notice, in this case there are NO DON'T CARES! All four value combinations of  $g_0 g_1$  will be produced.

- 3) The support set of  $f$  is  $w, x, y, z$ . We are asked to partition the variables in such a way that the bound set size is two. The possibilities are:  $(w, x; y, z), (w, y; x, z), (w, z; x, y), (x, y; w, z), (x, z; w, y), (y, z; w, x)$ . When you draw the corresponding Karnaugh-maps and look for column multiplicity  $\leq 2$  (for feasible simple decompositions), only three partitions lead to a possible decomposition. These are:  $(w, x; y, z), (w, y; x, z)$  and  $(x, y; w, z)$ .

Let us consider the one with  $w, x$  as the bound set variables.

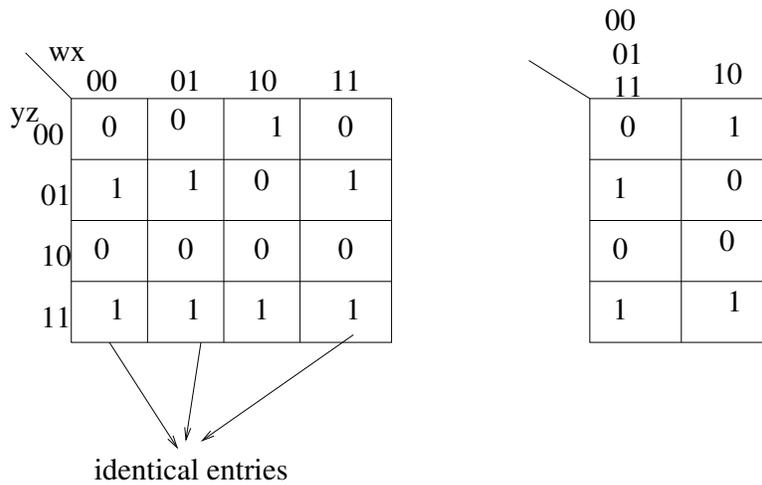


Fig. 2. Functional Decomposition Chart.

Notice that the function can be decomposed simple-disjunctively, and encoded with  $\phi(w, x) = w'x' + w'x + wx$  and  $\bar{\phi} = wx'$  or *vice-versa*. The entries in the  $\phi$  column that have ones correspond to the free-set terms:  $\phi \cdot y'z + \phi \cdot yz$ . Similarly, free-set terms corresponding to  $\bar{\phi}$  are  $y'z' + yz$ . Hence,  $F$  is decomposed as:  $\phi \cdot y'z + \phi \cdot yz + \bar{\phi}y'z' + \bar{\phi}yz$ .

If you transpose the above matrix - you'll have  $w, x$  as the free set and  $y, z$  as the bound set. But this won't work as column multiplicity is 4 (with one column of all 0s). This would require Ashenhurst-Curtis Decomp..... but thats another story.

- 4)  $x, y$  as bound set (and  $w, z$  as free set) does give column multiplicity = 2; which means that the # of cut-set nodes in the BDD should be equal to 2. And you can confirm that by looking at the following BDD.

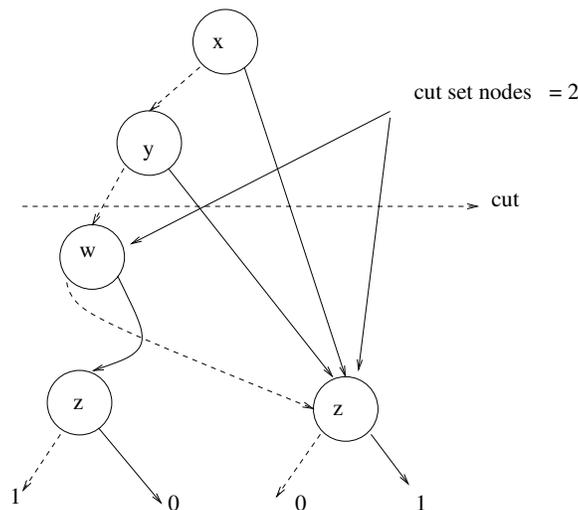


Fig. 3. Simple Decomposition on BDD