Using symmetry in FPGA layout

Benjamin Drucker, Craig Files, and Peter Siracusa
Department of Electrical Engineering
Portland State University
Portland, OR 97207-0751, USA
EE 572/672 Final Exam

1 Introduction

2 Selecting an ordering

Once the symmetrical pairs of variables are moved to the top of the lattice, the remaining variables must be placed in the lattice. Placing the remaining variables in the lattice is a very complex problem, in that a change of ordering will give a completely different lattice structure. But, the realization that is wanted, of course, is the smallest representation. This is a matter of checking possible orderings to find the smallest representation.

The whole approach is designed to fit, as close as possible, the function into that of a fully symmetric function. Figure 1, shows the layout of a 4 variable, fully symmetric function. This layout shows a karnough map, with integer values in each of the boxes. If a 4 variable function was fully symmetric then if it was mapped on the the layout of Figure 1, then all boolean outputs in the squares labeled 1 are equal, labeled 2 are equal, labeled 3 are equal and labeled 4 are equal.

	00	01	11	10
00	0	1	2	1
01	1	2	3	2
11	2	3	4	3
10	1	2	3	2

Figure 1: Karnough map for 4 variable symmetry

In Figure 2, the Karnough map from Figure 1 is mapped on to a lattice of Multiplexors. Notice that the bottom level of Multiplexors, depending on the input to the multiplexor, can output the labels 1, 2 or 3 differently. Thus, the boolean output values under the labels 1, 2 and 3 are split based on the input to the multiplexor.

An example of this is shown in Figure 3, where the Karnough map is labeled with variables A, B, C, and D, and the variable A is chosen to be the input to the bottom level of multiplexors. This results in a renumbering sequence from the previous karnough map, since the labels 1, 2 and 3 are split based on the value of A.

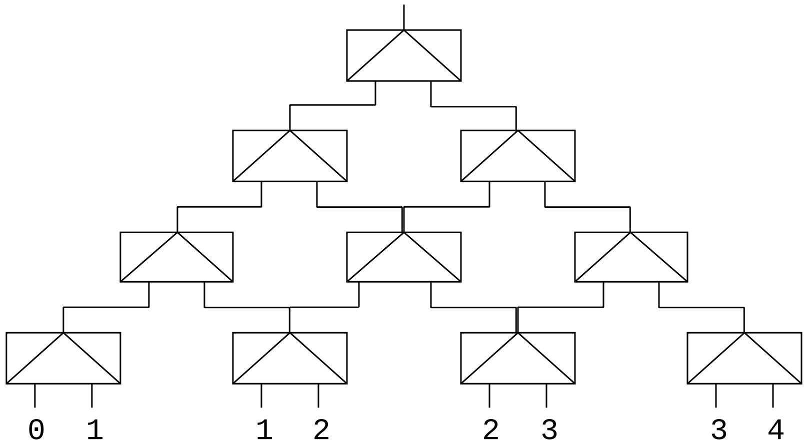
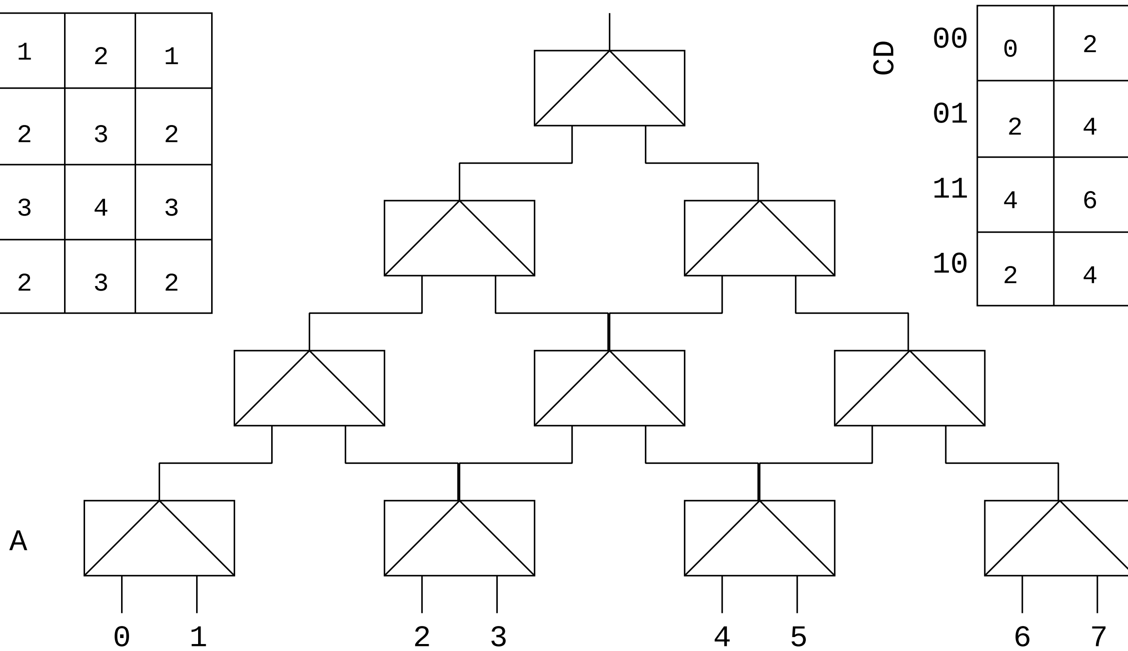


Figure 2: Lattice for 4 variable symmetry

CD \ AB	00	01	11	10
00	0	1	2	1
01	1	2	3	2
11	2	3	4	3
10	1	2	3	2



CD \ AB	00	01	11	10
00	0	2	3	1
01	2	4	5	3
11	4	6	7	5
10	2	4	5	3

Figure 3: A as the bottom level multiplexor input

To explore the idea of mapping a Boolean function in to a symmetric lattice, the the following example is given.

Given the boolean function given in Figure 4, the variable A is chosen as the bottom node in the lattice. Notice

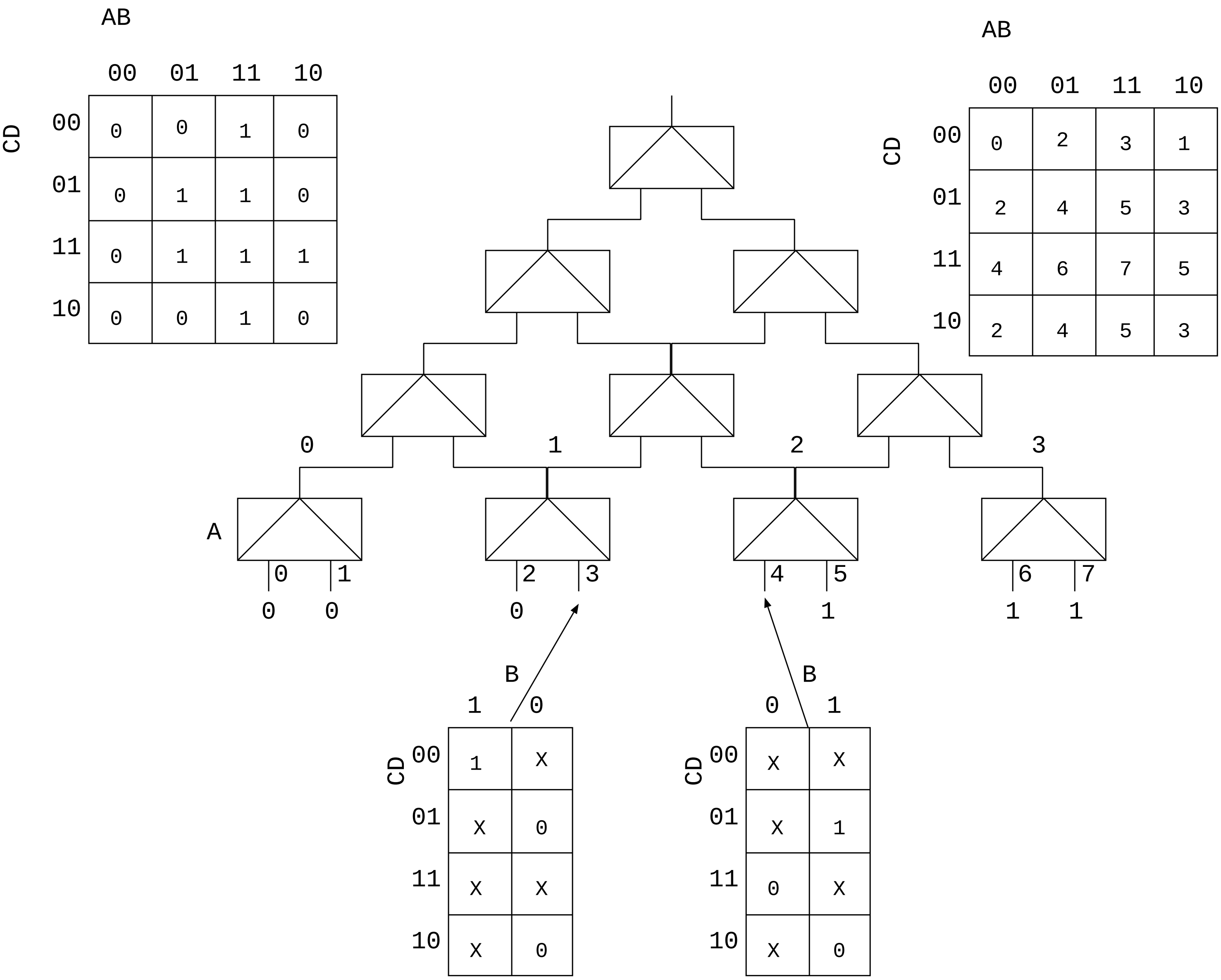


Figure 4: Example of mapping onto the lattice

that when splitting the label 2 (label 3 and 4 when A is the bottom of the lattice), the result is not a binary value, but new functions, with many Don't Cares. Note, than the first of these new functions can be realized as just B (after setting the Don't Cares). The second function, can be realized as just C. So the resulting lattice is shown in Figure 5.

So, the idea for ordering these variables is to check all possible variables as the bottom nodes on the lattice, then recursively checking the next lattices that occur underneath this lattice for the smallest structure that exists. This will be done by removing vacuous variables, checking for possible symmetries for a given pair of variables, and running the ordering algorithm.

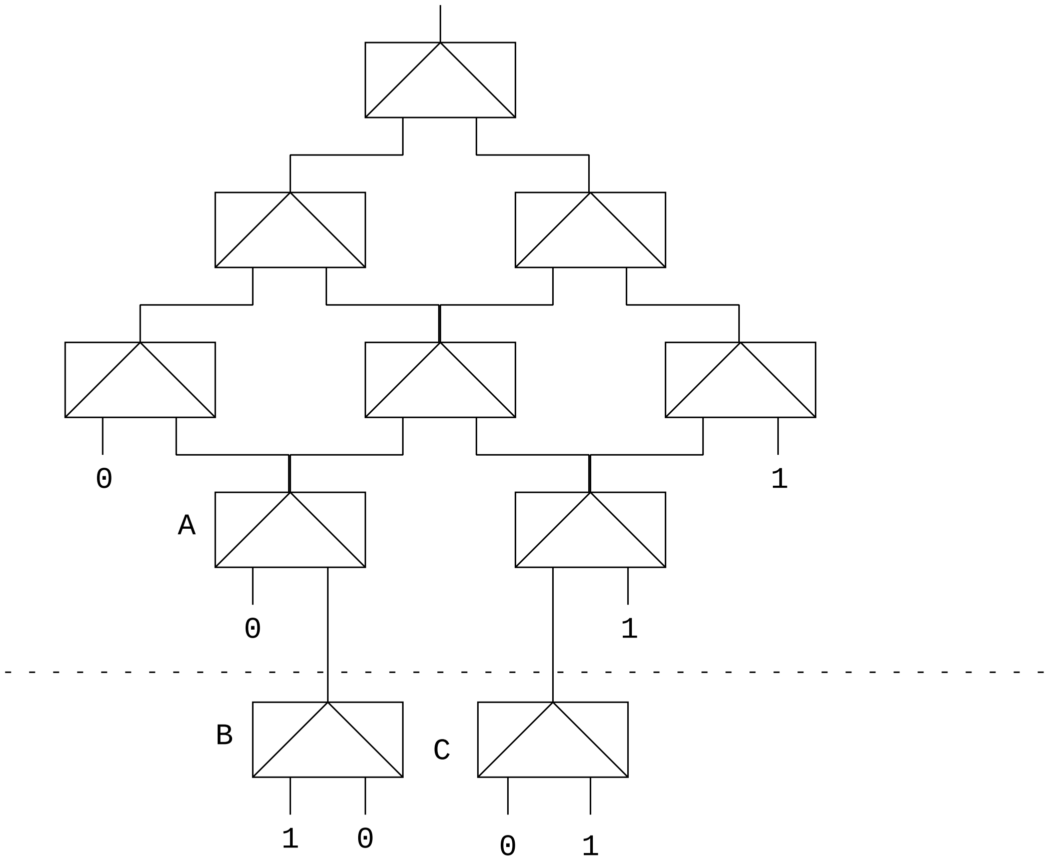
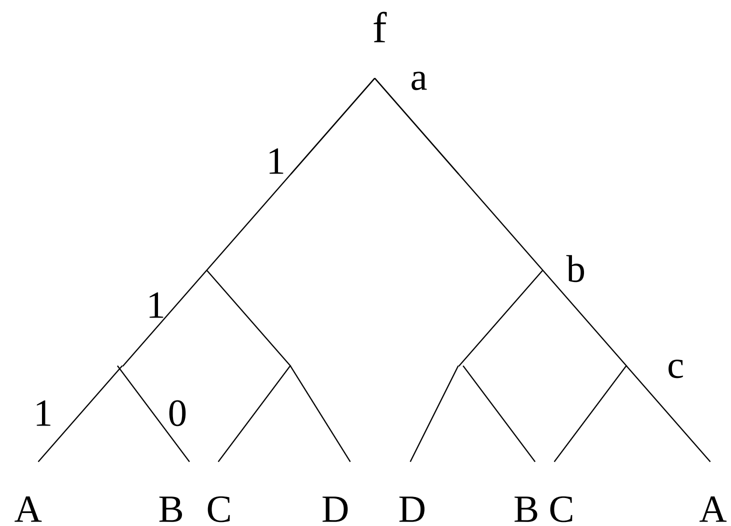


Figure 5: Reducing the next function

3 Linear Independence

Variable ordering ignores the important question: Can we create better variables?

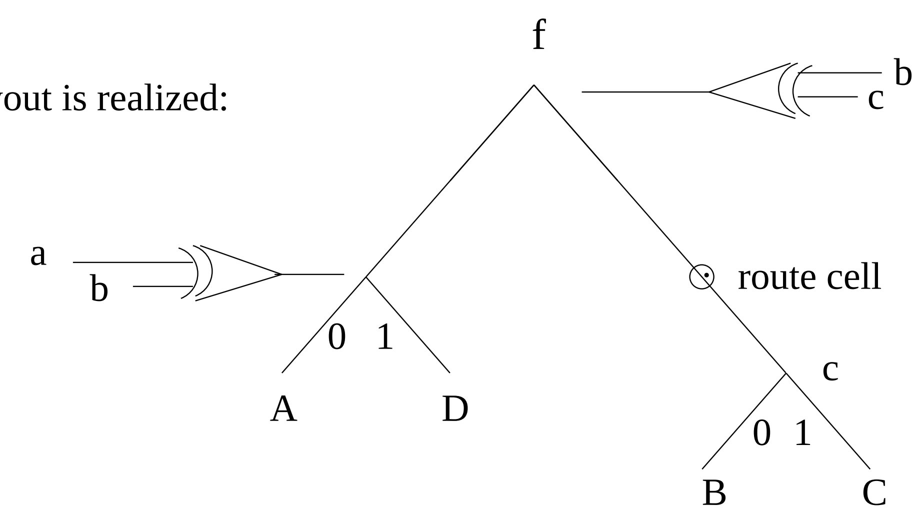
For the function f represented by the BDD,



,

symmetry methods do poorly.

With different decomposition variables a lattice layout is realized:



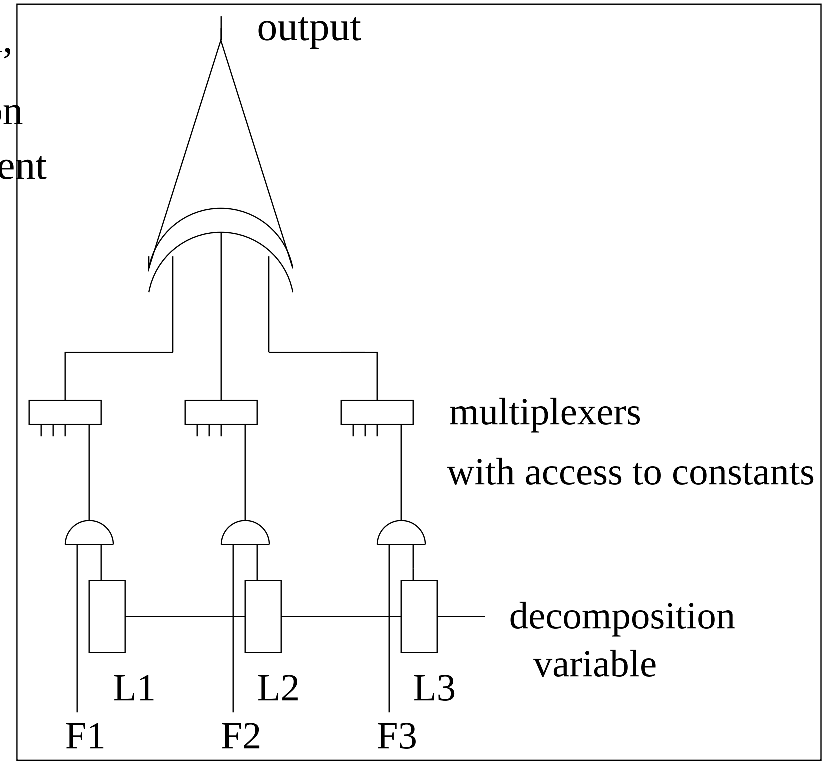
"Created" variables allow us to reduce redundancy

even when symmetry based methods fail.

Figure 6: Variable reordering

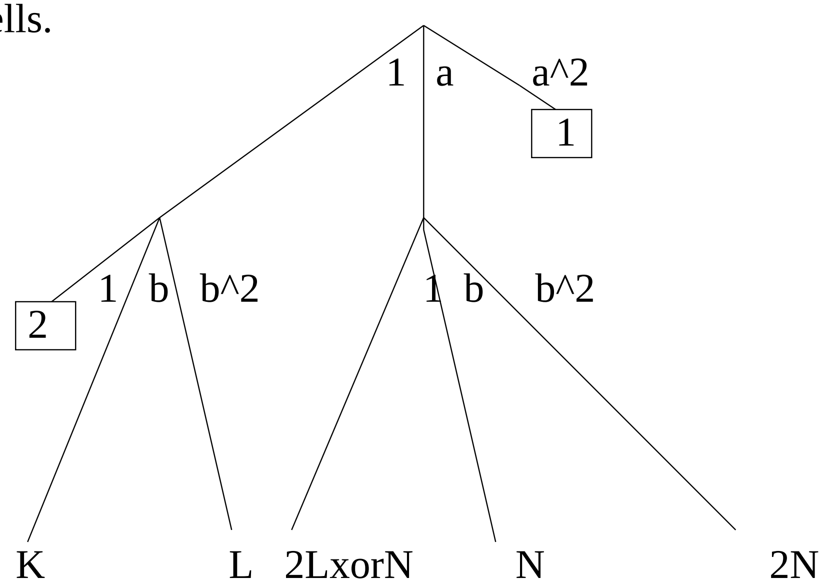
A Lattice Realization by convenient choice of Linearly Independent Basis

An example in GF(3): Consider a lattice constructed with the cell,



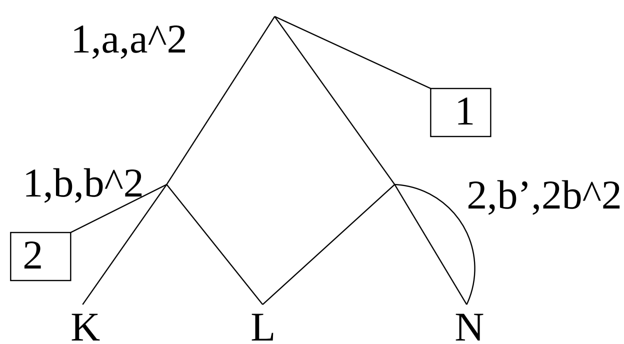
where F1, F2, F3 are subfunctions subject to further decomposition in subsequent cells. The L1, L2, L3 represent a linearly independent basis. We will consider the case where the choice of basis can be decided independently in each cell.

Consider the problem of mapping the following DD to a lattice of cells.



has no symmetries.

By a change of basis in one of the nodes we are able to reduce the tree so that it maps to the lattice:



Operations on bases also transform the subfunctions and may create opportunites for reductions by tautologies and constants. Strict symmetry methods do not alter the subfunctions, and missing hidden redundancies.

Figure 7: Basis for Linear Independence