# VC-2
# August 2014 QE

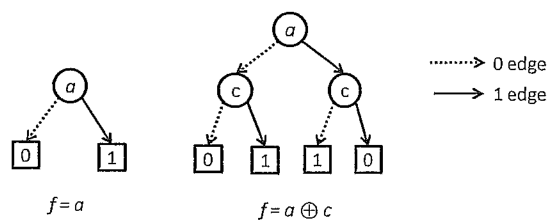
## 1 Boolean Functions (10+20 = 30 points)

Boolean functions can be represented in several ways, such as a truth table or a sum-of-products expression. This question explores two graph-based representations of Boolean functions, namely Binary Decision Trees and Binary Decision Diagrams.

### 1.1 Binary Decision Trees (10 points)

A *Binary Decision Tree* (BDT) is a representation of a logic function as a decision tree. A BDT has *internal nodes* and *leaves*. Each internal node is shown as a circle, and represents a decision on the input variable that it is labeled with. Each internal node has exactly two outgoing edges, which correspond to the input variable taking a value of 0 (left edge, drawn as a dotted line) or 1 (right edge, drawn as a solid line). Each leaf is shown as a rectangle, representing one of the two possible logic values: 0 or 1.

The following are two examples of BDTs ($f$ is the output; $a$ and $c$ are the inputs).



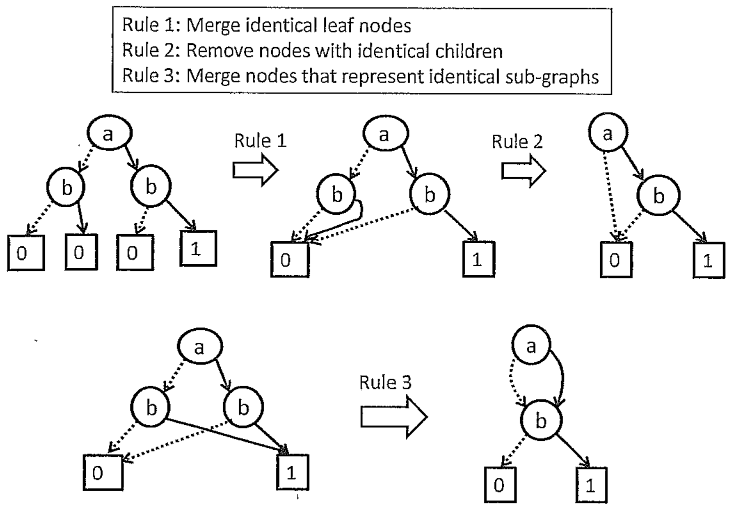

Draw *two different BDTs* for the function $g = x_1x_2 + x_3x_4$ that differ in the ordering of variables from the root to the leaves. For the first BDT, use the following variable order: $x_1 < x_2 < x_3 < x_4$ ($x_1$ should appear first along any path from root to leaf, and $x_4$ last). For the second BDT, use the following variable order: $x_1 < x_4 < x_2 < x_3$.

Write in Exam Book Only

### 1.2 Binary Decision Diagrams (20 points)

Binary Decision Diagrams (BDDs) are graph-based representations that are, in general, more compact than BDTs. Like BDTs, BDDs also have *internal nodes*, *edges*, and *leaves*.

It is possible to construct a BDD representation for a function from a BDT, by recursively applying a set of rules that are stated and illustrated in the following figure.




- Apply the rules to derive BDDs from each of the BDTs that you created in Question 1.1. Recall that both BDTs represent the same function, but with a different variable ordering.
- Compare the sizes of the BDDs in terms of the number of internal nodes. What conclusions can you derive from these numbers?

Write in Exam Book Only

## 2 Logic minimization (10+10+10=30 points)

During two-level logic minimization, it is useful to compute *prime implicants* and *essential prime implicants.* An implicant is a collection of on-set minterms of a function that can be represented as a conjunction (logic AND) of the input variables or their complements. A prime implicant is an implicant that is not contained within any other implicant of the function. An essential prime implicant contains a minterm that is not contained by any other prime implicant.

Consider the Boolean function represented by the following expression: $f(a, b, c, d, e) = ab + be + ace + bcd + cde$

**10 points.** Identify all prime implicants of the function.

**10 points.** For each of the prime implicants, state whether or not it is an essential prime implicant.

**10 points.** Explain how the use of prime implicants and essential prime implicants enables faster two-level minimization.

## 3 Timing Optimization (20+20=40 points)

A logic circuit's clock period is determined by the longest time or delay that the combinational logic can take to produce its outputs once its inputs are applied. The delay of a combinational circuit may be computed as the delay of its longest path(s).

Timing optimization therefore involves re-designing the circuit so as to reduce the delay of its longest path(s).

Consider the circuit given in Figure 1. Assume that (i) all 2-input gates have a delay of 1 unit, and (ii) a path's delay is the sum of the delays of the gates along the path. Answer the following questions:

**15 points.** What is the delay of this circuit? Identify all longest paths.

**25 points.** Re-structure the circuit to reduce its delay. The restructured circuit should only use 2-input gates.

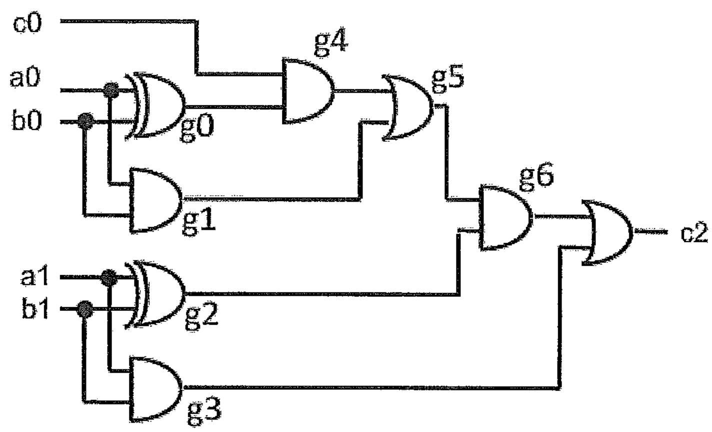



Figure 1: A combinational circuit

Write in Exam Book Only