Chapter 6: Multilevel Combinational Circuits

Name: Lương Văn Minh
No. : 09070452

Overview

- 6.1 Boolean Networks
- 6.2 Special Classes of Circuits
- 6.3 Binary Decision Diagrams

Overview

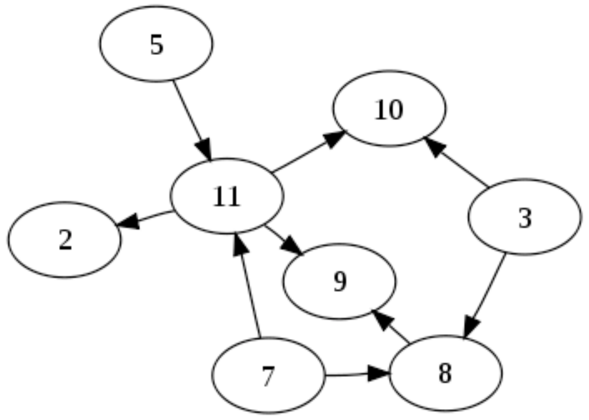
- 6.1 Boolean Networks
- 6.2 Special Classes of Circuits
- 6.3 Binary Decision Diagrams

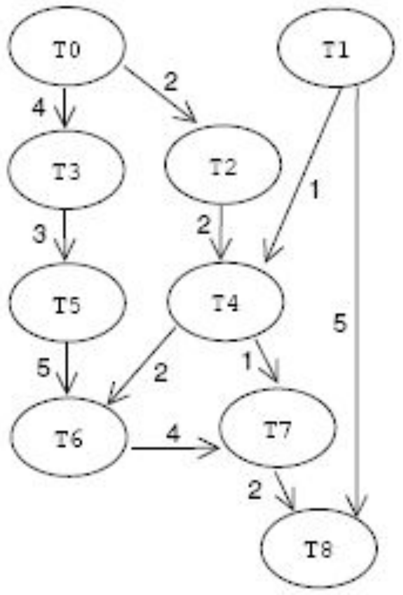
6.1 Boolean Networks

- A combinational logic circuits is represented as a labeled, directed, acyclic graph (DAG) G = (V, E).
- Each vertex v labeled with the name of a primitive gate such as AND, OR, or NOT, or with name of a primary input or output
- Each gate and edge in the circuit has an associated delay

6.1 Boolean Networks (cont.)

- Directed acyclic graph (DAG) G = (V, E) example:





6.1 Boolean Networks (cont.)

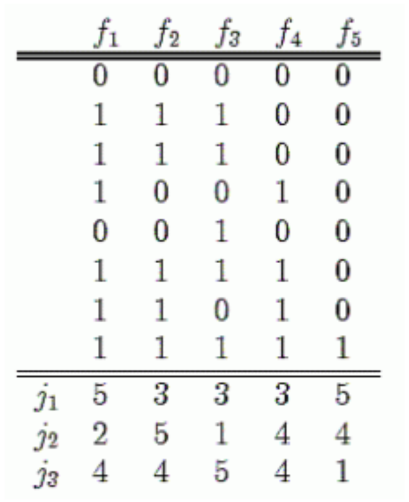
- The fan-out of a gate g (or a wire) is defined as the set of gates that use as an input the value generated by g.

- A Boolean network η is a DAG, each node i in η there is an associated cover Fi and a Boolean variable yi representing the output of Fi.

- May implement a Boolean network η by replacing each cover Fi in η by a NAND-NAND network.

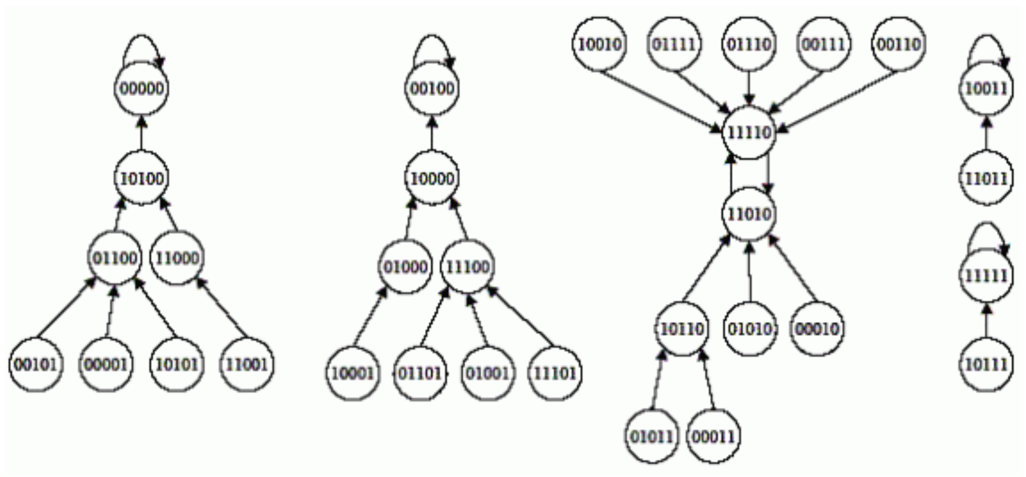
6.1 Boolean Networks (cont.)

- Boolean Networks example:



	f_1	f_2	f_3	f_4	f_5
	0	0	0	0	0
	1	1	1	0	0
	1	1	1	0	0
	1	0	0	1	0
	0	0	1	0	0
	1	1	1	1	0
	1	1	0	1	0
	1	1	1	1	1
j_1	5	3	3	3	5
j_2	2	5	1	4	4
j_3	4	4	5	4	1

➜



Overview

- 6.1 Boolean Networks
- 6.2 Special Classes of Circuits
- 6.3 Binary Decision Diagrams

6.2 Special Classes of Circuits

- 6.2.1 Fan-out-Free Circuits
- 6.2.2 Leaf-DAG Circuits
- 6.2.3 Algebraically Factored Circuits
- 6.2.4 Multiplexor-Based Circuits

6.2.1 Fan-out-Free Circuits

- A fan-out-free circuit is one in which the output of each gate and each circuit input to at most one gate.
- A fan-out-free circuit is also called a tree. For example, the circuit f = a b + c d is a tree.

6.2.1 Fan-out-Free Circuits (cont.)

- Many testability results have been proven. These are:

1. There exists a set of tests which detect all single and multiple stuck-at fault and is of minimal cardinality among all test sets for single faults.

2. The number of tests required to detect all stuck-at faults is bounded above by n:1 and is bounded below by 2√n where n is the number of circuit inputs.

3. A set of tests which detect all stuck-at faults on circuit inputs will detect all single stuck-at faults

6.2.2 Leaf-DAG Circuits

- A leaf-DAG circuit is a generalization of a fan-out-tree circuit where only the primary inputs are allowed to fan out to multiple gates.
- Any circuit can be converted into a leaf-DAG circuit by gate duplication.

6.2.3 Algebraically Factored Circuits

- An algebraically factored circuit is a circuit which is derived by a sequence of algebraic transformations from a two-level sum-of-products representation.
- For example, the circuit (a + b) . (c + d) is an algebraic factorization of the sum-of-products expression a.c + a.d + b.c + b.d

6.2.4 Multiplexor-Based Circuits

- Arbitrary Boolean function can be implemented using circuits whose only constituent gates are two-input multiplexors.

Overview

- 6.1 Boolean Networks
- 6.2 Special Classes of Circuits
- 6.3 Binary Decision Diagrams

6.3 Binary Decision Diagrams

- Binary decision diagrams (BDDs) were first proposed by Lee, further developed by Akers

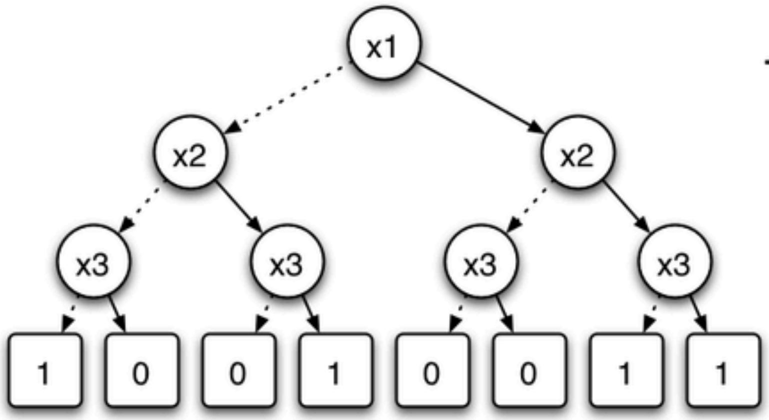
- BDD is a rooted, directed graph with vertex set V containing two types of vertices:

1. a nonterminal vertex v has as attributes an argument index index(v) ∈ { 1,..., n } and two children low(v), high(v).

2. a terminal vertex v has as an attribute a value value(v) ∈ { 0 , 1}

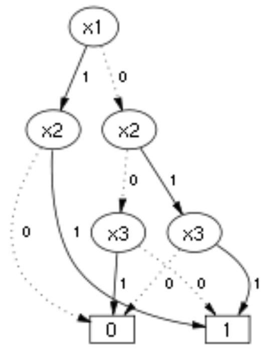
6.3 Binary Decision Diagrams (cont.)

BDDs example:



x1	x2	x3	*f*
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Binary decision tree and truth table for the function f(x1, x2, x3) = -x1 * -x2 * -x3 + x1 * x1 * x2 + x2 * x3



BDD for the function f

6.3 Binary Decision Diagrams (cont.)

- A BDD G having root vertex v defines a function fv defined recursively as:

1. If v is a terminal vertex:

    a) If value(v) = 1 then fv = 1

    b) If value(v) = 0 then fv = 0

2. If v is a nonterminal vertex with index(v) = i then fv is the function:

    $fv(x1, \ldots, xn) = \neg xi \,.\, f\,low(v)(x1, \ldots, xn) + xi \,.\, f\,high(v)(x1, \ldots, xn)$

    xi called the decision variable for vertex v

6.3 Binary Decision Diagrams (cont.)

- BDDs can be categorized based on two additional properties:

1. Freedom: When traversing any path from a terminal vertex to the root vertex we encounter each decision variable at most one

2. Ordered: place the restriction that for any nonterminal vertex v.

   - if low(v) is also nonterminal then must have index(v) < index(low(v))
   - if high(v) is also nonterminal then must have index(v) < index(high(v))