**Synthesizing OBDDs - A relational mu-calculus**

**Synthesizing OBDDs**

The method used for producing an OBDD for the transition relation was to compute first the truth table and then an OBDD which might not be in its fully reduced form; hence the need for a final call to the reduce function. However, this procedure would be unacceptable if applied to realistically sized systems with a large number of variables, for the truth table’s size is exponential in the number of boolean variables. The key idea and attraction of applying OBDDs to finite systems is therefore to take a system description in a language such as SMV and to synthesise the OBDD directly, without having to go via intermediate representations (such as binary decision trees or truth tables) which are exponential in size.

SMV allows us to define the next value of a variable in terms of the current values of variables. This can be compiled into a set of boolean functions fi, one for each variable xi, which define the next value of xi in terms of the current values of all the variables.

Modelling sequential circuits As a further application of OBDDs to verification, we show how OBDDs representing circuits may be synthesised.

Synchronous circuits. Suppose that we have a design of a sequential circuit. This is a synchronous circuit (meaning that all the state variables are updated synchronously in parallel) whose functionality can be described by saying what the values of the registers x1 and x2 in the next state of the circuit are.



**A simple synchronous circuit with two registers.**

**A relational mu-calculus**

that evaluating the set of states satisfying a CTL formula in a model may involve the computation of a fixed point of an operator. For example, [[EF φ]] is the least fixed point of the operator F : P(S) → P(S) given by F(X) = [[φ]] ∪ pre∃(X).

we introduce a syntax for referring to fixed points in the context of boolean formulas. Fixed-point invariants frequently occur in all sorts of applications. so it makes sense to have an intermediate language for expressing such invariants syntactically. This language also provides a formalism for describing interactions and dependences of such invariants. We will see shortly that symbolic model checking in the presence of simple fairness constraints exhibits such more complex relationships between invariants.