

ECEN 5139
Formal Verification
of VLSI Systems

Bounded Model Checking

November, 2003

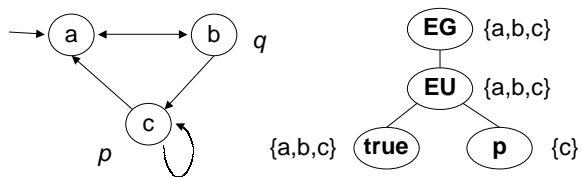
Lecturer: Chao Wang

Electrical & Computer Engineering
 University of Colorado, Boulder

BMC – I

10/20, 2003

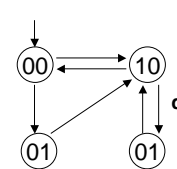
State Explosion ! ?



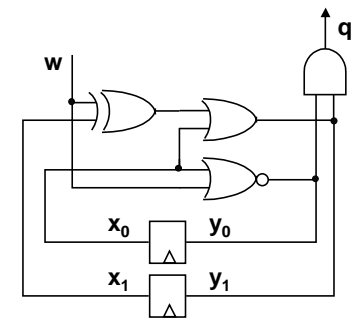
Num of state vars = N (size of the model)
 Num of states = 2^N (size of the STG)
 ^^

Circuit is compact \rightarrow Let's try a different approach ...

A Simple Circuit



$S_0 = 00$
 $\phi = AG \neg q$

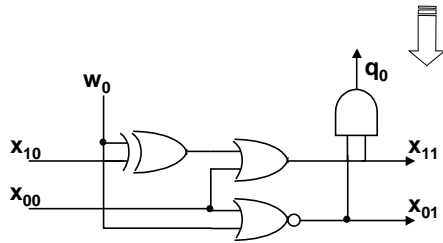


At Time Frame 0 (t=0)

$S_0 = 00$
means

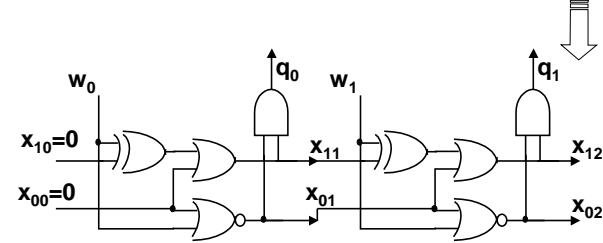
$x_{10} = 0$

$x_{00} = 0$



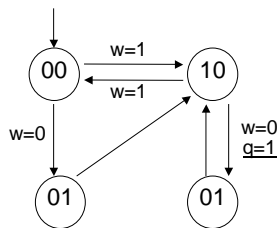
AG¬q : Can q_0 be 1 ?
No !

At Time Frame 1 (t=1)

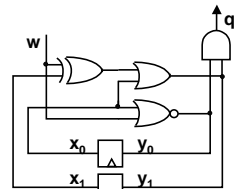


AG¬q : Can q_1 be 1 ?
Yes, If $w_0=1, w_1=0$

Assignment → Counter Example

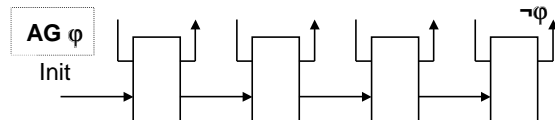
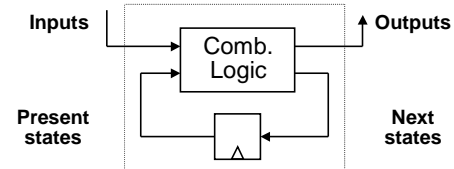


AG¬q : Can q_1 be 1 ?
Yes, If $w_0=1, w_1=0$

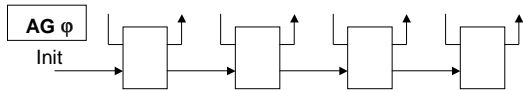


$S_0 = 00$
 $\varphi = \text{AG } \neg q$

Summarize What We Did Just Now



When to Stop ?

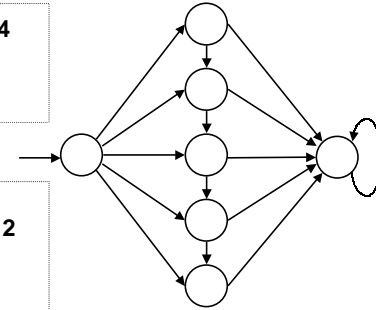


- Num. of states *(too large)*
- Num. of reachable states *(unknown)*
- Diameter of the STG *(unknown)*
- Reachable diameter of the STG *(unknown)*

Reachable Diameter ?

Diameter = 4
The length of the longest "shortest path"

Reachable Diameter = 2
The length of the longest "shortest path from initial states"



Debugging vs. Verification

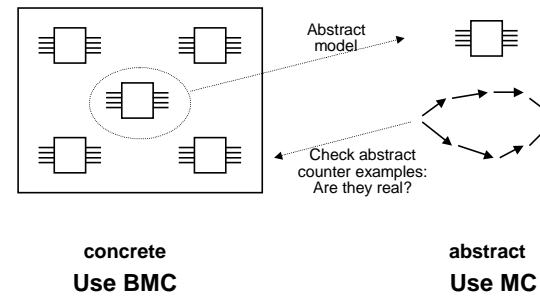
Model Checking is complete

Bounded Model Checking *can be* incomplete. However,

- Scale to larger circuits
- A good debugging tool
- On going research ...

Combine MC with BMC ?

Combine MC with BMC in Abstraction Refinement



BMC - II

10/22, 2003

Linear Time Logic (LTL)

LTL φ means $A \varphi$

“ φ must be satisfied by all the paths”

E.g. $FG p$ means $A (FG p)$

$A \varphi = \neg (E \neg \varphi)$

Let $(\psi = \neg \varphi)$, $A \varphi = \neg (E \psi)$

$(E \psi)$ is an “Existential LTL” formula

BMC for Existential LTL

Definition: $E \varphi$, where φ is in LTL

$E (F p)$	$E (p U q)$
$E (G p)$	$E (p R q)$
$E (X p)$...

$E (GF p)$

Meaning:

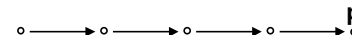
$(E \varphi)$ is TRUE
iff there exists a path that satisfies φ

A Finite Witness (loop-free)

$E (X p)$



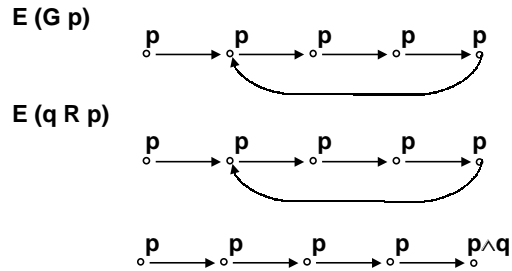
$E (F p)$



$E (p U q)$



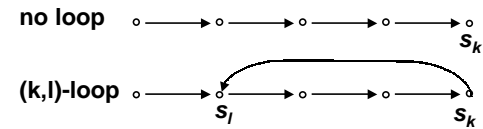
A Finite Witness (with loop)



Bounded Paths on M

π : an infinite path $(s_0, s_1, \dots, s_k, \dots)$

π_k : a length-k prefix is (s_0, s_1, \dots, s_k)



π_k with (k,l) -loop represents an infinite path

Bounded Semantics of LTL

With (k,l) -loop

π_k models $(\varphi) \Leftrightarrow \pi$ models (φ)

No loop

π_k models $(\varphi) \rightarrow \pi$ models (φ)

Therefore,

$\exists k . M_k$ models $E(\varphi) \Leftrightarrow M$ models $E(\varphi)$

Bounded Semantics (no loop)

$\pi \models_k^i p$ iff $p \in L(\pi(i))$

$\pi \models_k^i \neg p$ iff $p \notin L(\pi(i))$

$\pi \models_k^i f \wedge g$ iff $\pi \models_k^i f$ and $\pi \models_k^i g$

$\pi \models_k^i f \vee g$ iff $\pi \models_k^i f$ or $\pi \models_k^i g$

$\pi \models_k^i Gf$ is always false

$\pi \models_k^i Ff$ iff $\exists j, i \leq j \leq k. \pi \models_k^j f$

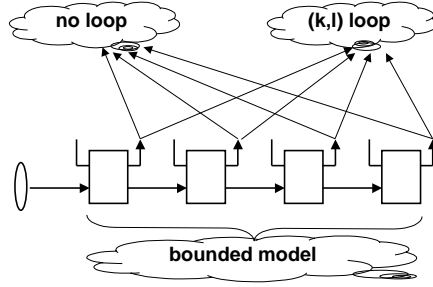
$\pi \models_k^i Xf$ iff $i < k$ and $\pi \models_{k+1}^{i+1} f$

$\pi \models_k^i f U g$ iff $\exists j, i \leq j \leq k. \pi \models_k^j g$ and $\forall n, i \leq n < j. \pi \models_k^n f$

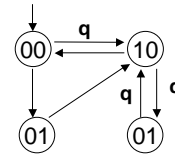
$\pi \models_k^i f R g$ iff $\exists j, i \leq j \leq k. \pi \models_k^j f$ and $\forall n, i \leq n \leq j. \pi \models_k^n g$

(duality between G and F no longer holds)

Translate LTL MC into SAT



Example: LTL → SAT Translation



$$[M]_2 = I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2)$$

$${}_0L_2 = T(s_2, s_0)$$

$${}_1L_2 = T(s_2, s_1)$$

$${}_2L_2 = T(s_2, s_2)$$

Check E (G q)

$$[Gq]_2 = q(s_0) \wedge q(s_1) \wedge q(s_2)$$

- (00,01,10) (2,0)-loop
- ➔ (00,10,01) (2,1)-loop
- (00,10,00) (2,1)-loop

$$[M]_2 \wedge ({}_0L_2 \vee {}_1L_2 \vee {}_2L_2) \wedge [Gq]_2$$

LTL Translation (with (k,l)-loop)

$${}_i[[p]]_k := p(s_i) \quad {}_i[[\neg p]]_k := \neg p(s_i)$$

$${}_i[[f \wedge g]]_k := {}_i[[f]]_k \wedge {}_i[[g]]_k$$

$${}_i[[Xf]]_k := {}_i[[f]]_{\text{succ}(i)_k}$$

$${}_i[[Ff]]_k := {}_i[[f]]_k \vee {}_i[[Ff]]_{\text{succ}(i)_k}$$

$${}_i[[Gf]]_k := {}_i[[f]]_k \wedge {}_i[[Gf]]_{\text{succ}(i)_k}$$

$${}_i[[fUg]]_k := {}_i[[g]]_k \vee ({}_i[[f]]_k \wedge {}_i[[fUg]]_{\text{succ}(i)_k})$$

$${}_i[[fRg]]_k := {}_i[[g]]_k \wedge ({}_i[[f]]_k \vee {}_i[[fRg]]_{\text{succ}(i)_k})$$

LTL Translation (no loop)

$$[[p]]_k := p(s_i)$$

$$[[\neg p]]_k := \neg p(s_i)$$

$$[[f \wedge g]]_k := [[f]]_k \wedge [[g]]_k$$

$$[[f]]_k^{K+1} := 0$$

$$[[Xf]]_k := [[f]]_{i+1_k}$$

$$[[Ff]]_k := [[f]]_k \vee [[Ff]]_{i+1_k}$$

$$[[Gf]]_k := [[f]]_k \wedge [[Gf]]_{i+1_k}$$

$$[[fUg]]_k := [[g]]_k \vee ([[f]]_k \wedge [[fUg]]_{i+1_k})$$

$$[[fRg]]_k := [[g]]_k \wedge ([[f]]_k \vee [[fRg]]_{i+1_k})$$

General Translation

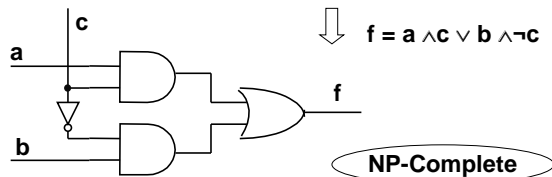
φ : LTL formula
M : Kripke structure
k : $k \geq 0$

$$[[M, \varphi]]_k := [[M]]_k \wedge \left((\neg L_k \wedge [[\varphi]]_k^0) \vee \bigvee_{l=0}^{k-1} (L_k \wedge_l [[\varphi]]_k^l) \right)$$

SAT-I

10/24, 2003

Circuit SATisfiability



Choices to solve it

1. Build the function $f(a,b,c)$, check if $(f=0)$
2. Recursive search:
 set $f=1$, then $t1=1$, then $a=1$ and $c=1$, done!

CNF SAT

CNF (Conjunctive Normal Form)

$$(a \vee \neg c) \wedge (b \vee \neg c) \wedge (\neg a \vee \neg b \vee c) \wedge \dots$$

\uparrow
literal
 \uparrow
clause

How to solve it ?

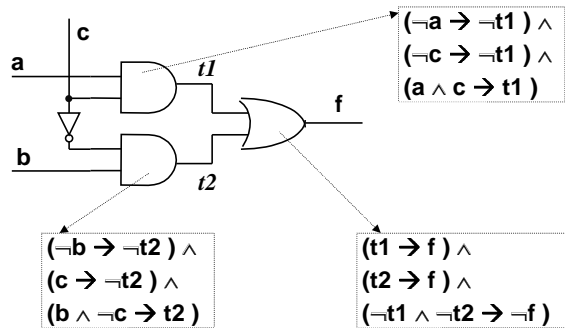
Recursive search !

set $(a=0)$ then check if CNF($a=0$) is sat

set $(a=1)$ then check if CNF($a=1$) is sat

Circuit SAT → CNF SAT

Translation is linear



Davis-Putnam Procedure (1960,1962)

```

Sat(d) {
  If ( Decide(d) == ALL-DECIDE ) return SAT;
  While (1) {
    If ( Deduce(d) != CONFLICT ) {
      If ( Sat(d+1) == SAT ) return SAT;
      else if ( β < d || d == 0 )
        { Erase(d); return UNSAT; }
    }
    If ( Diagnose(d) == BACK-TRACK )
      return UNSAT
  }
}
    
```

From GRASP

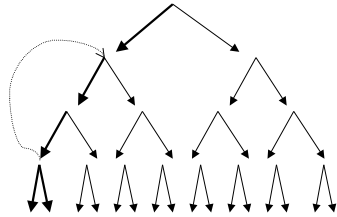
Boolean Constrain Propagation

$(\neg x1 \vee x2) \wedge (\neg x1 \vee x3 \vee x5) \wedge (\neg x2 \vee x4) \wedge (\neg x3 \vee \neg x4) \wedge (\neg x5)$
Unit clause: $x5=0$ d
 $(\neg x1 \vee x2) \wedge (\neg x1 \vee x3) \wedge (\neg x2 \vee x4) \wedge (\neg x3 \vee \neg x4)$
Decide: $x1=1$ d+1
 $(x2) \wedge (x3) \wedge (\neg x2 \vee x4) \wedge (\neg x3 \vee \neg x4)$
Unit clauses: $x2=1, x3=1$ d+1
 $(x4) \wedge (\neg x4)$
Conflict! Flip the last decision: $x1=0$ d+1
 $(\neg x2 \vee x4) \wedge (\neg x3 \vee \neg x4)$
Decide: $x4=1$ d+2
 $(\neg x3)$
Unit clause: $x3=0, Done!$ d+2

One-Phase Variables

$(\neg x1 \vee x2) \wedge (\neg x1 \vee x3 \vee x5) \wedge (\neg x2 \vee x4) \wedge (\neg x3 \vee \neg x4) \wedge (\neg x5)$
Unit clause: $x5=0$ d
One-phase variable: $x1=0$ d+1
 $(\neg x2 \vee x4) \wedge (\neg x3 \vee \neg x4)$
Decide: $x4=1$ d+2
 $(\neg x3)$
Unit clause: $x3=0, Done!$ d+2

Search Tree



Backtracking

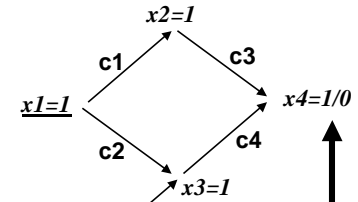
Chronological, or non-chronological

Conflict Analysis

- $c1 = (\neg x1 \vee x2)$
- $c2 = (\neg x1 \vee x3 \vee x5)$
- $c3 = (\neg x2 \vee x4)$
- $c4 = (\neg x3 \vee \neg x4)$



$(\neg x1 \vee x5)$



Implication graph

Example from
Biere et al.

No New Information: Conflict Clause & Resolution

- $c1 = (\neg x1 \vee x2)$
- $c3 = (\neg x2 \vee x4)$
- $c2 = (\neg x1 \vee x3 \vee x5)$
- $c4 = (\neg x3 \vee \neg x4)$

- c1 & c3:
 $(\neg x1 \vee x4)$
- c2 & c4:
 $(\neg x1 \vee \neg x4 \vee x5)$



$(\neg x1 \vee x5)$

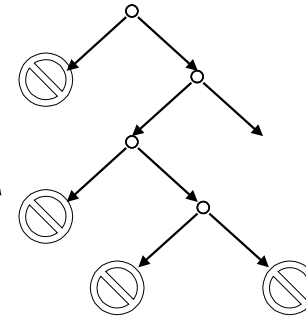


$(\neg x1 \vee x5)$



Prune the Search Space

By adding
conflict
clauses



Efficient BCP: The Intuition

$$(\neg x_1 \vee x_2) \wedge (\neg x_1 \vee x_3 \vee x_5) \wedge (\neg x_2 \vee x_4) \wedge (\neg x_3 \vee \neg x_4) \wedge (x_5)$$

After set $(x_1=1)$:

Have to visit every literal in every clause ?

Yes

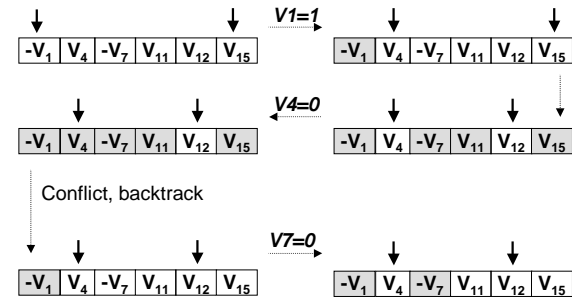
Can we skip even $(\neg x_1 \vee x_3 \vee x_5)$?

Yes, it will NOT become "unit"

Ideally, only need to visit new "unit" clauses

How to do it ?

Efficient BCP: Two Watched Literals



From Chaff

BMC - Induction

10/24/2003

General Induction Proof

Simple Induction

Base: $\varphi(0)$ is true

Inductive: $\varphi(n)$ is true $\rightarrow \varphi(n+1)$ is true

K-step Induction

Base: $\varphi(0), \varphi(1), \dots, \varphi(k)$ are true

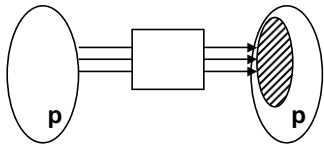
Inductive: $n-k < i \leq n, \varphi(i)$ is true
 $\rightarrow \varphi(n+1)$ is true

Simple Induction in BMC

Simple Induction for $A(G p)$ under all $k \geq 0$

Base: $I \subseteq p$

Inductive: $EY(p) \subseteq p$



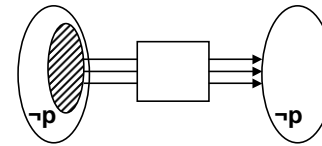
Simple Induction (backward)

Simple Induction for $A(G p)$

Base: $I \cap \neg p = \emptyset$

$(I \subseteq p)$

Inductive: $EX(\neg p) \subseteq \neg p$

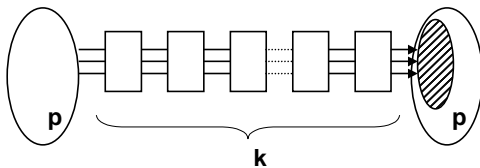


K-Step Induction

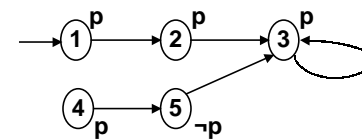
K-step induction

Base: $I \subseteq p$, and $0 < i \leq k$, $EY^i(I) \subseteq p$

Inductive: $EY^k(p) \subseteq p$



Can K-Step Be Better ?



Simple induction

$EY(p) = \{2,3,5\} \not\subseteq p$

2-step induction

$EY EY(p) = EY\{2,3,5\} = \{3\} \subseteq p$

Longest Simple Path

Simple path: No two states are the same

Theorem: Let L be the length of longest simple path, D be the reachable diameter

$$D \leq L \quad (\text{completeness threshold !})$$

$$\bigwedge_{j=0}^n \bigwedge_{k=j+1}^{n+1} (s_j \neq s_k) \wedge \left(\mathbf{I}(s_0) \wedge \bigwedge_{i=0}^n (\mathbf{p}(s_i) \wedge \mathbf{T}(s_i, s_{i+1})) \right)$$

BMC with Longest Simple Paths

```
k = 0;
While (1) {
  If ( BMC(k) )
    return FALSE;
  If ( ¬FwdSimplePath(k) || ¬BwdSimplePath(k) )
    return TRUE
  k++;
}
```

[Sheeran et al.]

A (G p)

Still Not Tight Enough, But ...

Reachable
Diameter
 $D = 2$

Length of the
longest simple
path
 $L = 6$

