# Transition-Based Coverage Estimation for Symbolic Model Checking

Xingwen Xu

Shinji Kimura

Kazunari Horikawa

Takehiko Tsuchiya

# Agenda

- Introduction
- Transition Coverage Metric
- Transition Coverage Computation
- Experimental Results
- Summary

# Introduction

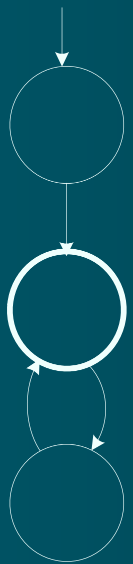
- Coverage for model checking
  - The validation is exhaustive only for functions specified by properties
    - Properties are written manually.
    - Hard to determine completeness of properties
  - A design error might not be detected by model checking
    - if the erroneous behavior is not checked by specified properties.
  - Coverage for model checking is responsible for revealing such unchecked behaviors.
    - Specifying additional properties for higher verification quality

# Introduction

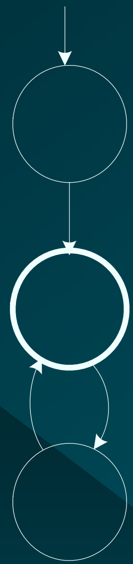
- Related works
  - State coverage Hoskote et al. DAC’99 Jayakumar. et al. DAC03 Chockler et al. CAV’01
    - Select a signal as observed signal
    - Change the value of the signal on one state
    - Whether any property get failed
  - High-level fault model Fummi et al MEMOCODE’03
    - Mutation based on RT-level fault model
  - Transition coverage Chockler et al. CHARM’03
    - Omit or replace transitions (or paths)
  - for symbolic simulation Wang et al. FORTE’03
    - Numerical safety analysis for real-time system
  - Transition Traversal coverage Xu et al. ASICON’05
    - Transitions are traversed by CTL operators

# Introduction

- Related works
  - State coverage metric



the original state machine satisfies the property AX(q=1)



select q as observed signal, change its value on state s1, AX(q=1) is no longer satisfied

➔ State s1 is covered for q.

  - The coverage metric is practically useful.
  - One of the limitations
    - State based, not transition or path based.
    - How about transition coverage metric?

# Introduction

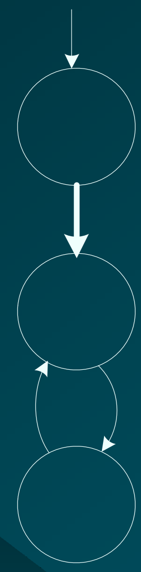
- Problem Definition

Input:

- given a state machine
- a set of satisfied property
- Select one observed signal

Output

- a set of transitions,
- On which the value of the signal is (not) checked

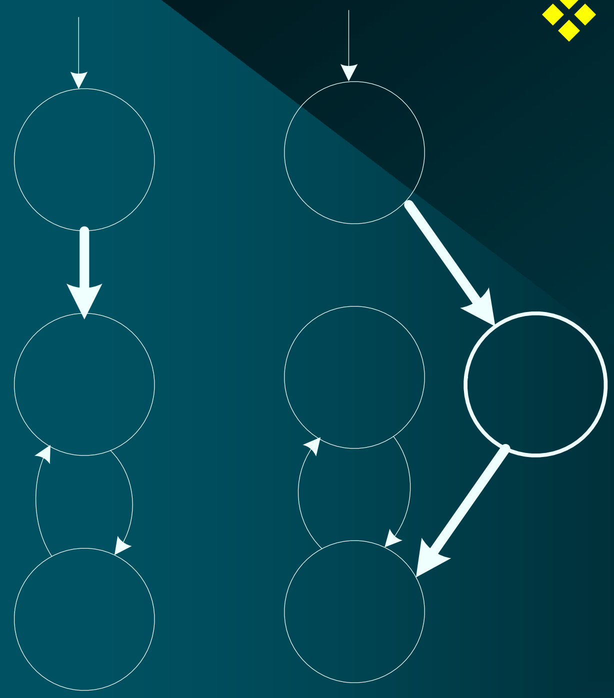


the state machine satisfies the property AX(q=1), which transition is covered for q? (r1)

# Introduction

- Our contribution
  - A novel transition coverage metric
    - Extension of the state coverage metric
  - Symbolic algorithms for coverage computation
    - Based on states traversing
    - Use states to represent covered transitions
    - Focus on transitions of FSM for HW verification
  - Practicality
    - Meaningful coverage holes uncovered with low computation overhead

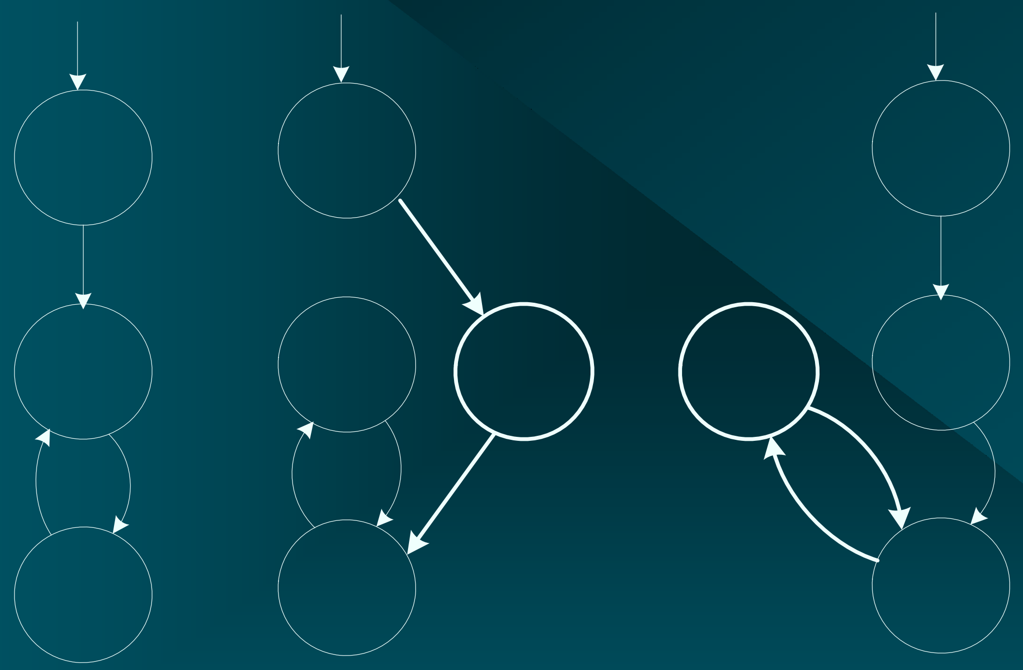
# Transition Coverage Metric



- Transition Perturbation
  - For a general state transition diagram
    - Select one observed signal q
    - For one transition r1=(S0,S1)
    - Make a copy of state S1 as Sq
    - Change the value of q in Sq
    - Re-direct r1 to Sq
    - Copy the transitions starting from S1 to Sq.
  - Not only change state labels, but also change transition relation

# Transition Coverage Metric

- A transition is covered if any property satisfied by original STD gets failed on the perturbed STD.

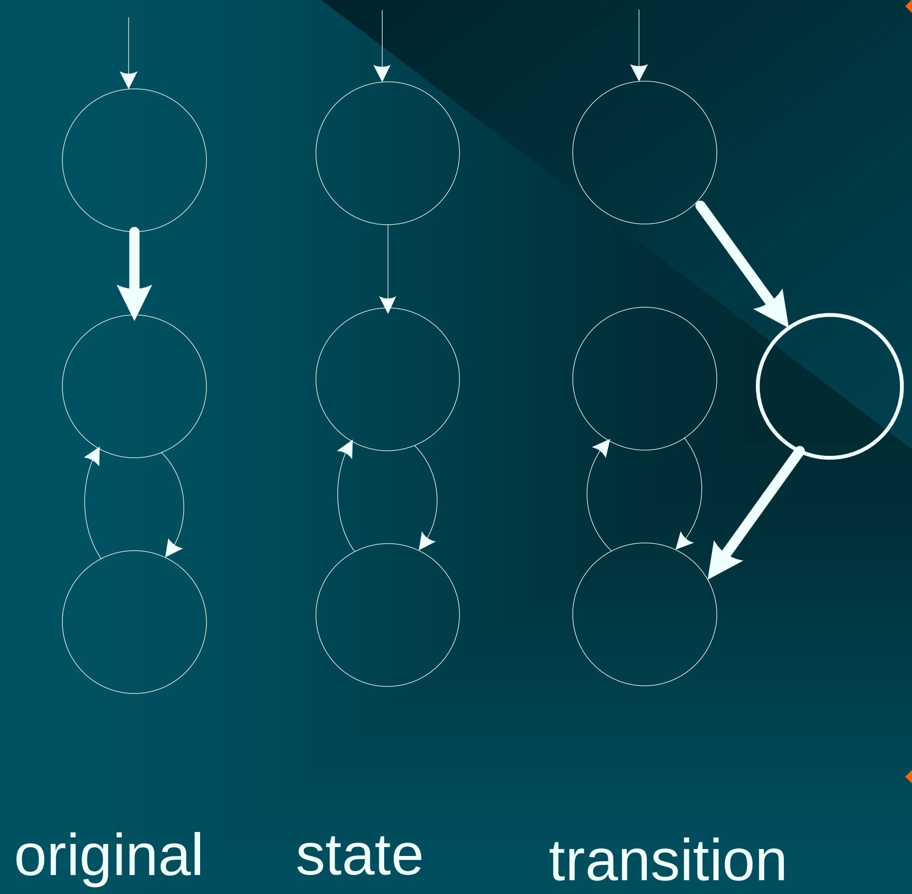


AXq=1

AXAXAXq=1

# Transition Coverage Metric

- Intuition of the coverage



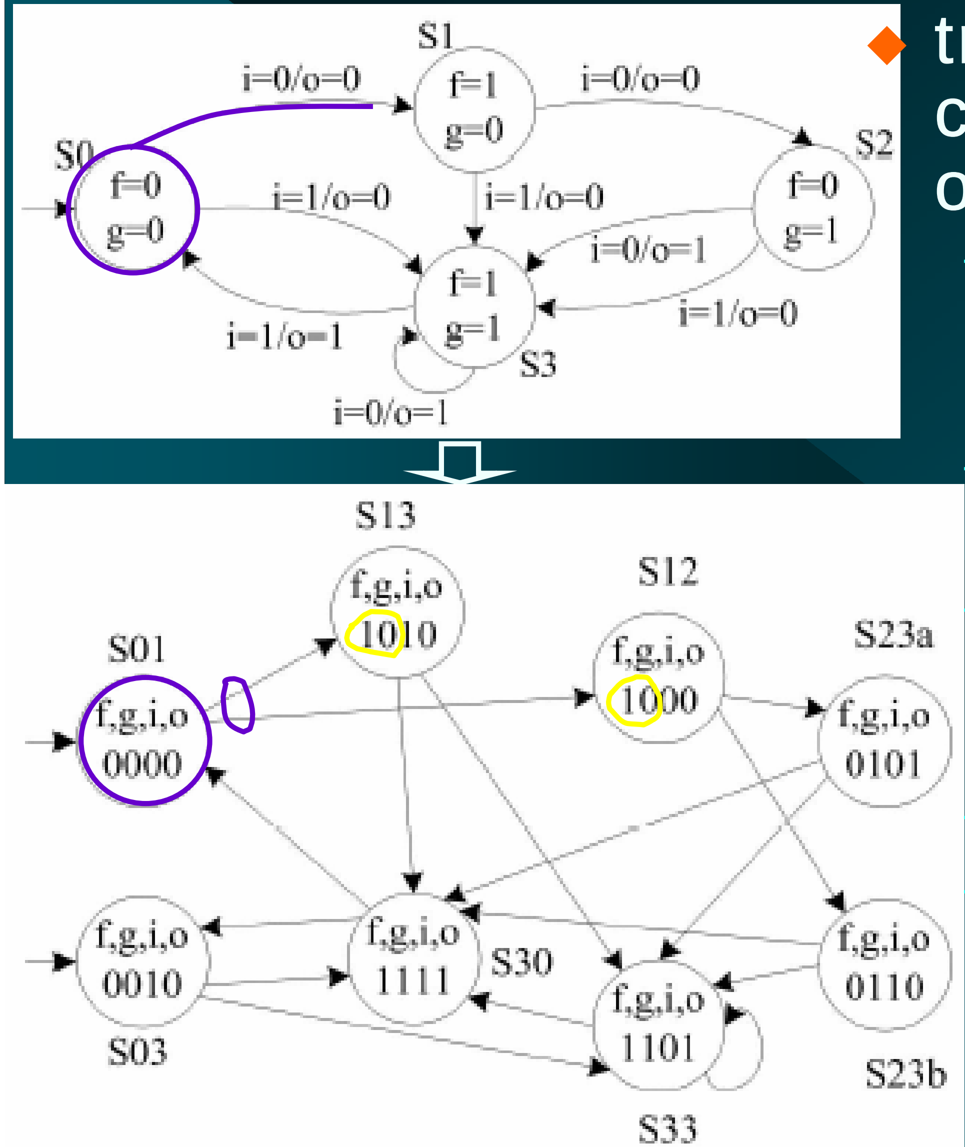

- Pinpoint the transition through which the value of observed signal is checked
  - change the labels of a state same as state metric
  - change the transition relation so that only one transition reaches the changed state, while all other transitions are reserved
- Provide coverage information on both signal value and transition relation

# Transition Coverage Metric

- Hardware Verification
  - Designs are modeled as FSM
  - We talk about the transition coverage of FSM
  - FSM is translated to Kripke structure for model checking properties referring to input signals
    - The FSM <S,I,O, $\delta$, $\lambda$ $S_0$> is translated to the Kripke structure <S$\times$I,R,L,$S_{k0}$>, where
      - $(\langle s,i\rangle,\langle s',i'\rangle) \in R$ iff $\delta(s,i)=s'$
      - $L(\langle s,i\rangle)=i \cup s \cup \lambda(s,i)$

# Transition Coverage Metric

- Hardware Verification




- transition coverage can be computed based on the states of the kripke structure
  - For each state of K, the next states have same values for FSM state variables.
  - Select FSM state variable as observed signals
  - when a transition is covered, all transitions from the same state are covered.
  - just need to record the state
  - the state represents a transition of FSM

  $\langle s,i \rangle \rightarrow \delta(s,i)=s'$

# Transition Coverage Metric

- Formal Definition on Kripke structure for FSM
  - According to the transition from FSM to Kripke structure, we formally define our coverage as:
- For the Kripke structure, given an observed signal q and state $r_i$ (it represents a transition of FSM), for each state $r_j$ with $(r_i, r_j) \in R$, add a state $r_j^q \in S^q$

*For each state t*

$$L_r^q(t) = \begin{cases} L(t) & \text{if } t \notin S^q; \\ L(r_j) \setminus \{q\} & \text{if } t = r_j^q \text{ and } \{q\} \in L(r_j); \\ L(r_j) \cup \{q\} & \text{if } t = r_j^q \text{ and } \{q\} \notin L(r_j). \end{cases}$$

*For each state pair* $(t_i, t_j)$,

$$(t_i, t_j) \in R_r^q \Leftrightarrow \begin{cases} (r_j, t_j) \in R & \text{if } t_i = r_j^q; \\ true & \text{if } t_i = r_i \text{ and } t_j = r_j^q; \\ false & \text{if } t_i = r_i \text{ and } t_j = r_j; \\ (t_i, t_j) \in R & \text{otherwise.} \end{cases}$$

- $r_i$ is covered w.r.t. q if any property is no longer satisfied on the perturbed Kripke structure

$$coverage = \frac{number\ of\ covered\ transitions}{number\ of\ reachable\ transitions}$$

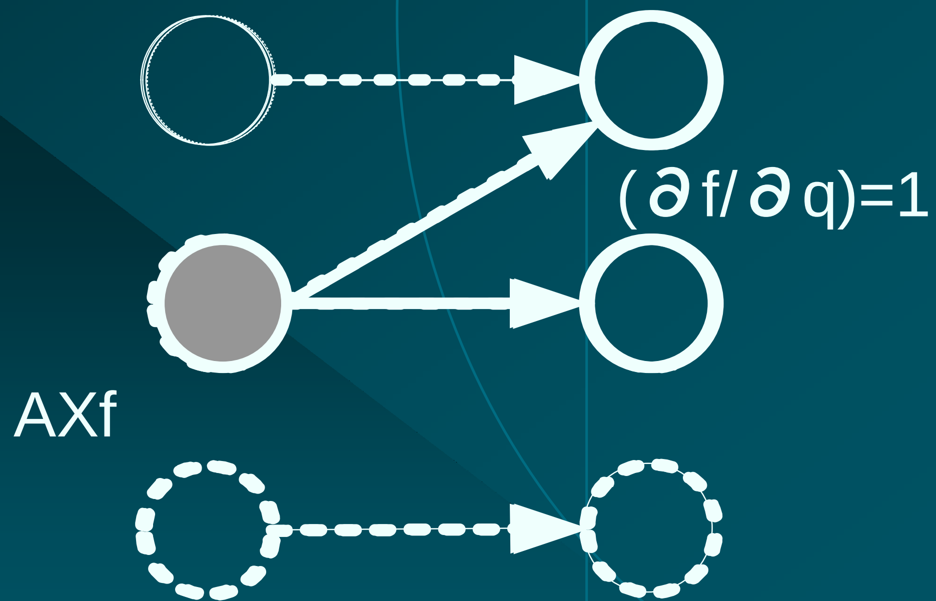
# Transition Coverage Computation

- The transition coverage metric is general for any specification language like CTL
- consider a subset of CTL for easy computation
  - Expressive for most properties
  - The subset is defined as:
    - *if b propositional, then b is within the subset;*
    - *if f and g are within the subset, then so are AXf, AGf, AfUg, AfRg,* $f \wedge g$, $b \rightarrow f$.
- The computation is performed on Kripke structure
- Symbolic algorithm based on BDD and fix-point operation

# Transition Coverage Computation

- while traversing a transition, we extract the correctness conditions from the property for the reached states, the transition is identified as covered if the correctness condition depends on the value of the observed signal.

- Three steps:
  - Traverse transitions according to CTL operators
  - Check the value of observed signal on states
  - Backward traversing for the covered transitions



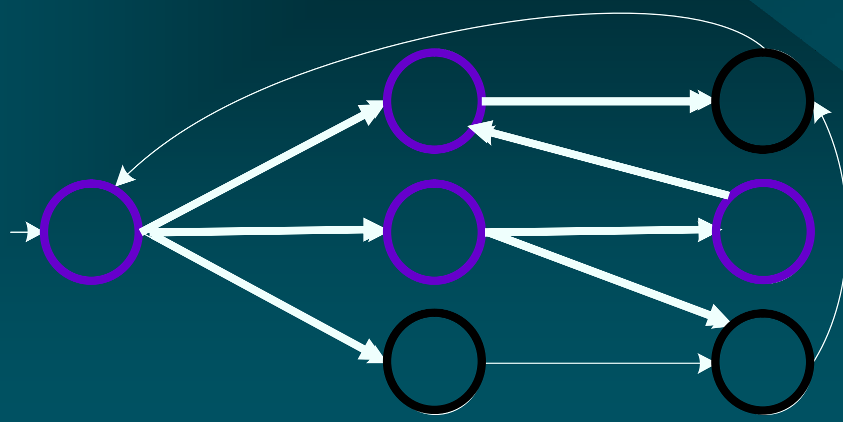
# Transition Coverage Computation

```
Cov(φ, S){
  if (S == empty) return empty;
  if (φ is propositional) return empty;
  switch (φ)
  case f ∧ g : result = Cov(f, S) ∪ Cov(g, S);
  case b → g : result = Cov(g, S ∩ Sat(b));
  case AGf : result = Cov(f, Rch(S));
  caseAXf :
    cf = Chk(f, Fwd(S));
    r1 = Bwd(cf) ∩ S; r2 = Cov(f, Fwd(S));
    result = r1 ∪ r2;
  case AfUg :
    fTrv = empty; gTrv = empty;
    gS = Sat(g); doS = S;
    do{
      gTrv = gTrv ∪ (doS ∩ gS);
      fS = doS \ gS;
      if(fS ≠ empty){
        fTrv = fTrv ∪ fS;
        doS = Fwd(fS) \ (fTrv ∪ gTrv);
      }
    }while(fS ≠ empty & doS ≠ empty);
    c1 = Chk(f, fTrv); c2 = Chk(g, gTrv);
    r1 = fTrv ∩ Bwd(c1); r2 = fTrv ∩ Bwd(c2);
    r3 = Cov(f, fTrv); r4 = Cov(g, gTrv);
    result = r1 ∪ r2 ∪ r3 ∪ r4;
  casefRg : //similar to fUg
  default : Unacceptable formula;
  return result;
}
```

```
Chk(φ, S){
  if (S == empty) return empty;
  if (φ is propositional)
    result = S ∩ Sat(¬φ|q→¬q);
    return result;
  switch (φ)
  case f ∧ g : result = Chk(f, S) ∪ Chk(g, S);
  case b → g : result = Chk(g, S ∩ Sat(b));
  case AGf : result = Chk(f, S);
  caseAXf : result = empty;
  case AfUg :
    r1 = Chk(g, S ∩ Sat(g));
    r2 = Chk(f, S \ Sat(g));
    result = r1 ∪ r2;
  casefRg : //similar to fUg
  default : Unacceptable formula;
  return result;
}
```

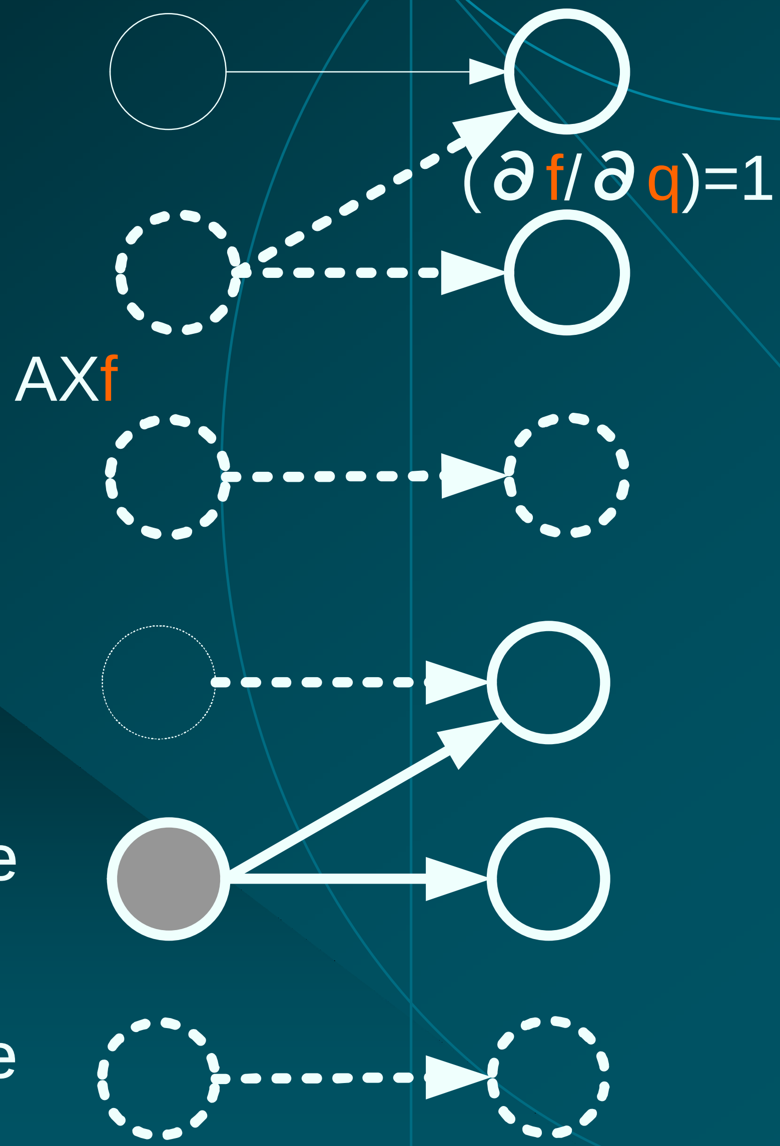
# Transition Coverage Computation

- Forward transition traversing according to the semantics of CTL
  - Different with computing witness paths
    - No need for the sequence of  transitions
  - For example: the STD satisfies AfUg
    - There are two set of states traversed by AfUg
    - f set {S0,S1,S2,S5}; g set {S3,S4,S6}



# Transition Coverage Computation

- Dependency check for observed signal
  - whether the satisfaction of the sub-formula on reached states is dependent on the value of observed signal
- Backward traverse to obtain the covered “transitions”
  - Our target is transitions of FSM which are represented as states of kripke structure
  - Different transitions from one state in the kripke structure reflect all possible inputs in the next clock



# Experimental Results

- Implemented based on VIS using CUDD
  - Language C
  - About 1K lines
- Intermediate results by Model Checking is used by coverage estimation
  - Save computation
- experiments are run on IBM IntelliStation Z-Pro
  - 3.0GHZ CPU
  - 2.3GB RAM.
  - Linux system

# Experimental Results

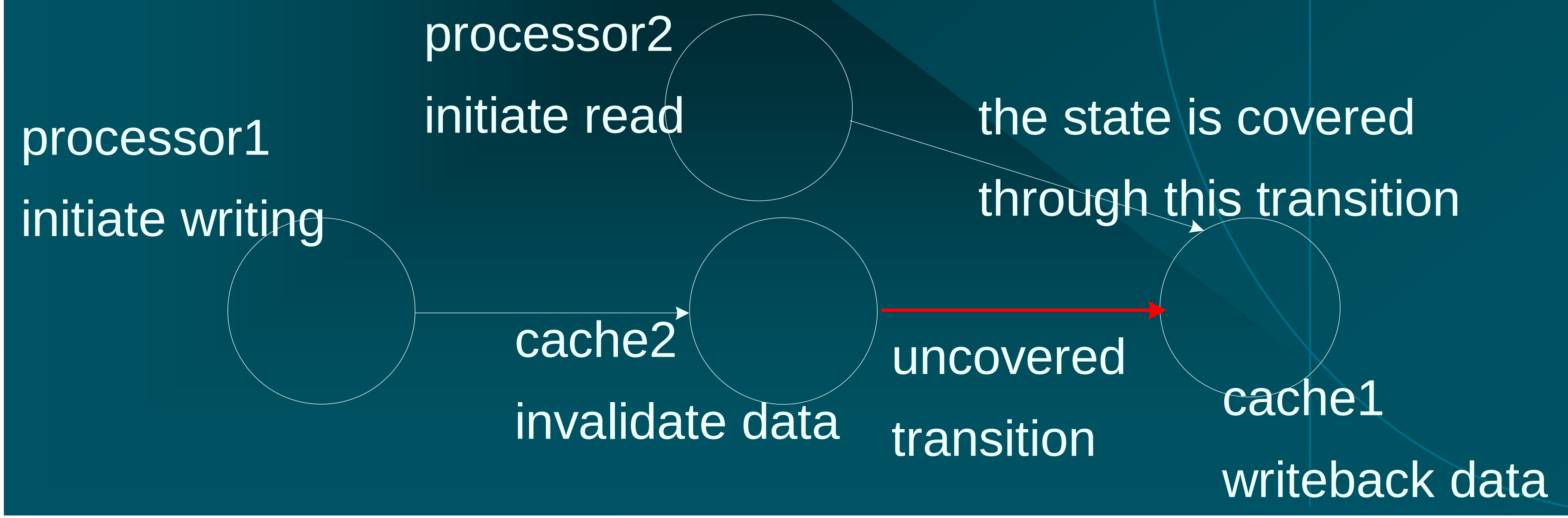
- circuit
  - Full-map directory-based cache coherence protocol
  - Simplified with only one bit per cache
  - Configurable number of processor and memory entry
- Properties
  - 19 properties
  - Not include invariant properties AG(b)
  - Most are in the form of AG(b→AfRg)
- Observed signals
  - 3 observed signals

# Experimental Results

- Coverage results

| Observed Signal | Number of Properties | State Coverage | Transition Coverage |
|---|---|---|---|
| ack1 | 9 | 100% | 100% |
| c1_o | 7 | 100% | 99.72% |
| dirty[0] | 3 | 100% | 95.49% |

- Transition coverage can reveal subtle coverage holes



# Experimental Results

- Computation overhead
  - T1: plain model checking
  - T2: model checking with state coverage estimation
  - T3: model checking with transition coverage estimation

| Protocol Configure | T1 (Seconds) | T2 (Seconds) | T3 (Seconds) |
|---|---|---|---|
| 2p2m | 51 | 65 | 66 |
| 3p2m | 3956 | 4065 | 4071 |
| 2p4m | 9938 | 10444 | 12592 |

- about 20% computation overhead for model checking

# Summary

- Properties completeness analysis is an important issue for model checking
  - Less effort for higher verification quality
- Transition coverage method for circuit FSM
  - Target on transitions of FSM
  - Extension of state coverage
  - Pinpoint through which transition the value of observed signal is checked
  - Able to uncover subtle coverage holes related with transitions
  - low computation overhead

Thank you for your attention!!