

PCI-BUS Implementation Methodology

April. 5, 2001

Tadaaki Tanimoto
Yoichi Kobayashi

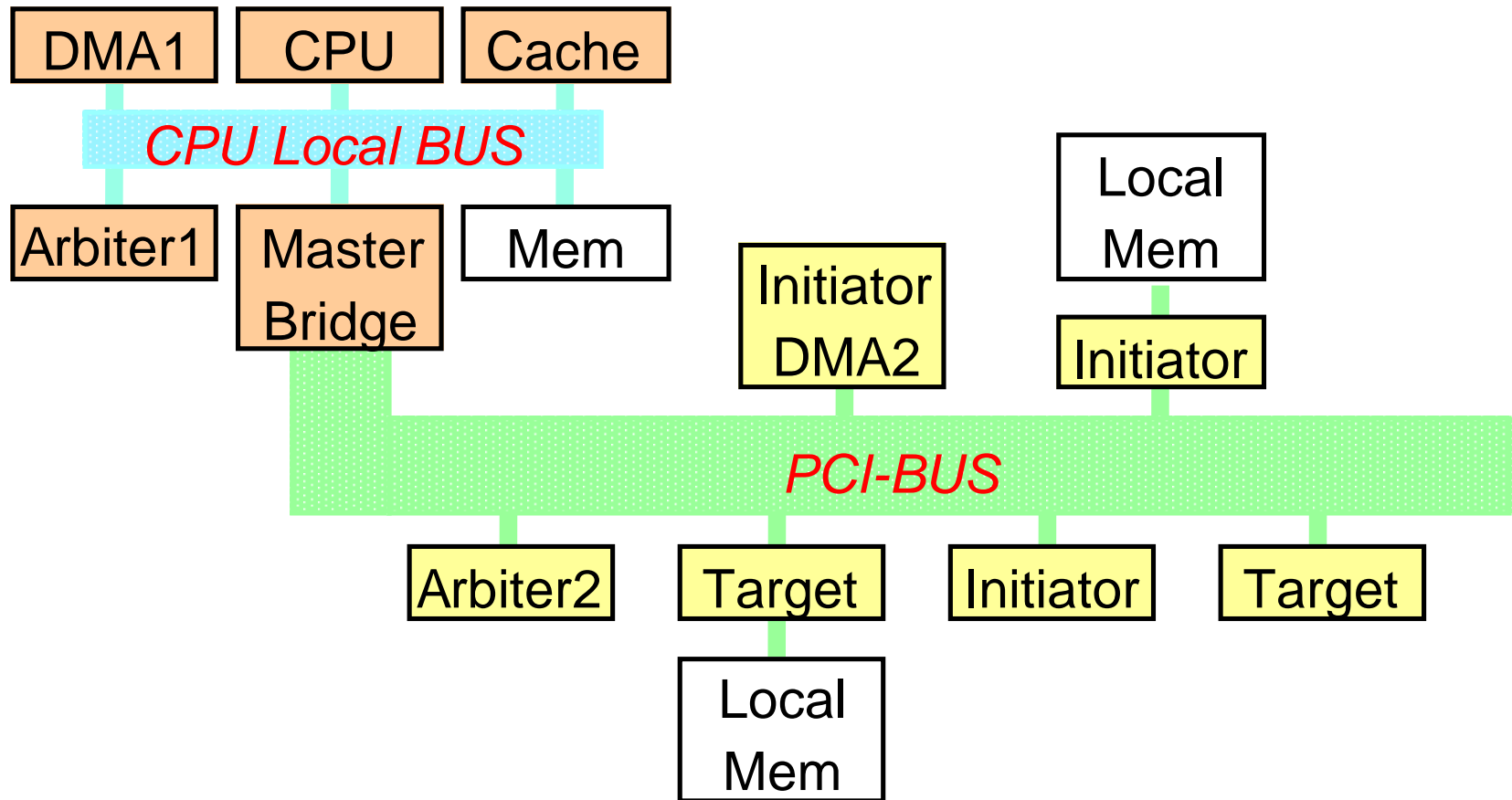
IP Technology Center
System LSI Business Division
Semiconductor & Integrated Circuits
Hitachi Ltd..

Agenda

1. Plan for Implementation
 - 1.1 PCI-BUS system diagram
 - 1.2 Schedule of Implementation
2. Modeling for PSM
3. Hierarchical of Bus transaction
4. Bus Transaction Diagram
5. Current Status
 - 5.1 Problems in using U.C.I Simulator
 - 5.2 Work Around
6. Conclusion

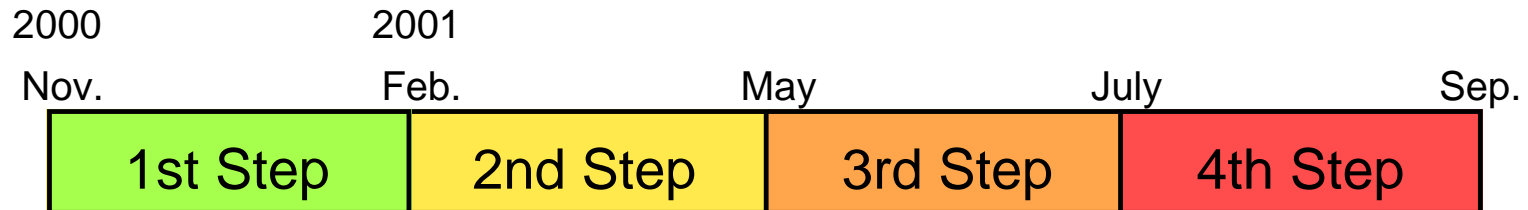
1. Plan for Implementation Implementation

1.1 PCI-BUS system diagram



1. Plan for Implementation Implementation

1.2 Schedule of Implementation



Maybe Release

1st Step. Support Basic Data Transfer

- Without Read / Write configuration registers
(use user defined files for configuration)
- Support N devices

2nd Step. Support DMA

3rd Step. Support ALL Data Transfer

4th Step. Support Configuration Cycle

2. Modeling for PSM

(1) About PSM

PSM can represent data, behavior, and status of system in a unified model.

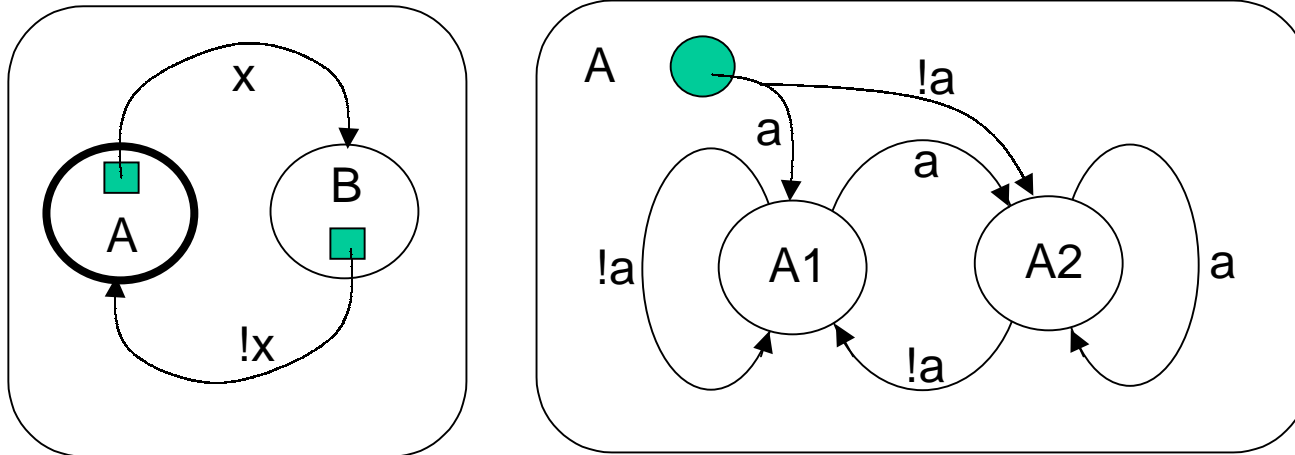
PSM is an extension of HCFSM(Hierarchical Concurrent FSM),that is HCSFSMD(Hierarchical Concurrent Super FSM with Data path).

(2) Apply PSM to model PCI-BUS

- PCI-BUS is fully synchronous BUS. All signals in PCI-BUS are synchronized with system clock.
- PSM enable us to classify the transaction of PCI-BUS hierarchically, thus we think PSM is suitable for modeling PCI-BUS transaction behavior.

3. Hierarchical of Bus transaction

(1) Terminate-On-Interrupt(TOI)

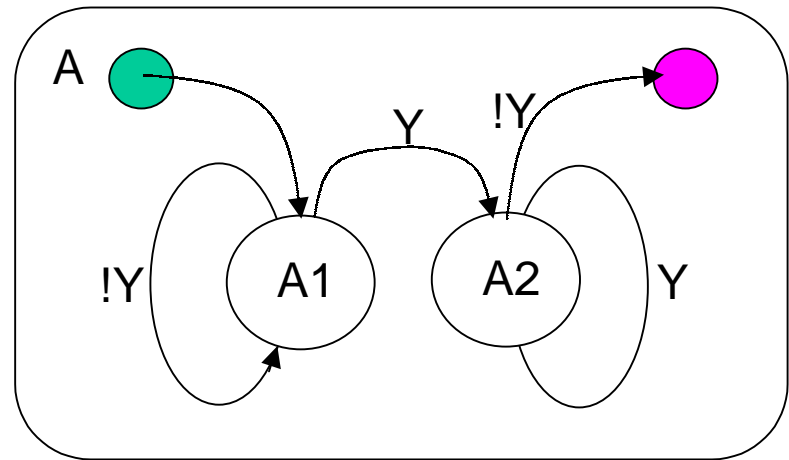
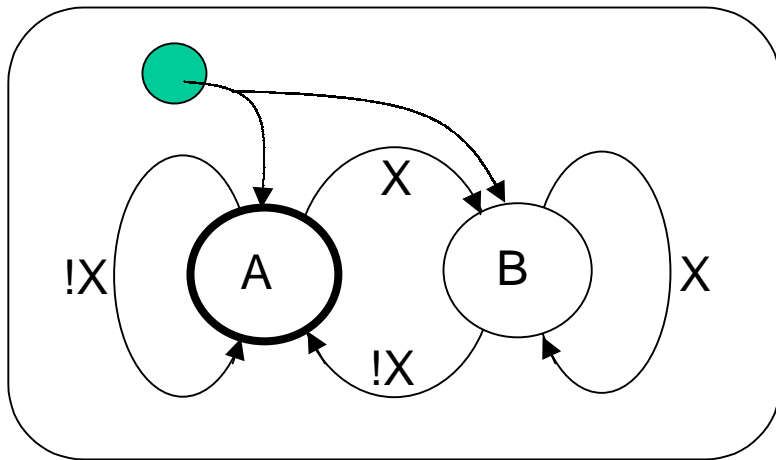


The example show PSM where

- A is hierarchical state
- A1 and A2 are two sub-states of A
- Both A1 and A2 are actual start states
- There is no “actual” stop state for A. However, the execution in A will terminate at any time when condition “x” is true(TOI).

3. Hierarchical of Bus transactionZ

(2) Transit-On-Completion(TOC)



The example show PSM where

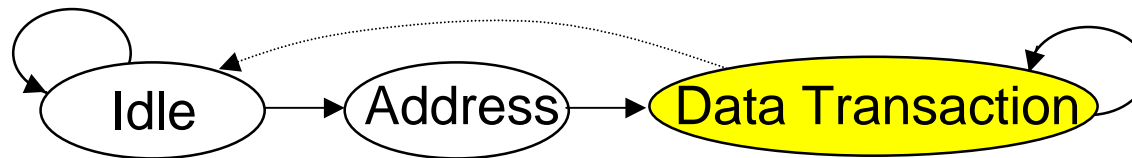
- A is hierarchical state

- A1 and A2 are two sub-states of A

- If the current state of the machine is A2, and if condition “Y” is false then the machine will exit the hierarchical state A. Otherwise, if “Y” is true, the machine will remain in state A2. **Note that condition “X” will be ignored since the transition from A to B is TOC.**

4. Bus Transaction Diagram

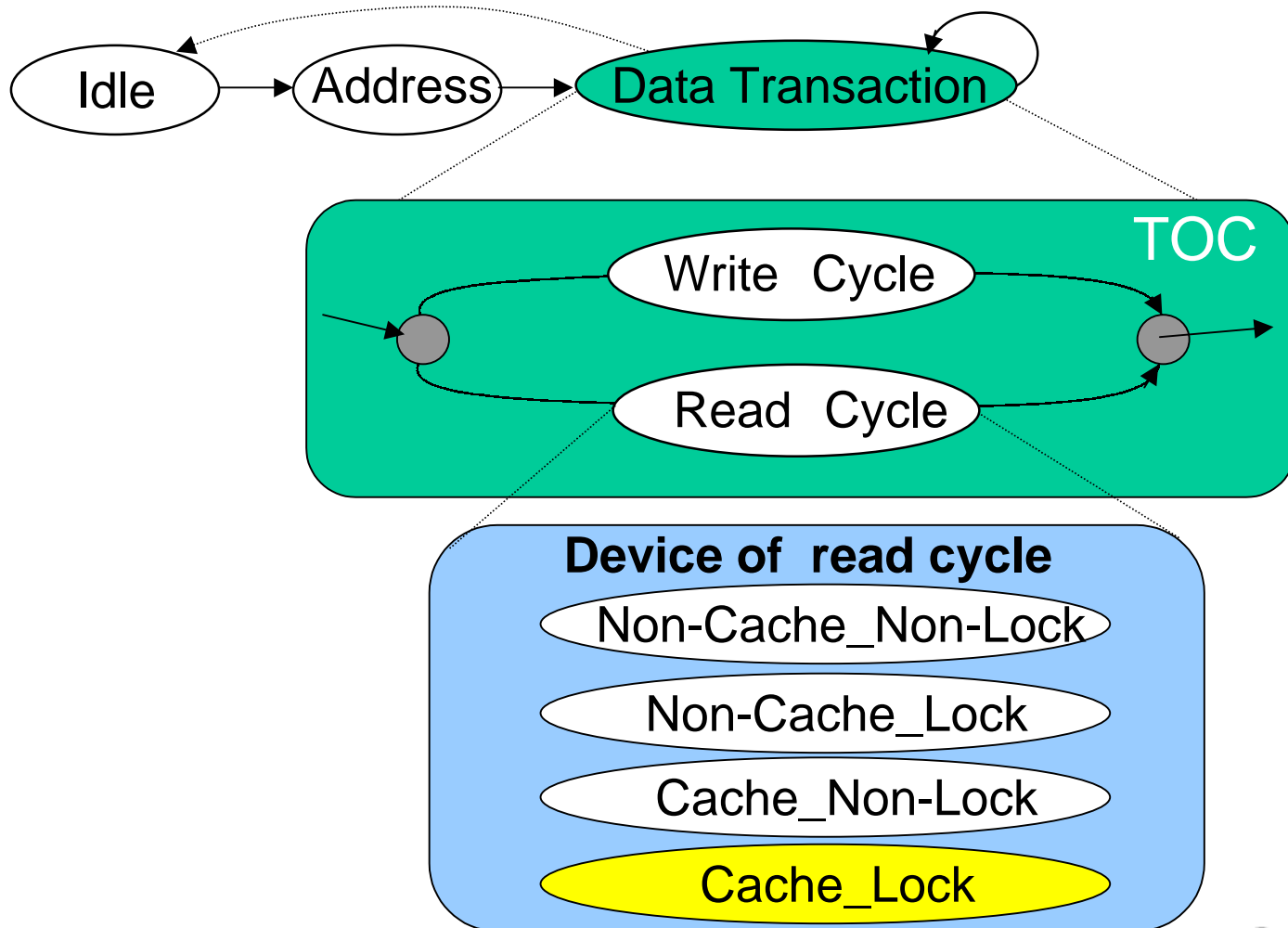
(1) PCI-BUS Based Cycle



Initiator (data sender) device and target (data receiver) device perform exact bus transaction behaviors, respectively.

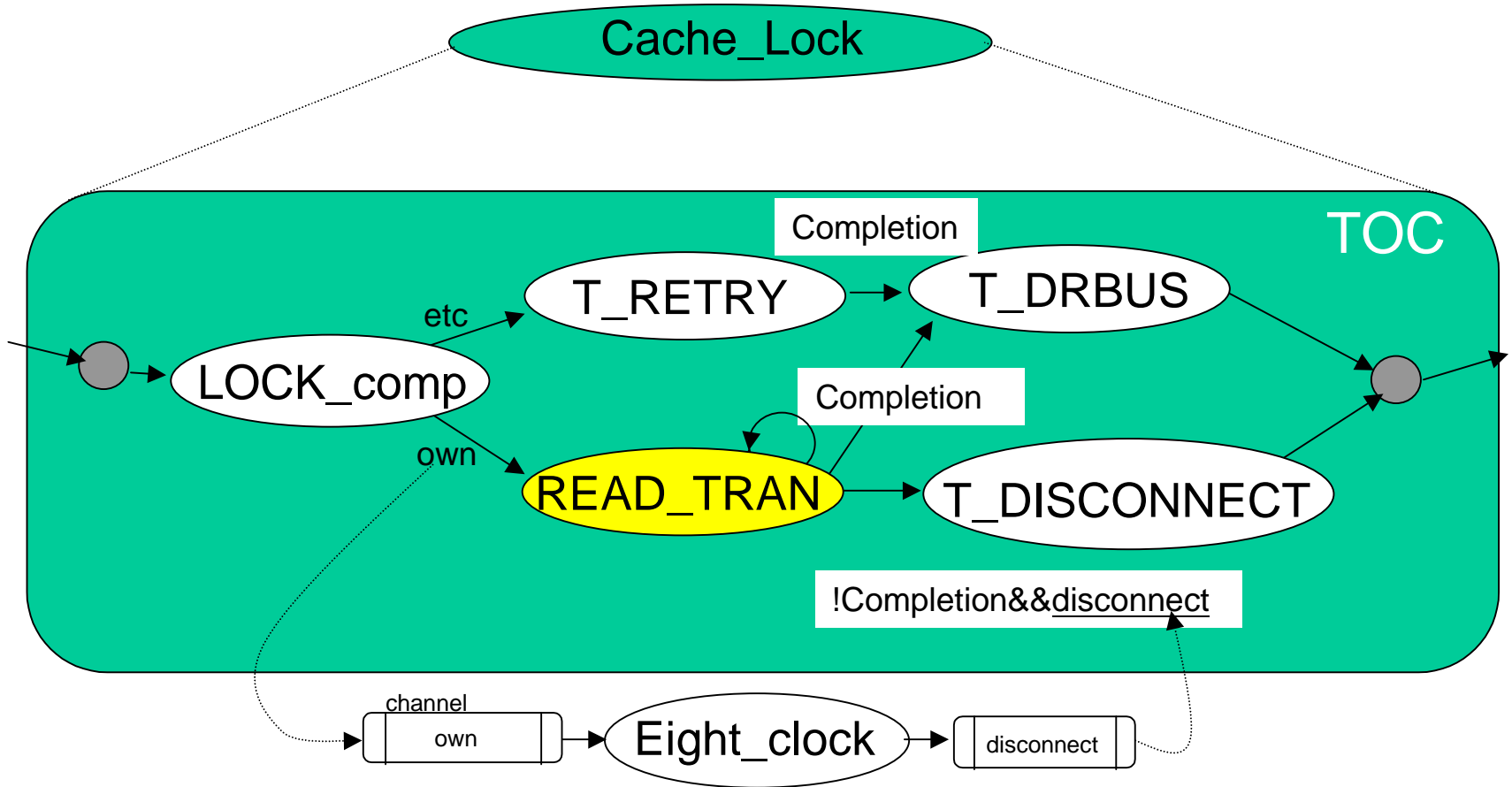
4. Bus Transaction Diagram

(2) Data transaction of target device

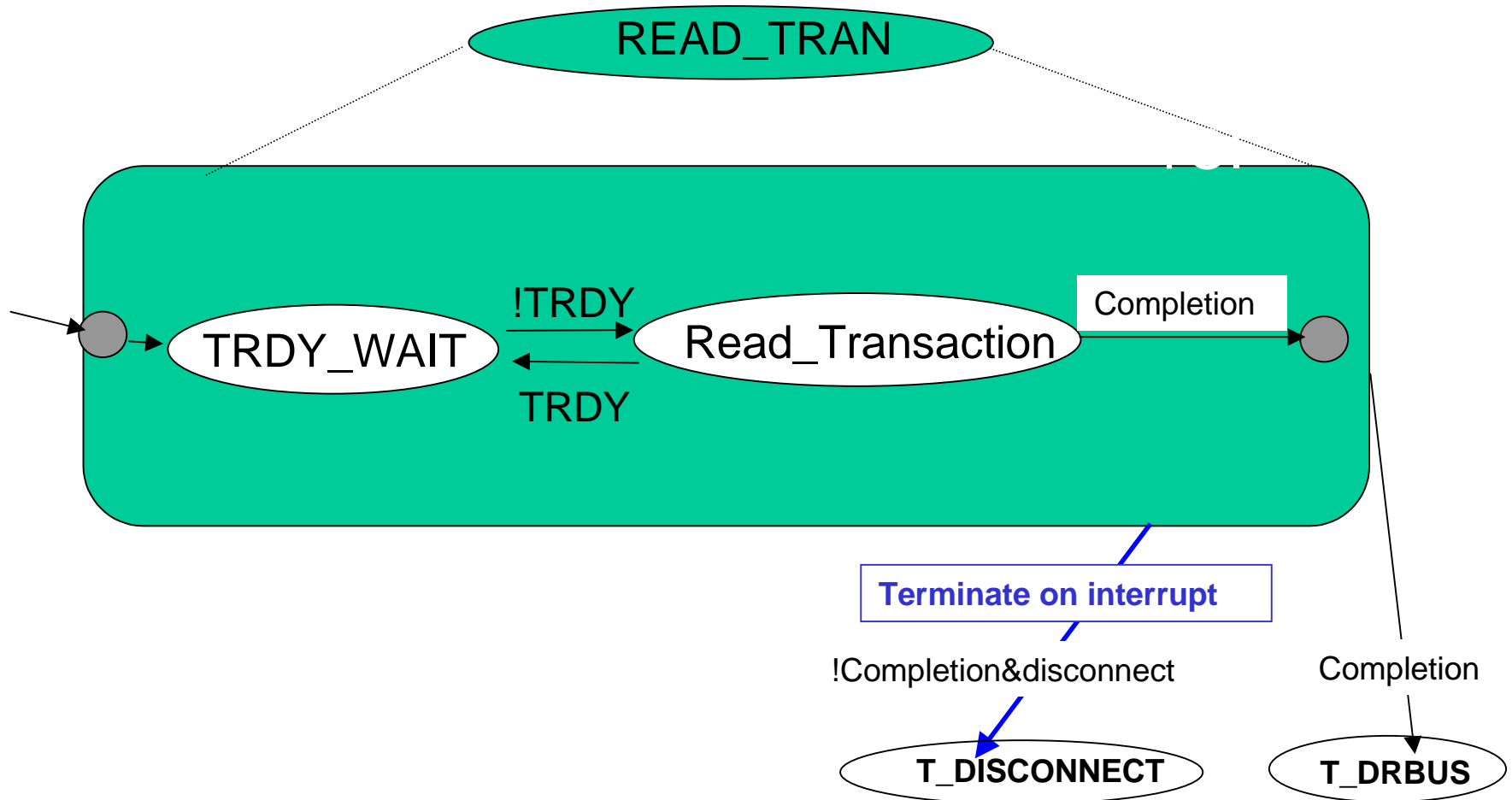


4. Bus Transaction Diagram

(3) Read Cycle of Cache_Lock



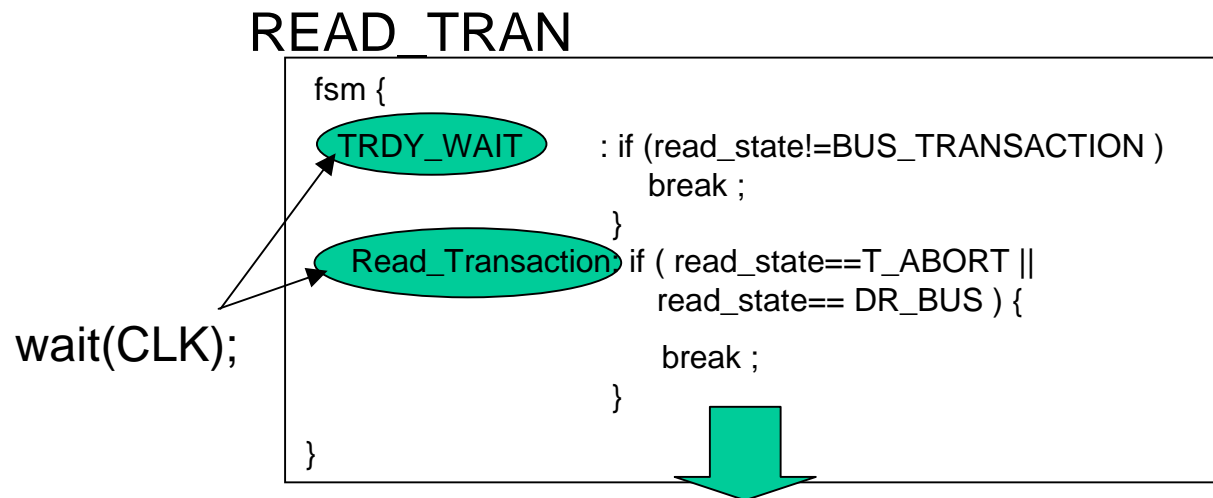
4. Bus Transaction Diagram (4) READ_TRAN



5. Current Status

5.1 Problem in using U.C.I. Simulator

(1) Can not realize FSM with even based transition



SpecC warning: dead lock in the program, aborted

All PCI-BUS transaction are fully synchronized with system clock, but we want to model them at bus transaction level which we think Super FSMD is suitable for describe.

→ We want to use FSM syntax combined with notify-wait which enable us to describe state transition triggered event not clock.

5. Current Status

5.1 Problem in using U.C.I. Simulator

(1) Can not realize FSM with even based transition.
(Sample)

```
#include <stdio.h>
#include <stdlib.h>
#include "state.sh"
#define N 2
behavior A(in event CLK) {

    void main(void) {
        printf("State A¥n");
        wait( CLK );
        printf("State A¥n");
    }
};
behavior B(in event CLK) {
    void main(void) {
        printf("State B¥n");
        wait( CLK );
        printf("State B¥n");
    }
};
```

```
behavior Main {
    event CLK ;
    A IDLE( CLK ) ;
    B ADDR( CLK ) ;
    int main(int argc, char **argv) {
        int i ;

        for(i=0;i<=N;i++) {
            notify( CLK );
            fsm { IDLE: { goto ADDR; }
                ADDR: { goto IDLE; }
            }
            printf( "AAA¥n" );
        }
    }
};
```

Result

SpecC warning: dead lock in the program, aborted State A

SpecC warning: dead lock in the program, aborted State A

5. Current Status

5.1 Problem ins using U.C.I. Simulator

(2) try-trap does not represent trap semantics.

Eight_clock

```
try { T_DEVICE.main(); }  
trap(disconnect) { T_DISCONNECT.main(); }
```

```
if ( ir8k>8 ) {  
    initial_micro = 0 ;  
    state = T_DISCONNECT ;  
    notify( disconnect ) ;  
}
```

Equal Try-interrupt !!



Can not realize TOI with try-trap.

Please fix the bug of try-trap

5. Current Status

5.1 Problem in using U.C.I. Simulator

(2) try-trap does not represent trap semantics.
(Sample)

```
#include <stdio.h>
#include <stdlib.h>
#include "state.sh"
behavior RAM_CLK( in event CLK ,
                 out event RST ) {

    void main(void) {
        printf( "Trap CLK¥n" );
        notify(RST);
        printf( "Trap CLK1¥n" );
    }
};

behavior RAM_RST( in event CLK ,
                 in event RST ) {

    void main(void) {
        printf( "Trap RST¥n" );
    }
};
```

```
behavior MASTER ( in event CLK , in event RST ) {
    RAM_CLK R_CLK(CLK, RST );
    RAM_RST R_RST(CLK, RST );
    void main( void ) {
        try    { R_CLK.main();}
        trap(RST) { R_RST.main();}
    }
};

behavior Main {
    event CLK ;
    event RST ;
    MASTER M( CLK , RST ) ;
    int main(int argc, char **argv) {
        int i ;
        for(i=0;i<=2;i++) {
            notify( RST );
            printf( "RST¥n" );
            M.main();
            waitfor( 0 );
        }
    }
};
```

Result

```
RST
Trap CLK
Trap RST
Trap CLK1
```

5. Current Status

5.2 Work Around

(1) Can not realize FSM with even based transition.

Can not realize FSM with even based transition. We employ “switch – case” instead of using fsm syntax in SpecC.

Since we can not use “notify(clock)” which describe clock edge, we use pre-cycle value and present cycle value for recognizing signal edge.

```
switch (read_state) {
  case TRDY_WAIT : WAIT.main();
                  if (read_state!=BUS_TRANSACTION )
                    break ;
                  }
  case Read_Transaction: Read_Transaction .main();
                        if ( read_state==T_ABORT ||read_state== DR_BUS ) {
                          break ;
                        }
}
```

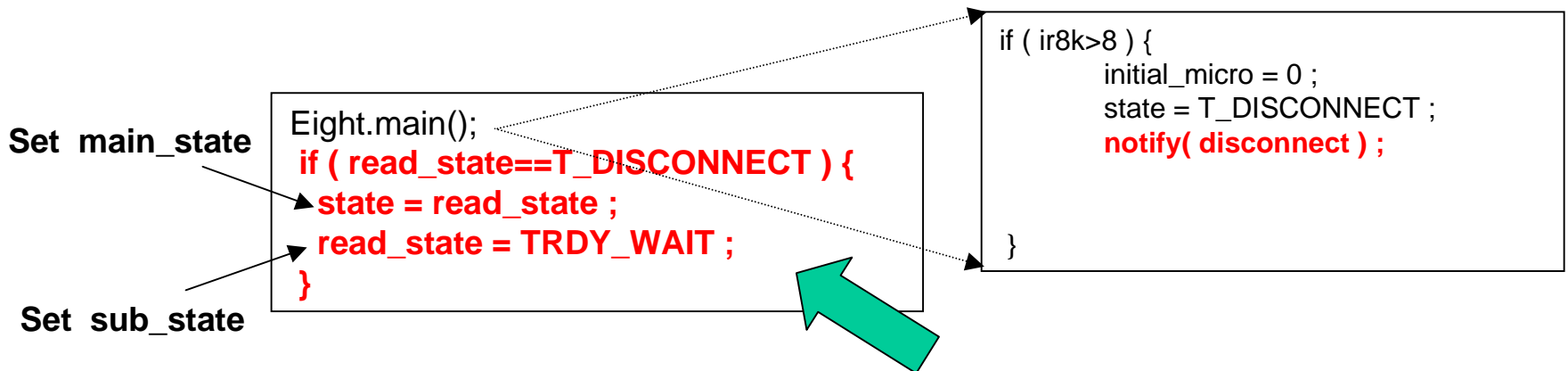
5. Current Status

5.2 Work Around

(2) try-trap does not represent trap semantics.

After Completion of notify(transaction), we should initialize FSM for preparation of next transaction, because **transaction should be start from fake start state in the light of TOI or TOC semantics.**

This kind of mechanism should be supported by fsm syntax/semantics.



Initialization of hierarchical state READ_TRAN

6. Conclusion

- (1) In order to realize synchronization in hardware semantics, we think fsm with event based transaction is inevitable.
- (2) For modeling Hardware, below item are must.
 - 1) par with fully parallel execution semantics.
 - 2) fsm with event based transition.
- (3) try-trap enable us to represent TOI behavior, but now we have problem the fact that try-trap does not represent trap semantics.

Above three items should be fixed ASAP.

6. Conclusion

6.1 Problem in synchro event

```
#include <stdio.h>
#include <stdlib.h>
#include "state.sh"
#define N 2
behavior A(in event CLK, out int O ) {
  void main(void) {
    printf("State A 1\n");
    wait( CLK );
    printf("State A 2\n");
    O = 1 ;
  }
};
behavior B(in event CLK, out int O) {
  void main(void) {
    printf("State B 1\n");
    wait( CLK );
    printf("State B 2\n");
    O = 0 ;
  }
};
```

```
behavior Main {

  event CLK ;
  int O ;
  A IDLE( CLK, O ) ;
  B ADDR( CLK, O ) ;

  int main(int argc, char **argv) {
    int i ;
    for(i=0;i<=N;i++) {
      notify( CLK );
      par { IDLE.main() ;
            ADDR.main() ;
          }
      printf( "O = %d\n", O );
    }
  }
};
```

Result

SpecC warning: dead lock in the program, aborted
State A 1
State B 1

No Output!!

6. Conclusion

6.2 Problem in par-statement

```
#include <stdio.h>
#include <stdlib.h>
#include "state.sh"
#define N 2
behavior A(out int O ) {
  void main(void) {
    if ( O == 0 ) {
      printf( "A O is 0\n" );
    } else {
      printf( "A O is 1\n" );
    }
    O = 0 ;
  }
};
behavior B(inout int O, event done) {
  void main(void) {
    if ( O == 0 ) {
      printf( "B O is 0\n" );
    } else {
      printf( "B O is 1\n" );
    }
    O = 1 ;
  }
};
```

```
behavior Main {

  int O ;
  event done ;

  A IDLE( O ) ;
  B ADDR( O, done ) ;

  int main(int argc, char **argv) {
    int i ;

    for(i=0;i<=N;i++) {
      O = 1 ;
      par { IDLE.main() ;
            ADDR.main() ;
          }
      // wait( done );
      printf( "O = %d\n", O );
    }
  }
};
```

Result

```
A O is 1
B O is 0
O = 1
A O is 1
B O is 0
O = 1
A O is 1
B O is 0
O = 1
```

Both A and B should be one.