bm_decomp (release v0.9): An Introduction and Tutorial

An Extension to the MINIMALIST CAD Package

Melinda Agyekum Steven Nowick

Columbia University

{melinda, nowick}@cs.columbia.edu}

November 15, 2007

This work was partially supported by NSF ITR Award No. NSF-CCR-0086036.

bm_decomp Developers

Principal Architects: [2006-present]

- –Melinda Y. Agyekum: system designer & primary implementer
- –Steven M. Nowick: project advisor

Documentation:

- –Publication: Melinda Y. Agyekum and Steven M. Nowick, *"A Cycle-Based Decomposition of Asynchronous Burst-Mode Controllers,"* In *Proc. of the 13th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)* , pages 129-142, March 12-14, 2007, Univ. of California, Berkeley (Berkeley, California, USA).

- –Presentation Slides: Melinda Y. Agyekum and Steven M. Nowick *<same title as publication>,* Presented at ASYNC 2007

bm_decomp: Download Site

Accessible on the Web from:

www1.cs.columbia.edu/~nowick/asynctools

Version: Linux

Includes:

- complete tutorial
- benchmark examples
- other documentation

Supporting Document

BM_DECOMP is an extension of the MINIMALIST CAD tool package.
To download MINIMALIST, see:

www1.cs.columbia.edu/~nowick/asynctools

MINIMALIST tool documentation:

– Book: Robert M. Fuhrer and Steven M. Nowick,
"Sequential Optimization of Asynchronous and Synchronous Finite-State Machines: Algorithms and Tools",
Kluwer Academic Publishers,
Boston, MA (2001), ISBN 0-7923-7425-8.

– PhD Thesis: Robert M. Fuhrer, *<same title as book>*,
Columbia University, Dept. of Computer Science,
May 1999.

Contributions

bm_decomp:

- Automated decomposition tool
  - Decomposes a single Burst-Mode (BM) controller specification into multiple Burst-Mode controller specifications
  - Together the decomposed BM controllers maintain the same behavior as the original
- Inter-controller communication protocol
  - Allows the decomposed controllers to communicate with each other
- Small amount of additional hardware required
  - Primary Input Latches
    - Used to filter unwanted inputs
  - Latch Enable Logic
    - Used to selectively enable the primary input latches
  - Primary Output Generators
    - Used to generate primary outputs

Motivation

Primary Goals:

- Synthesizability
  - Provide a means to synthesize larger controllers
- Runtime
  - Improve runtime of CAD tool (esp. for large controllers)

Secondary Goals:

- Smaller Next-State Complexity
  - Simplify BM fundamental mode timing constraint
- Potential Reduction in Power Consumption
  - Only a single controller is active at any given time

More Detailed Contributions

Method for Decomposition

- For Burst-Mode:

  4 Major Parts

  - Decomposition algorithm
  - Controller micro-architecture
  - Inter-controller communication protocol
  - Auxiliary hardware

    Optimizations to eliminate or simplify hardware

- Method for Extended Burst-Mode (XBM)

CAD Tool Implementation

- For Both BM & XBM

An extension to the MINIMALIST CAD tool package!

Outline

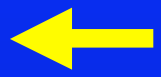
PART I: Technical Overview

- Quick Review: Burst-Mode Controllers
- Decomposition Method
  - Micro-Architecture
  - Communication Protocol
  - Hardware Implementation
- bm_decomp: Basic Tool Features

PART II: Tutorial

- Design Examples + Hands-On Tutorial

PART I: Technical Overview

- Quick Review: Burst-Mode Controllers 
- Decomposition Method
  - Micro-Architecture
  - Communication Protocol
  - Hardware Implementation
- bm_decomp: Basic Tool Features

Quick Review: "Burst-Mode" Controllers

Synthesis style for individual asynchronous FSM's:

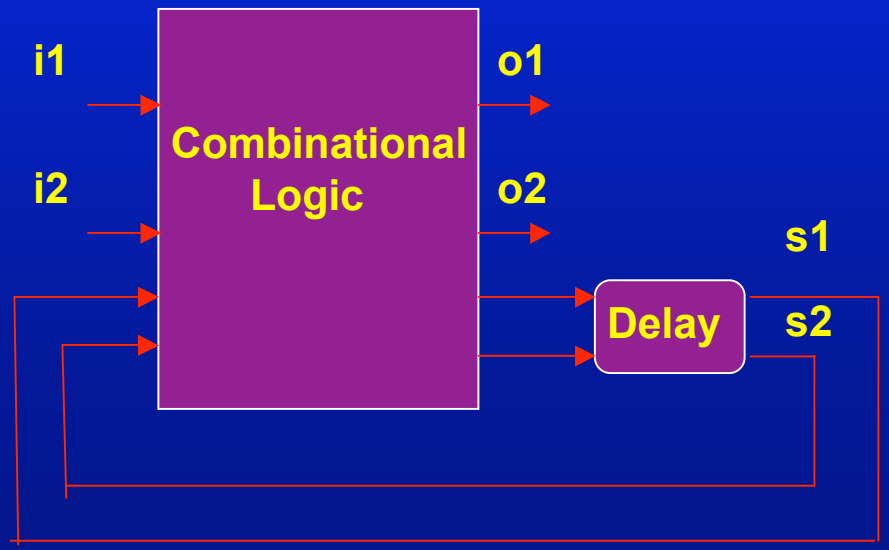
- Mealy-type
- allows:
  - *multiple-input changes*
  - concurrent behavior
- target technology: normal synchronous cell libraries
- optimization algorithms: comprehensive set

- Brief History:...
  - Based on informal approach at HP Labs:
    - Davis, Coates, Stevens [1986-, and earlier]
  - Formalized and constrained at Stanford: Nowick/Dill [91]
    - Nowick/Dill first to develop a correct synthesis method

Quick Review: "Burst-Mode" Implementation

Burst-Mode Controllers

- A Huffman-Style asynchronous state machine
- Consists of:
  - Primary inputs
  - Primary outputs
  - Fed-back state
- State is stored in the fed-back loops



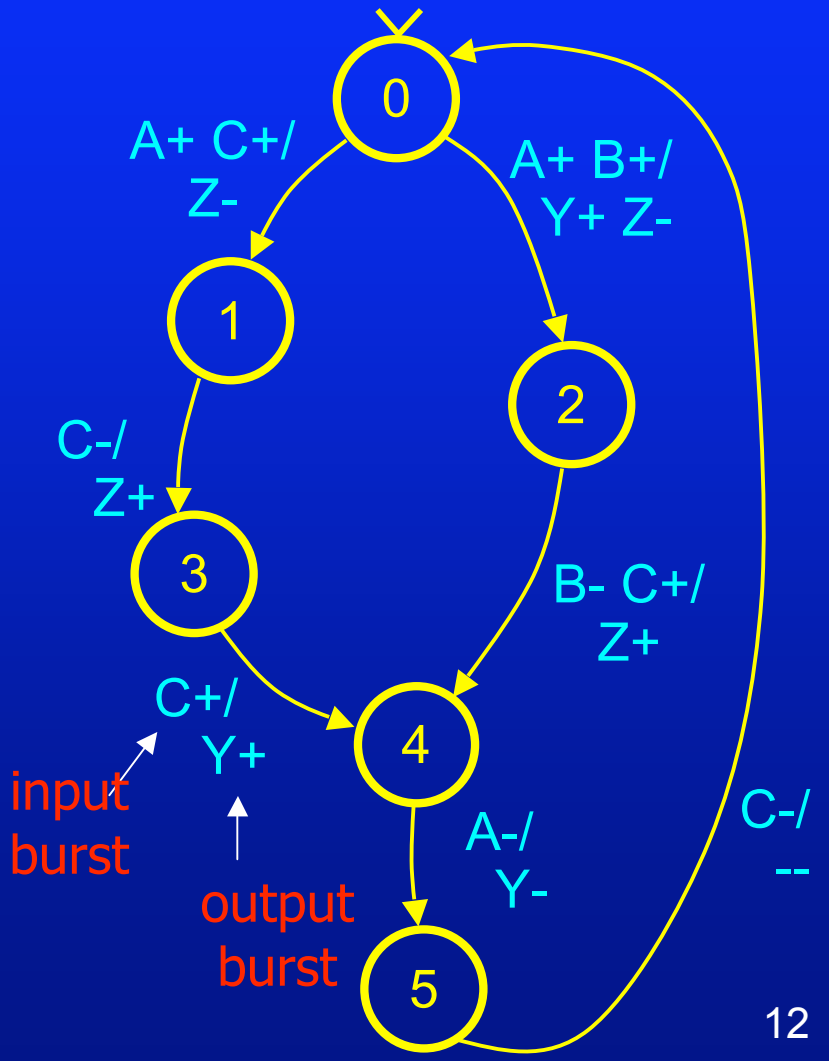
Quick Review: "Burst-Mode" Specifications

Example: Burst-Mode (BM) Specification:

Initial Values:
ABC = 000
YZ = 01

- Inputs in specified *"input burst"* can arrive in any order and at any time

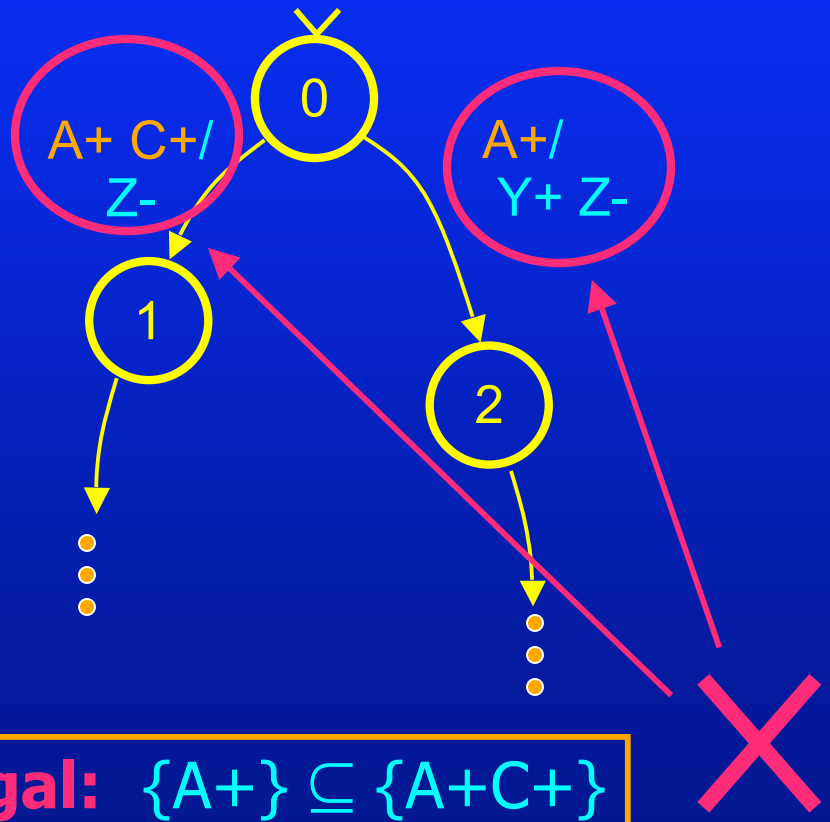
  - After all inputs arrive, generate *"output burst"*



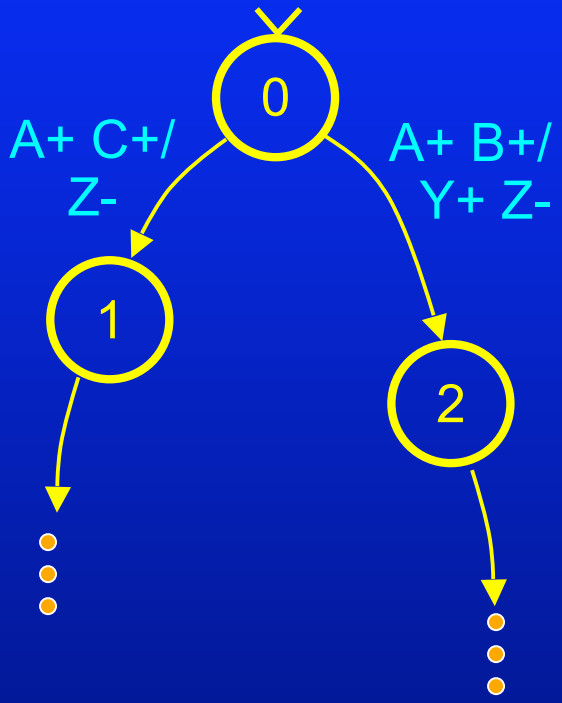
Note:
-input bursts: must be non-empty
(at least 1 input per burst)
-output bursts: may be empty
(0 or more outputs per burst)

Quick Review: "Burst-Mode" Specifications

1. ***"maximal set property":*** in each specification state, no input burst can be a subset of another input burst



illegal: {A+} ⊆ {A+C+}

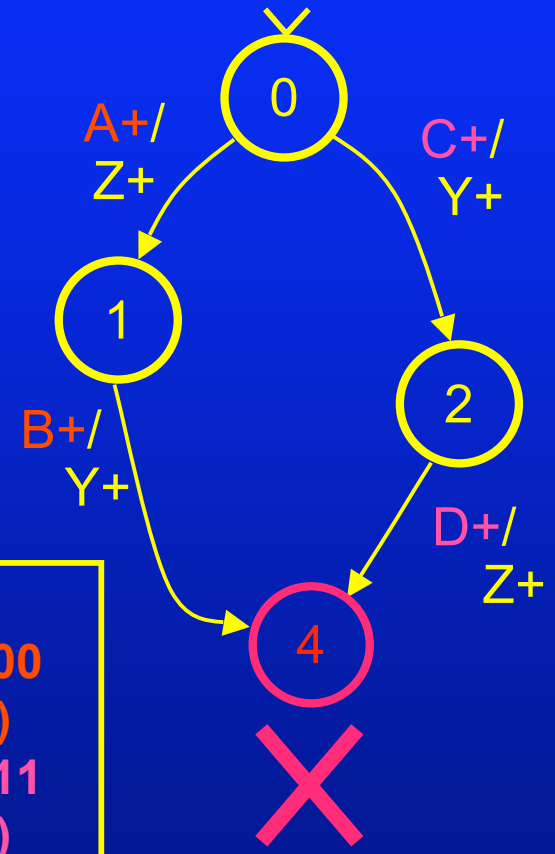


legal

…ambiguous: what to do when only input A+ arrives?:
- wait for C+? or output Y+ Z-??

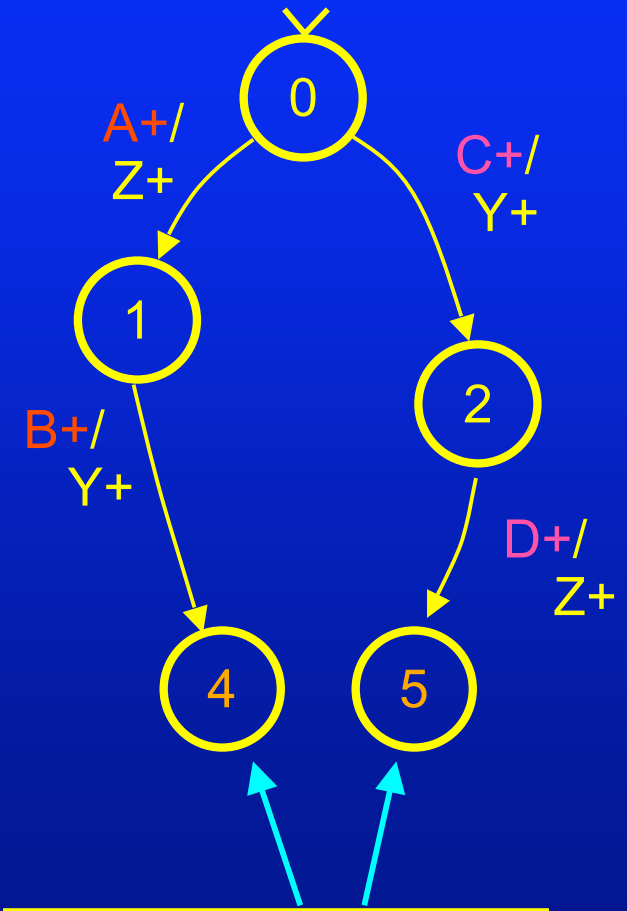
Quick Review: "Burst-Mode" Specifications

2. ***"unique entry point":*** each specification state must be entered at a 'single point' *(guarantees hazard-free synthesis)*



Entering State 4:
- from State 1: **ABCD=1100 (YZ=11)**
- from State 2: **ABCD=0011 (YZ=11)**

illegal: 2 different input/output values when entering state 4

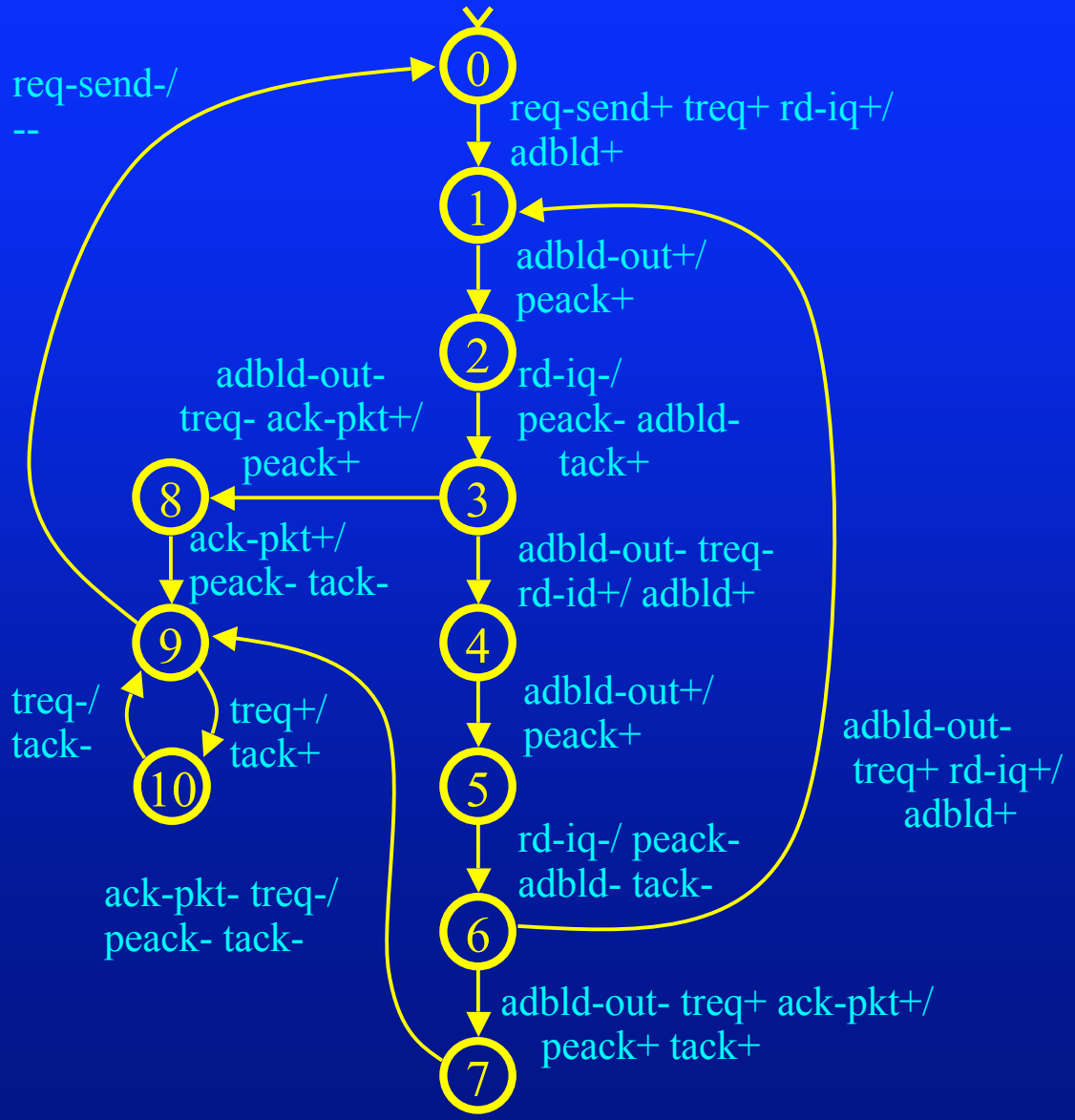


legal: solution=split state 4

Quick Review: Example "PE-SEND-IFC" (HP Labs)

Inputs:
req-send
treq
rd-iq
adbld-out
ack-pkt

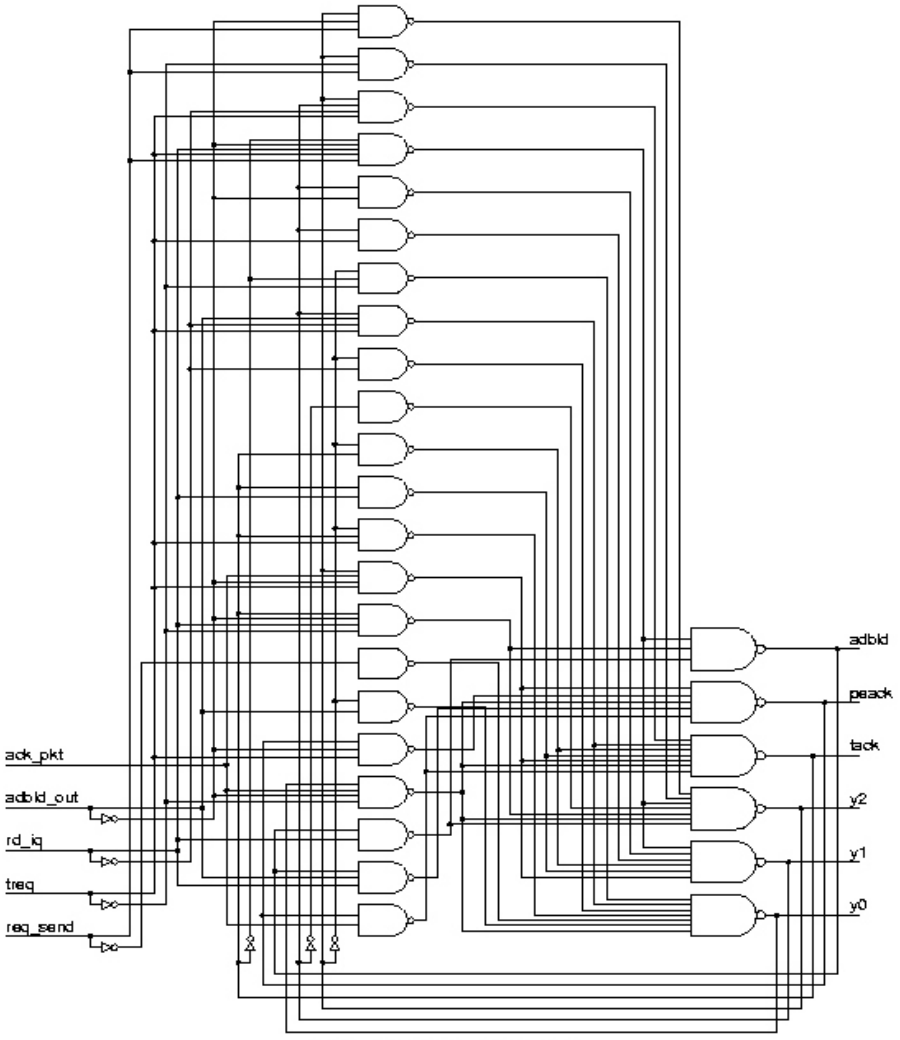
Outputs:
tack
peack
adbld



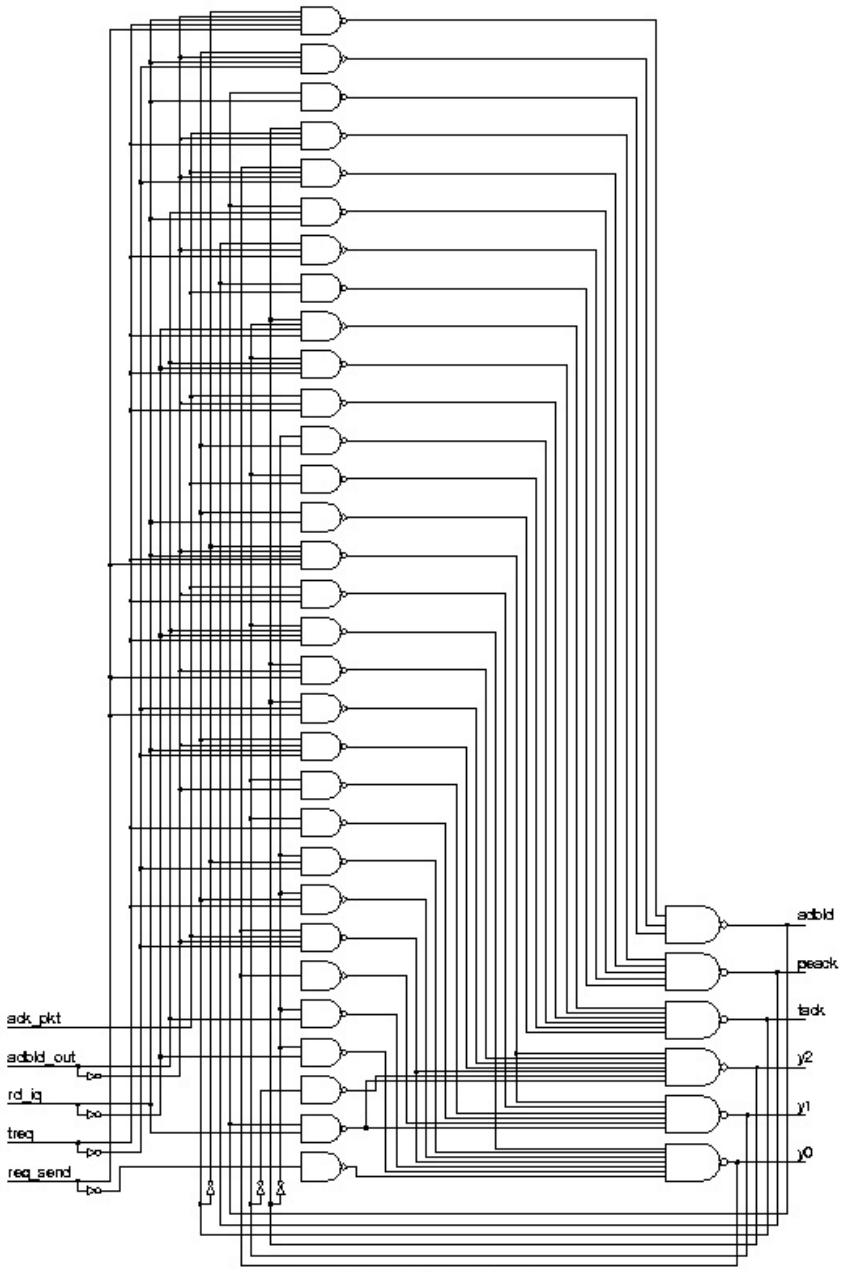
From HP Labs
"Mayfly" Project:
B.Coates, A.Davis, K.Stevens,
"The Post Office Experience: Designing a Large Asynchronous Chip",
INTEGRATION: the VLSI Journal, vol. 15:3, pp. 341-66 (Oct. 1993)

Quick Review: Ex. "PE-SEND-IFC" contd.

Design-Space Exploration using MINIMALIST:
optimizing for area vs. speed

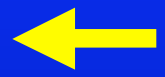


PE_SEND_IFC-FL



PE_SEND_IFC-Fs

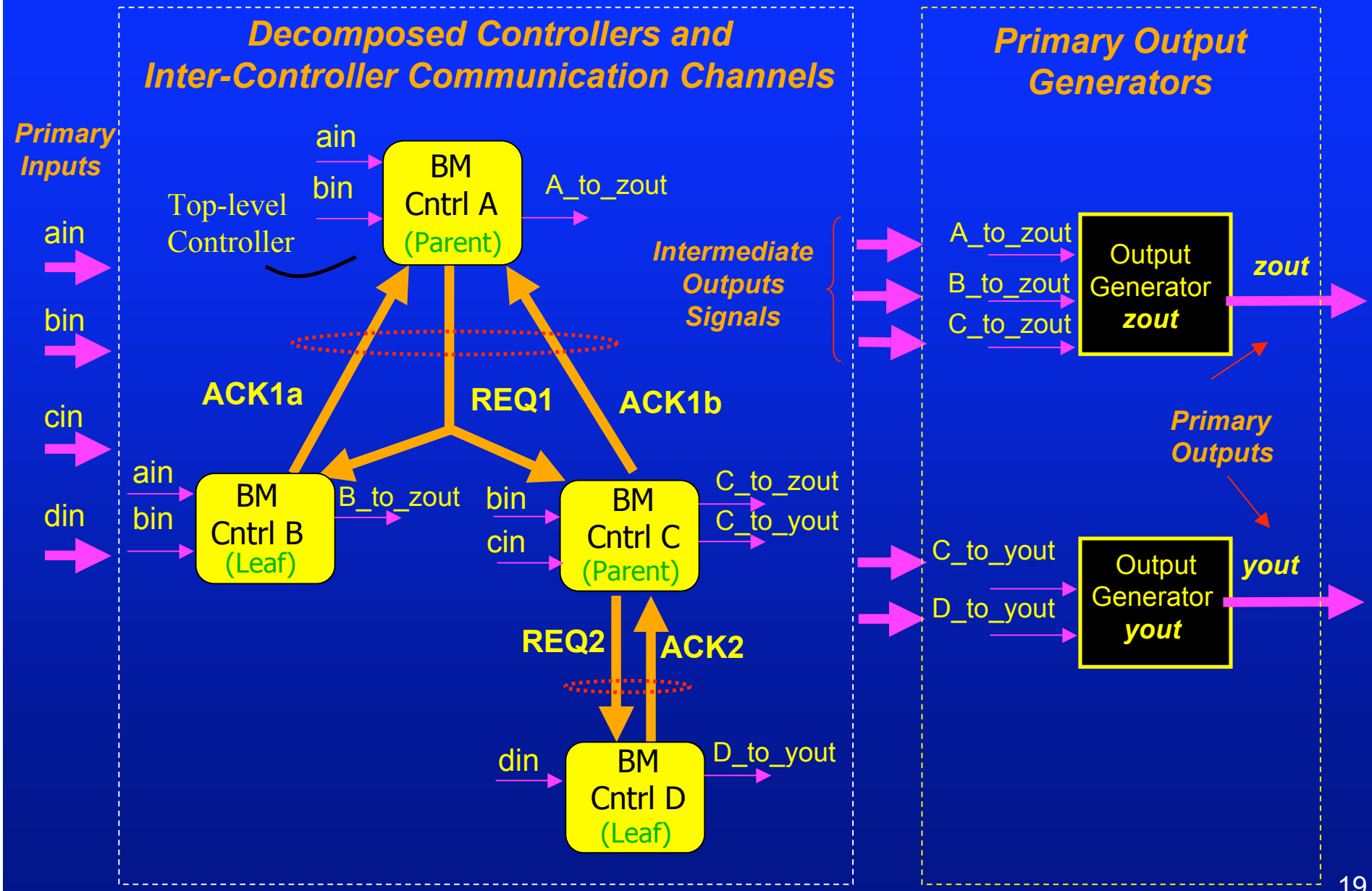
PART I: Technical Overview

- Quick Review: Burst-Mode Controllers
- Decomposition Method 
  - Micro-Architecture
  - Communication Protocol
  - Hardware Implementation
- bm_decomp: Basic Tool Features

Decomposition Method: Micro-Architecture

- Parts of the Micro-Architecture
  - A set of decomposed controllers
    - Includes input latches and latch enable hardware
  - Primary output generators
  - Point-to-point communication through handshake channels

Decomposition Method: Micro-Architecture

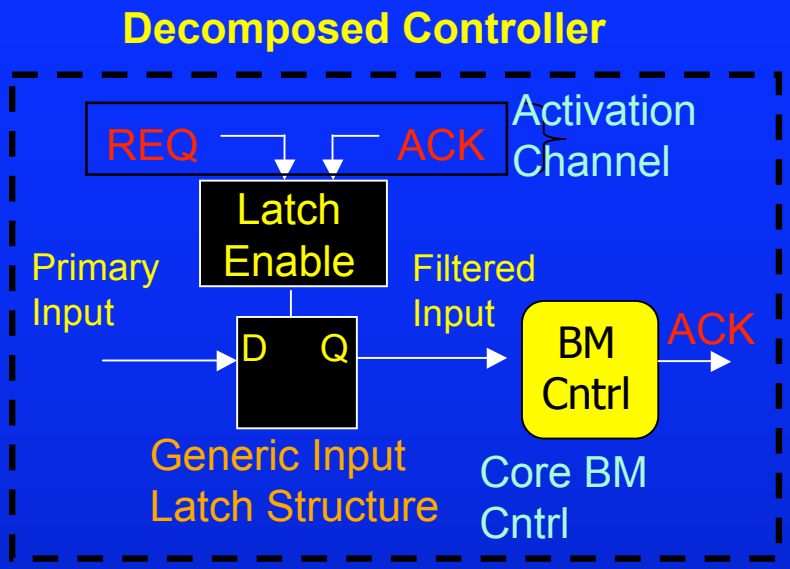


Decomposed Controllers and Communication Protocol

- Three Types of Controllers
  - Parent Controller
    - *Activates* children controllers by issuing a *REQ*
    - Becomes *activated* by a parent controller by responding with *ACK*
  - Leaf Controller
    - Becomes *activated* by a parent controller by responding with *ACK*
    - Cannot activate other controllers
  - Top-Level Controller
    - A specialized parent controller that is initialized at system start-up
    - Does not have a parent controller

Parents and children communicate through 4-phase handshaking protocol

Decomposed Controllers: Input Latch Hardware



Primary Input Latches

- Transparent D-latches
  - Control when primary inputs can safely be received
- By default all primary inputs are blocked

Latch Enable Hardware: controlled by "activation channel"

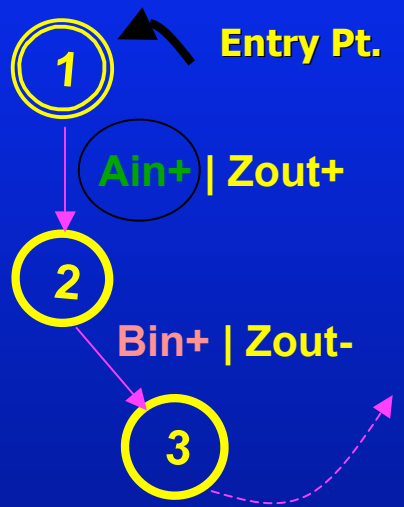
- Handles two scenarios:
  - "Controller as a leaf" ➔ has permitting unit
  - "Controller as a parent" ➔ has permitting and prohibiting units

Decomposed Controllers: Input Latch Hardware

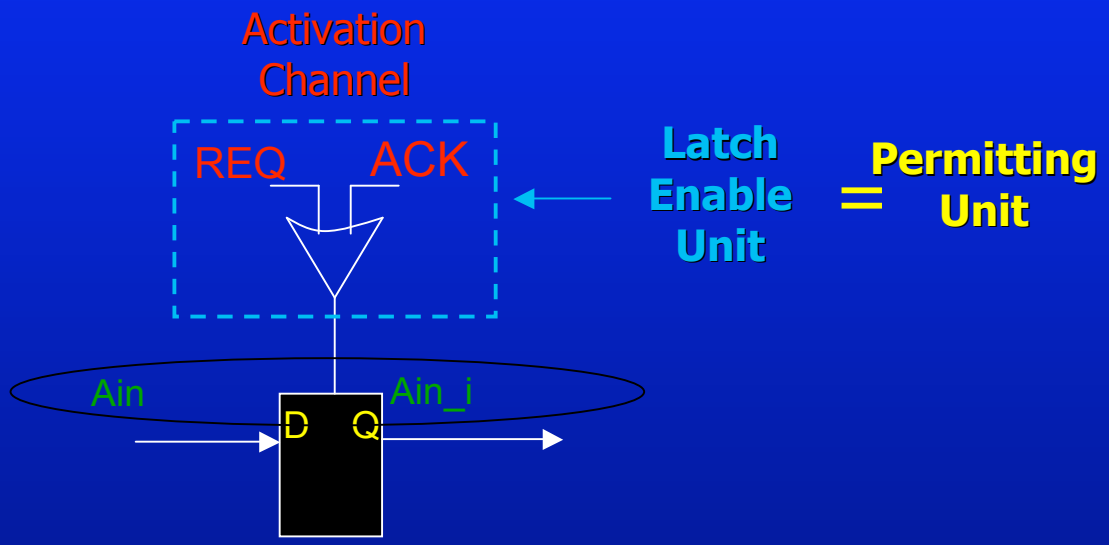
"Controller as a Leaf"

Idea:

- Latch permitting unit used to control the input latch
- Inputs are permitted when leaf gets activation from parent



BM Specification Fragment



Latch Structure for input Ain

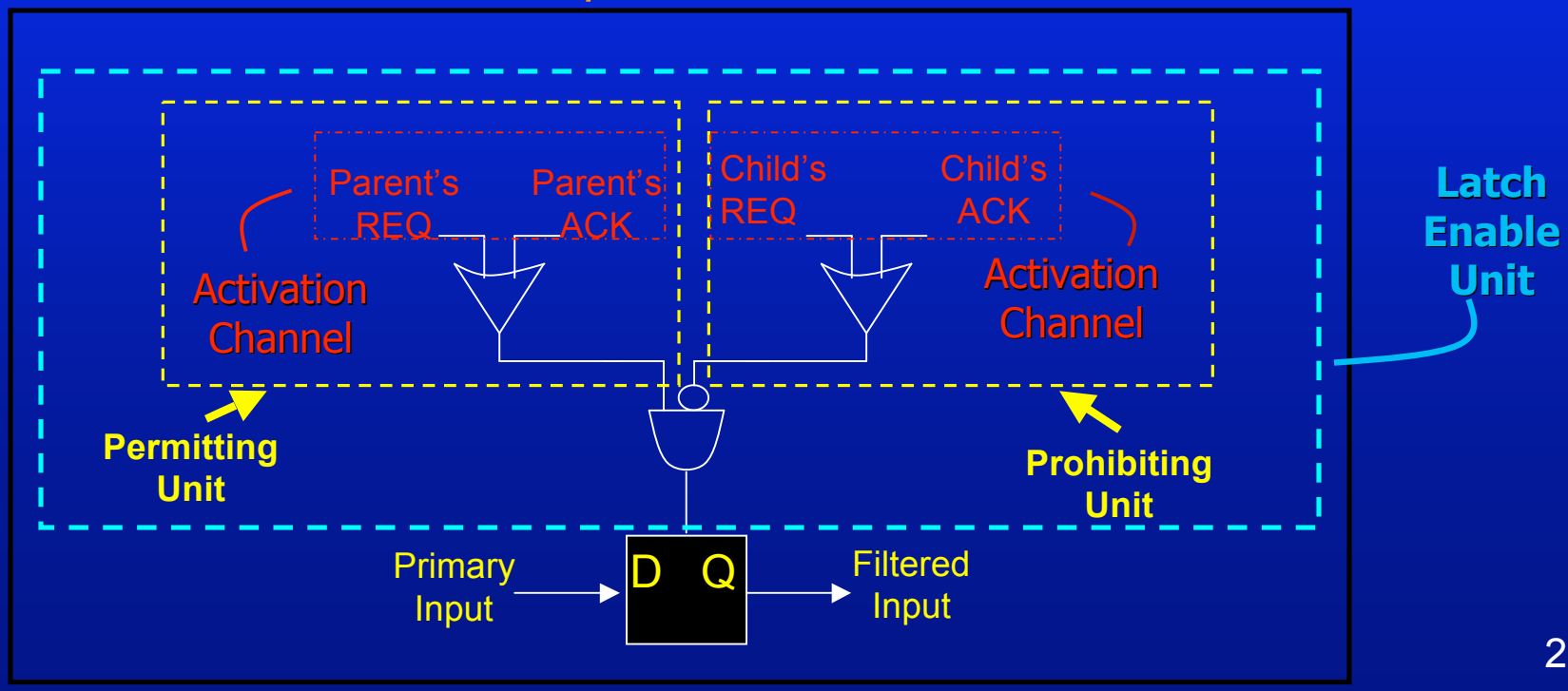
Decomposed Controllers: Input Latch Hardware

"Controller as a Parent"

Idea:

- Latch permitting and prohibiting unit used to control input latch
- "Parent" gets latch prohibited when control passed to child
- Latch re-enabled when child completes allowing inputs to be permitted

Generic Input Latch Structure

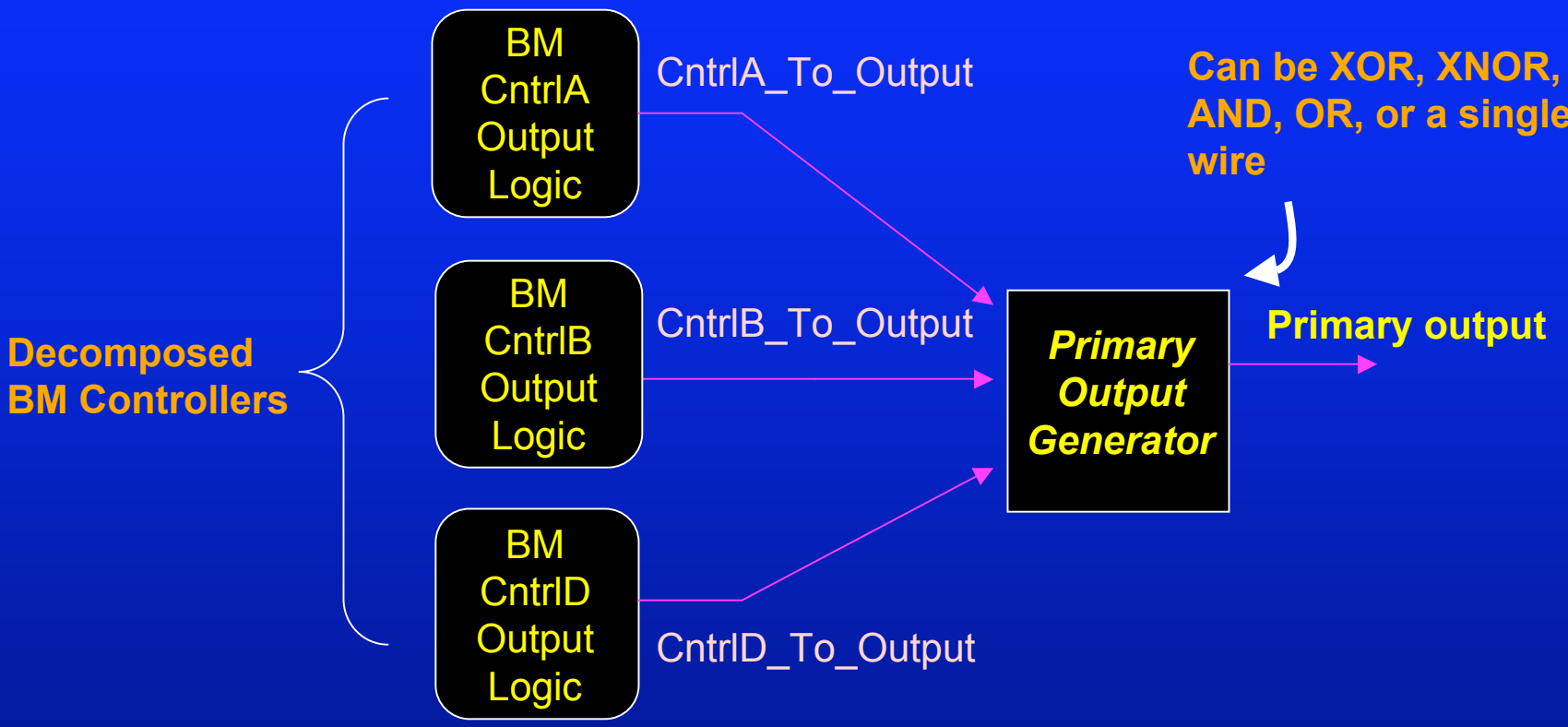


Overview of Hardware Imp.: Output Generator

- Primary Output Generators
  - Needed for generating and producing correct output values
  - Created for each primary output
    - Inputs
      - A signal from each controller that asserts the corresponding output
    - Outputs
      - Produced by a single XOR or XNOR gate
      - "Reduction-in-Strength" optimizations are applied so the logic gate can be AND, OR, NOR, or Single Wire

Overview of Hardware Imp.: Output Generator

Block View



Output generator is determined by the initial output value of the decomposed controllers and the original BM controller.

PART I: Technical Overview

- Quick Review: Burst-Mode Controllers
- Decomposition Method
  - Micro-Architecture
  - Communication Protocol
  - Hardware Implementation
- bm_decomp: Basic Tool Features ←

bm_decomp: Basic Tool Features

- Basic Mode
  - Input:
    1. An Original BM Controller *(one .bms file)*
       - A single BM controller specification
  - Output:
    1. Decomposed Controllers *(multiple .bms files)*
       - Multiple decomposed BM specifications
    2. Auxiliary Hardware Details *(one .auxhw file)*
       - List for each controller:
         - All inputs (primary inputs + communication channels)
         - Latch enable logic for each primary input
         - Optimizations for primary input latches
       - List for each primary output generator
         - Implementation details (using reduction in strength)

... *once completed, user can synthesize decomposed specs with Minimalist*

bm_decomp: Basic Tool Features

- Extended Mode
  - First: runs bm_decomp (i.e., as in basic mode)
  - Second: automatically calls Minimalist to synthesize decomposed specs
    - Provides an interactive menu which allows users to synthesize resulting decomposed specs automatically using Minimalist
- Additional Feature
  - Help Menu
    - Includes basic help menu which provides assistance on using the tool

PART II: Tutorial

Design Examples + Hands-On Tutorial

- Example 1: HP-IR
  - A. Marshall, B. Coates, and P. Siegel, *"Designing an Asynchronous Communications Chip",* IEEE Design & Test of Computers, vol. 11:2, pp. 8-21 (1994).
- Example 2: RF-Control
  - A. Marshall, B. Coates, and P. Siegel. "Designing an Asynchronous Communications chip". *IEEE Design and Test of Computers, 11(2):8.* 21, 1994.
- Example 3: PSCSI-TSEND
  - S. M. Nowick, K. Y. Yun, and D. L. Dill. "*Practical Asynchronous Controller Design.*" IEEE International Conference on Computer Design (Oct. 1992).

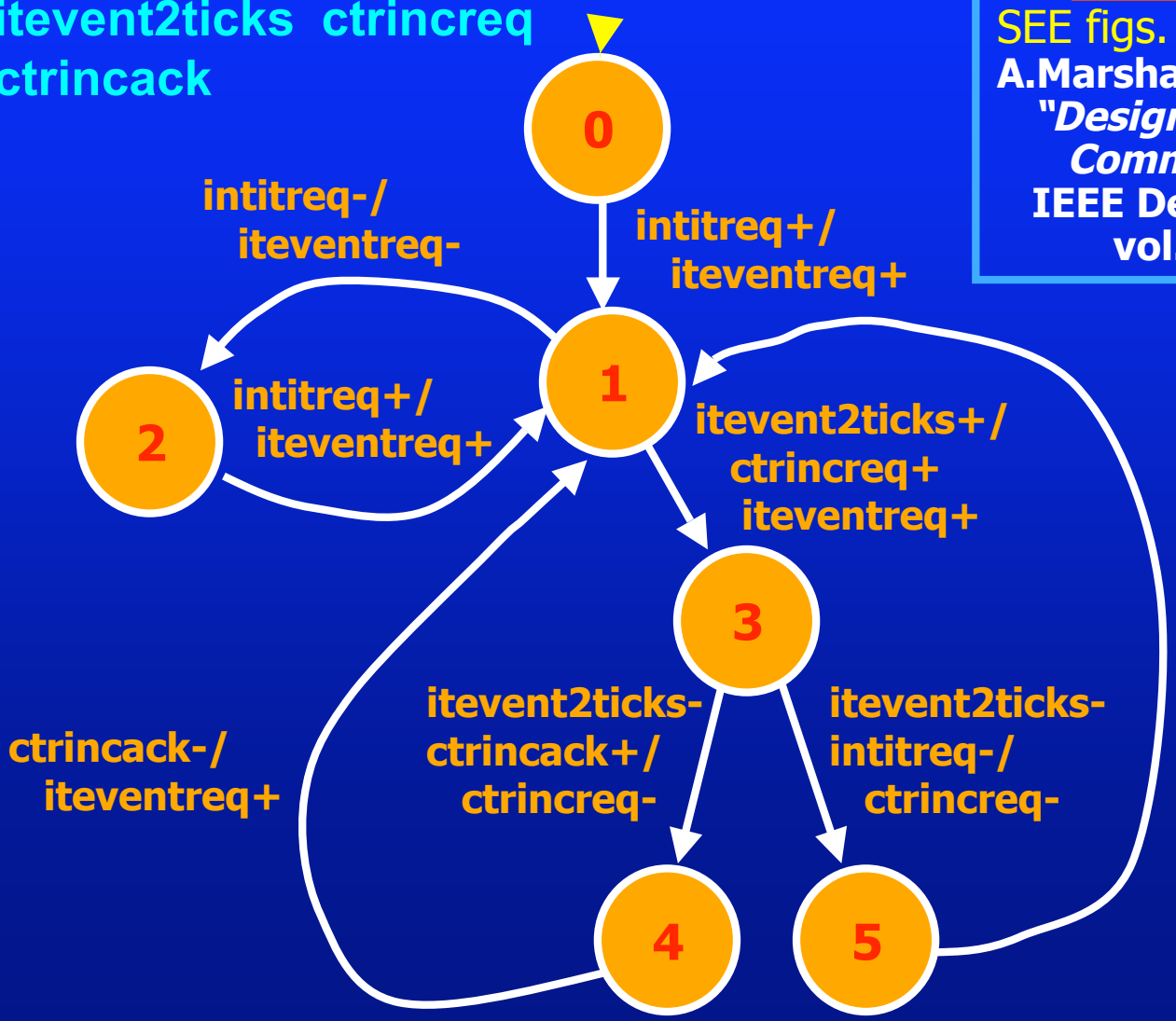
Example #1: "HP-IR" (HP Labs)

Inputs:
intitreq
itevent2ticks
ctrincack

Outputs:
iteventreq
ctrincreq

From HP Labs/Stanford "Stetson" Project:

SEE figs. 10, 11, pp. 17-18:
A.Marshall, B.Coates, P.Siegel, *"Designing an Asynchronous Communications Chip"*, IEEE Design&Test of Computers, vol. 11:2, pp. 8-21 (1994)



Initial values: in state #0, all inputs & outputs are 0

Example #1: "HP-IR" (HP Labs)

Running BM_DECOMP: the Simple Approach

Step #0. Getting Started ...

(a) go to the root directory ('bm_decomp') of the bm_decomp tool; then create a new subdirectory:

> mkdir test-demo

(b) go to it, then create a new subdirectory, and go to it:

> cd test-demo
> mkdir ex1
> cd ex1

(c) copy BM spec into the 'test-demo/ex1' subdirectory:

> cp ../../tutorial/HP-IR/hp-ir.bms .

Example #1: "HP-IR" (HP Labs)

Running BM_DECOMP: the Simple Approach

Step #1. Show BM Specification

Look at "BMS" text file:

>more hp-ir.bms

```
...
; NAME hp-IR
Input     IntITReq          0
Input     ITEvent2Ticks     0
Input     CtrIncAck         0
Output    ITEventReq        0
Output    CtrIncReq         0

0  1    IntITReq+   |   ITEventReq+
1  2    IntITReq-   |   ITEventReq-
2  1    IntITReq+   |   ITEventReq+
1  3    ITEvent2Ticks+   |   CtrIncReq+ ITEventReq-
3  4    ITEvent2Ticks- CtrIncAck+   | CtrIncReq-
3  5    ITEvent2Ticks- IntITReq-    |  CtrIncReq-
4  1    CtrIncAck-   |  ITEventReq+
5  1    IntITReq+   | ITEventReq+
```

Example #1: "HP-IR" (HP Labs)

Running BM_DECOMP: the Simple Approach

Step #2a. Display Help Menu:

> bm_decomp help

NOTE: This demo assumes you have set up your PATH variable so that you can call 'bm_decomp' directly. *(See README file)*

Step #2b. Run BM_DECOMP:

> bm_decomp hp-ir.bms

Step #2c. Skip automatic synthesis of decomposed controllers using Minimalist *(just decompose the original spec)*:

When prompted:

"Would you like to synthesize your bm_decomp results with Minimalist(Y/N)?"

> N

Example #1: "HP-IR" (HP Labs)

Running BM_DECOMP: the Simple Approach

Step #3. Display It:

(a) Text: Results Summary

> [see displayed text output]

(b) decomposed BM Specs

> more hp-ir_Master.bms [for top-level Master]

> more hp-ir_MachineX.bms [where X is a number]

(c) Auxiliary Hardware Details

> more hp-ir.auxhw

Example #1: "HP-IR" (HP Labs)

BM_DECOMP Results

3 BM Machines: 1 Top-Level Master & 2 Controllers

```
; name hp-ir_Master (Top-Level)

Input    IntITReq          0
Input    GlobalAckZero     0
Output   ITEventReq        0
Output   ReqZero           0

0  1    IntITReq+    MasterITEventReq+
        ReqZero+
1  2    GlobalAckZero+   |   ReqZero-
2  1    GlobalAckZero-   |   ReqZero+
```

```
; name hp-ir_Machine1 (Leaf)

    Input    IntITReq          1
    Output   ITEventReq        1
    Output   OneToZero         0

1  2   IntITReq-   | OntToITEventReq- OneToZero+
2  1   IntITReq+  | OnteToITEventReq+
       OneToZero-
```

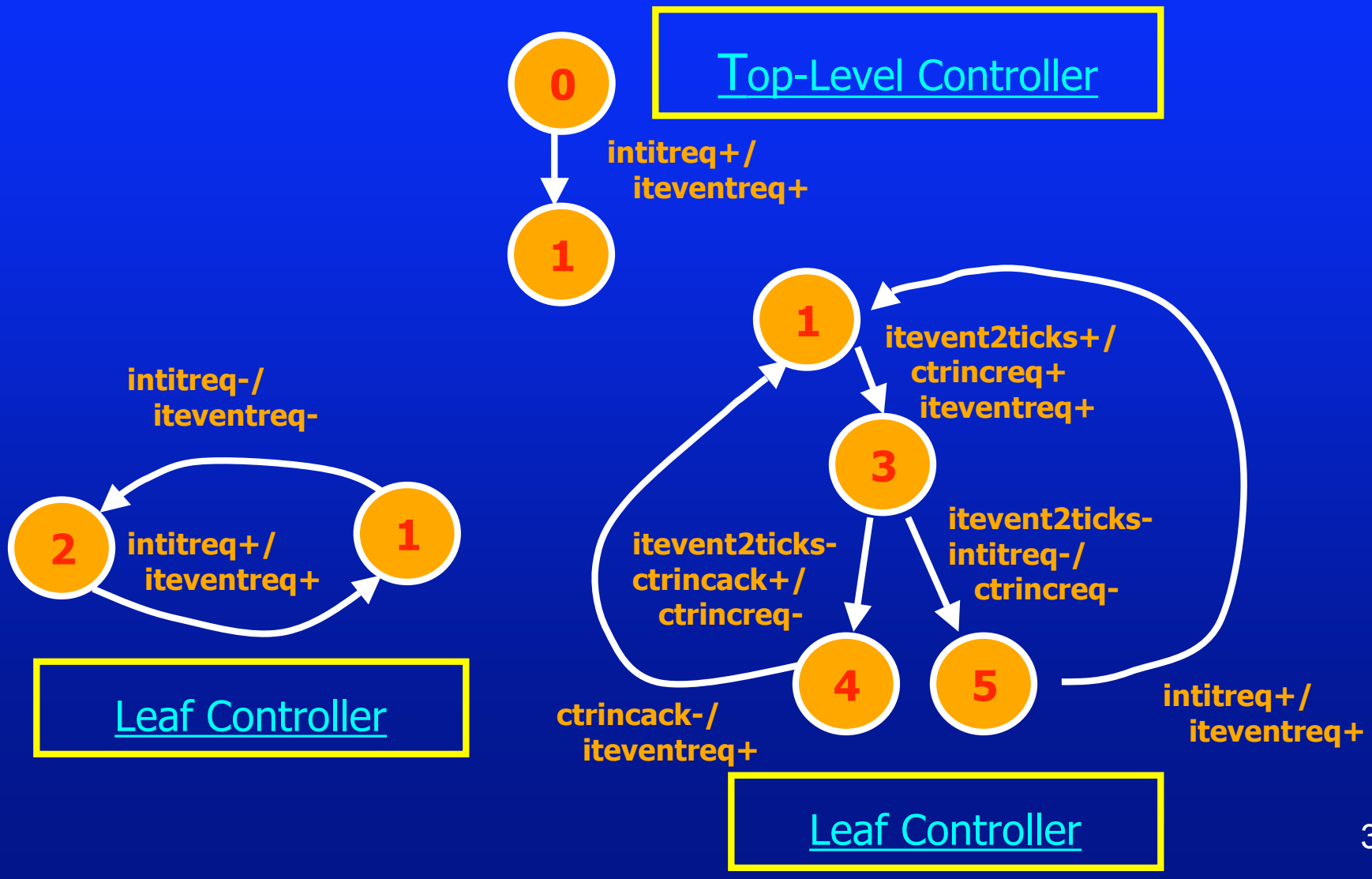
```
; name hp-ir_Machine2 (Leaf)

    Input    ITevent2Ticks     0
    Input    IntITReq          1
    Input    CtrIncAck         0
    Output   CtrIncReq         0
    Output   ITEventReq        1
    Output   TwoToZero         0

1  3   ITEvent2Ticks+  | TwoToCtrIncReq+
                         TwoToZero+
                         TwoToITEventReq-
3  5   ITEvent2Ticks- IntITReq-   | TwoToCtrIncReq-
5  1   IntITReq+     |  ITEventReq+ TwoToZero-
3  4   ITEvent2Ticks- CtrIncAck+ | TwoToCtrIncReq-
4  1   CtrIncAck-  | TwoToITEventReq+ TwoToZero-
```

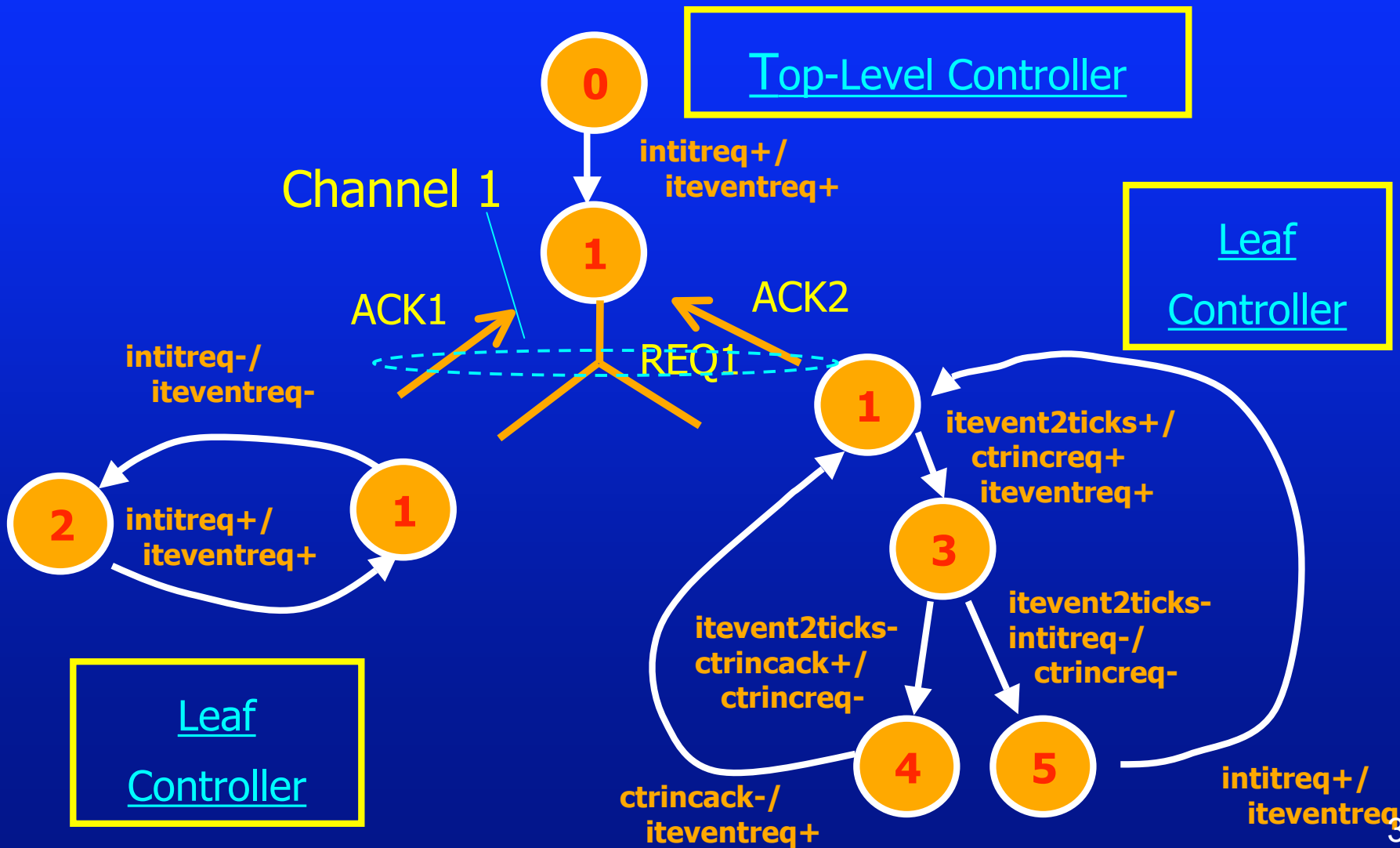
Example #1: "HP-IR" (HP Labs)

3 BM Machines: 1 Top-Level Master & 2 Leaf Controllers



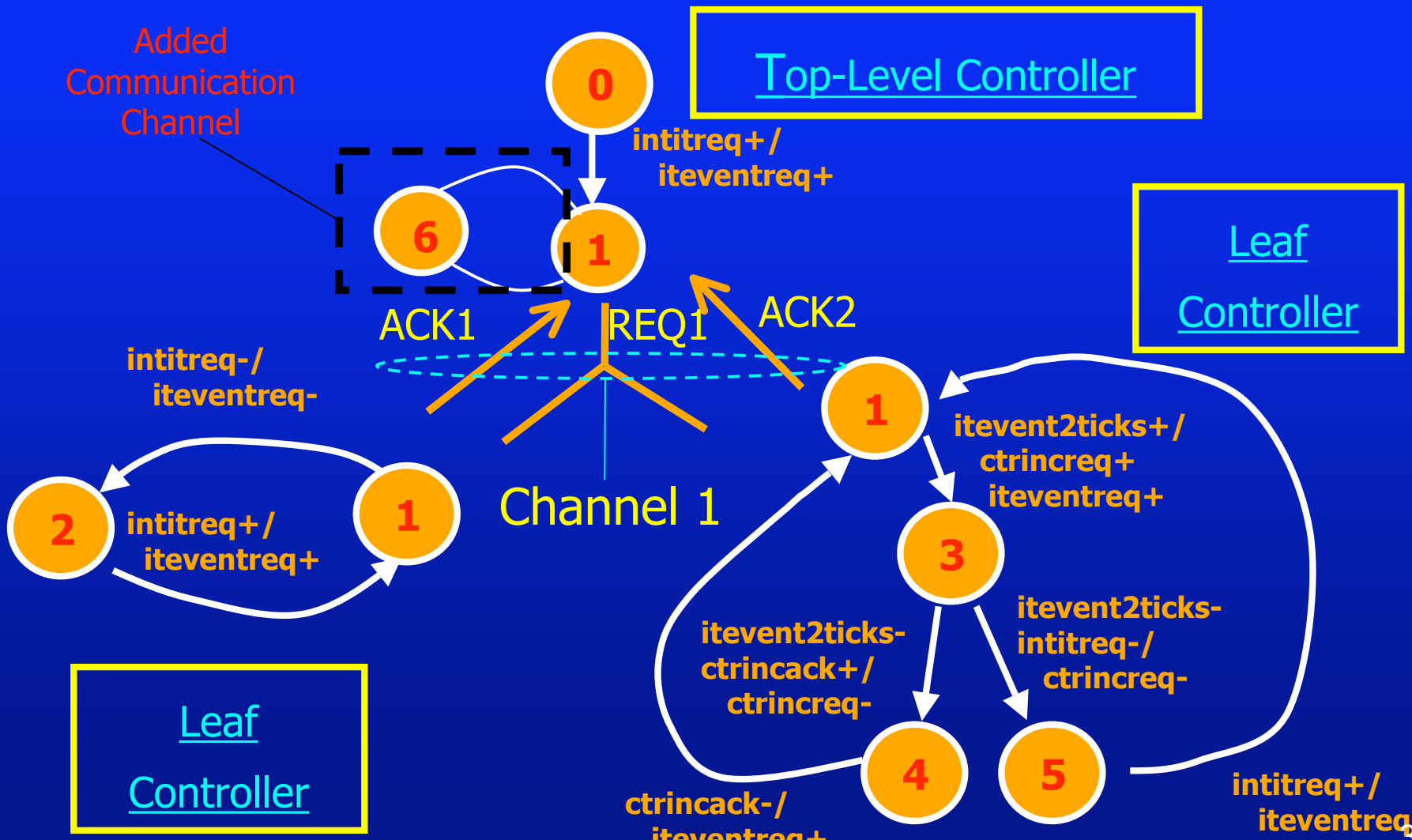
Example #1: "HP-IR" (HP Labs)

1 Symbolic Communication Channel (between parent and leaf controllers)



Example #1: "HP-IR" (HP Labs)

Additional Communication Transitions (for symbolic channel)



Example #1: "HP-IR" (hp-ir.auxhw)

Auxiliary File Generated by BM_DECOMP (BM Machines)

The results of bm_decomp are as follows:

Original BM Controller:	hp-ir.bms
# of Decomp. BM Controllers:	3
Names of Decomp. BM Controllers:	hp-ir_Master.bms hp-ir_Machine1.bms hp-ir_Machine2.bms

[text omitted:…See hp-ir.auxhw for key on interpreting the output]

Master Machine

Primary Input 1: IntITReq

LATCH REQUIRED: YES

LATCH ENABLE LOGIC:

Permitting Unit: Does not exist

Prohibiting Unit: ReqZero' AckOneToZero' AckThreeToZero'

Master_enable = ReqZero' AckOneToZero' AckThreeToZero'

Example #1: "HP-IR" (hp-ir.auxhw)

Auxiliary File Generated by BM_DECOMP (BM Machines)

Machine 1

Primary Input 1: IntITReq

LATCH REQUIRED: YES

LATCH ENABLE LOGIC:

Permitting Unit: (AckOneToZero + ReqZero)

Prohibiting Unit: Does not exist

Machine1_enable = (AckOneToZero + ReqZero)

Machine 2

Primary Input 1: IntITReq

LATCH REQUIRED : YES

Permitting Unit: AckTwoToZero

Prohibiting Unit: Does not exist

Machine2_enable = AckTwoToZero

Primary Input 2: ITEvent2Ticks

LATCH REQUIRED: NO

Latch removed by optimization

Example #1: "HP-IR" (hp-ir.auxhw)

Auxiliary File Generated by BM_DECOMP (BM Machines)

Machine 2 (cont'd)

Primary Input 3: CtrIncAck
LATCH REQUIRED: NO
*Latch removed by optimization

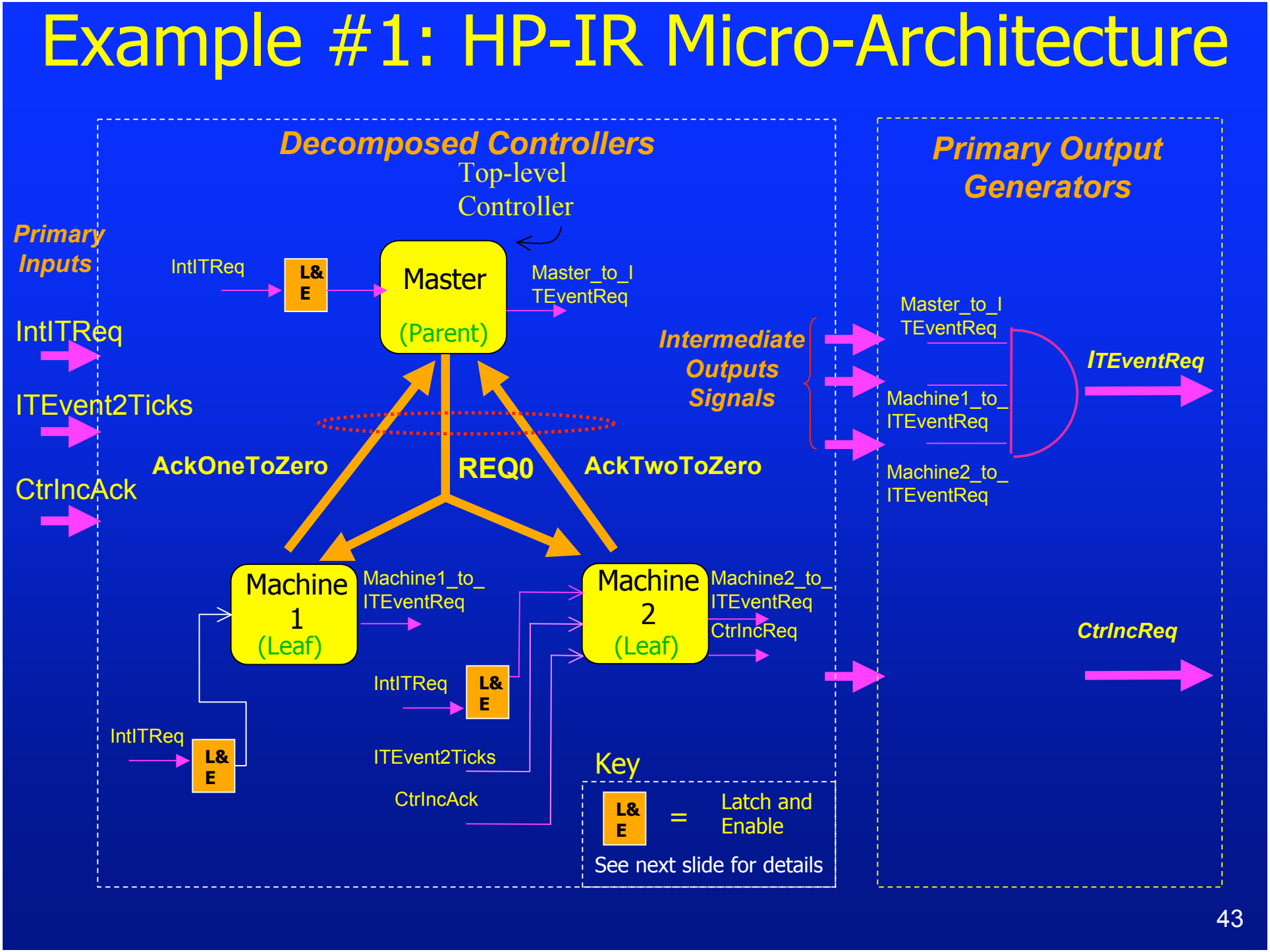
Example #1: "HP-IR" (hp-ir.auxhw)

Auxiliary File Generated by BM_DECOMP (Output Generators)

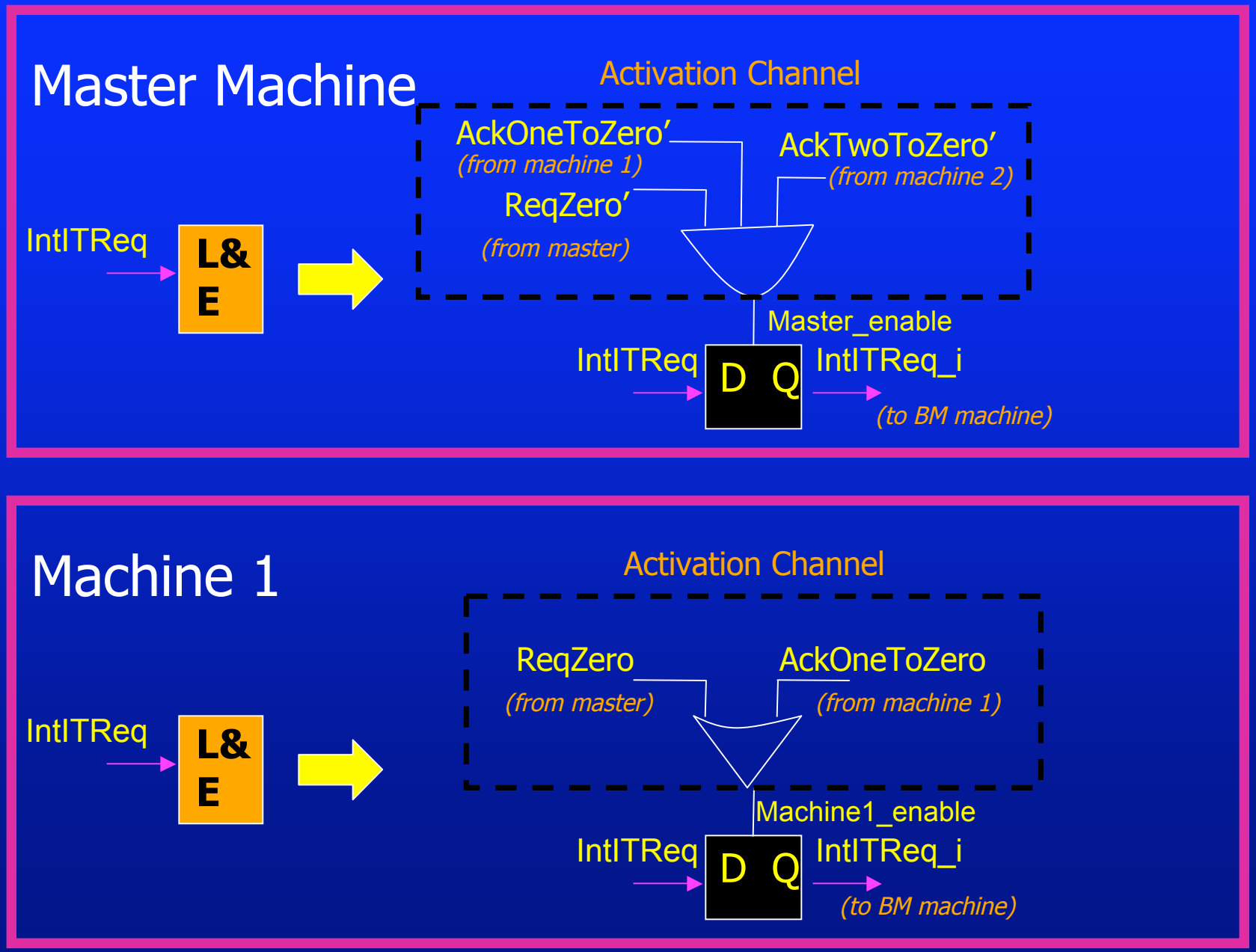
Primary Output Generators

- Primary Output 1 = ITEventReq
  - Generator : AND3 gate (Inputs from Machines Master, 1, 2)
  - Inputs: MasterToITEventReq, OneToITEventReq, TwoToITEventReq
- Primary Output 2 = CtrIncReq
  - Generator : Single Wire (Input from Machine 2)
  - Inputs: TwoToCtrIncReq

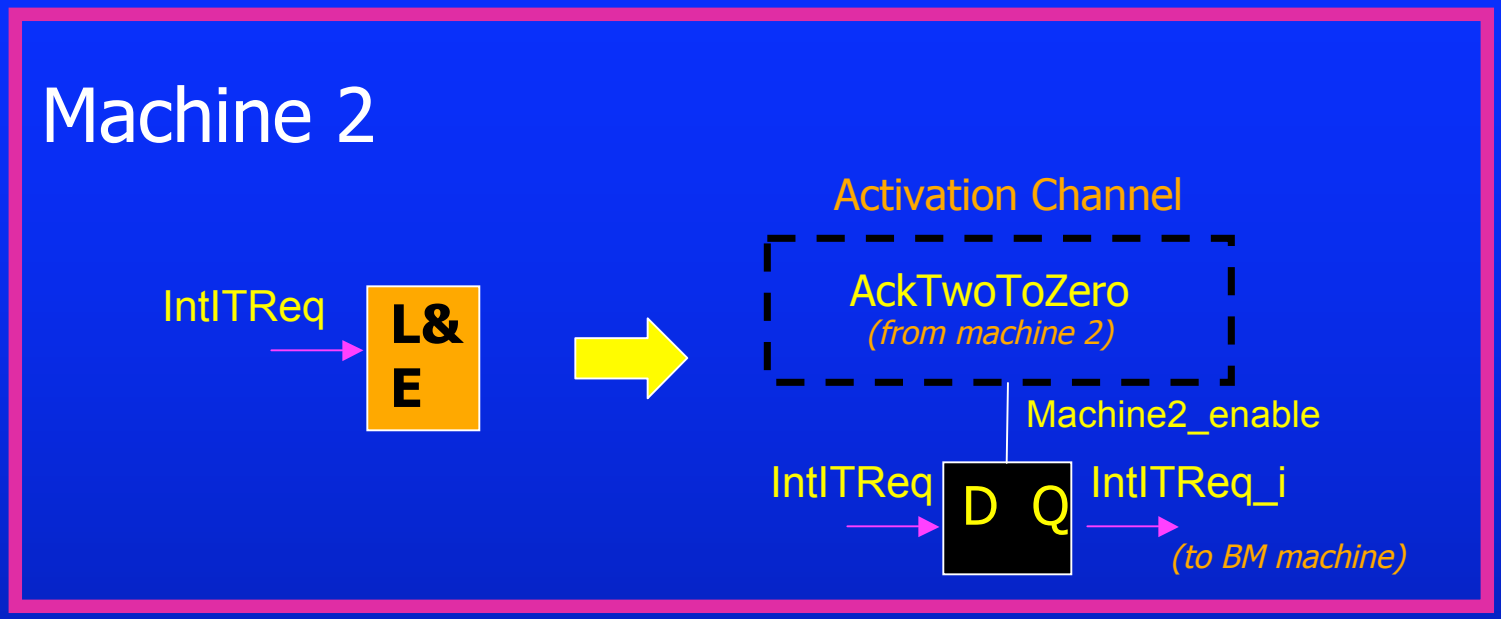
Example #1: HP-IR Micro-Architecture



Example #1: HP-IR Latch and Enable Details

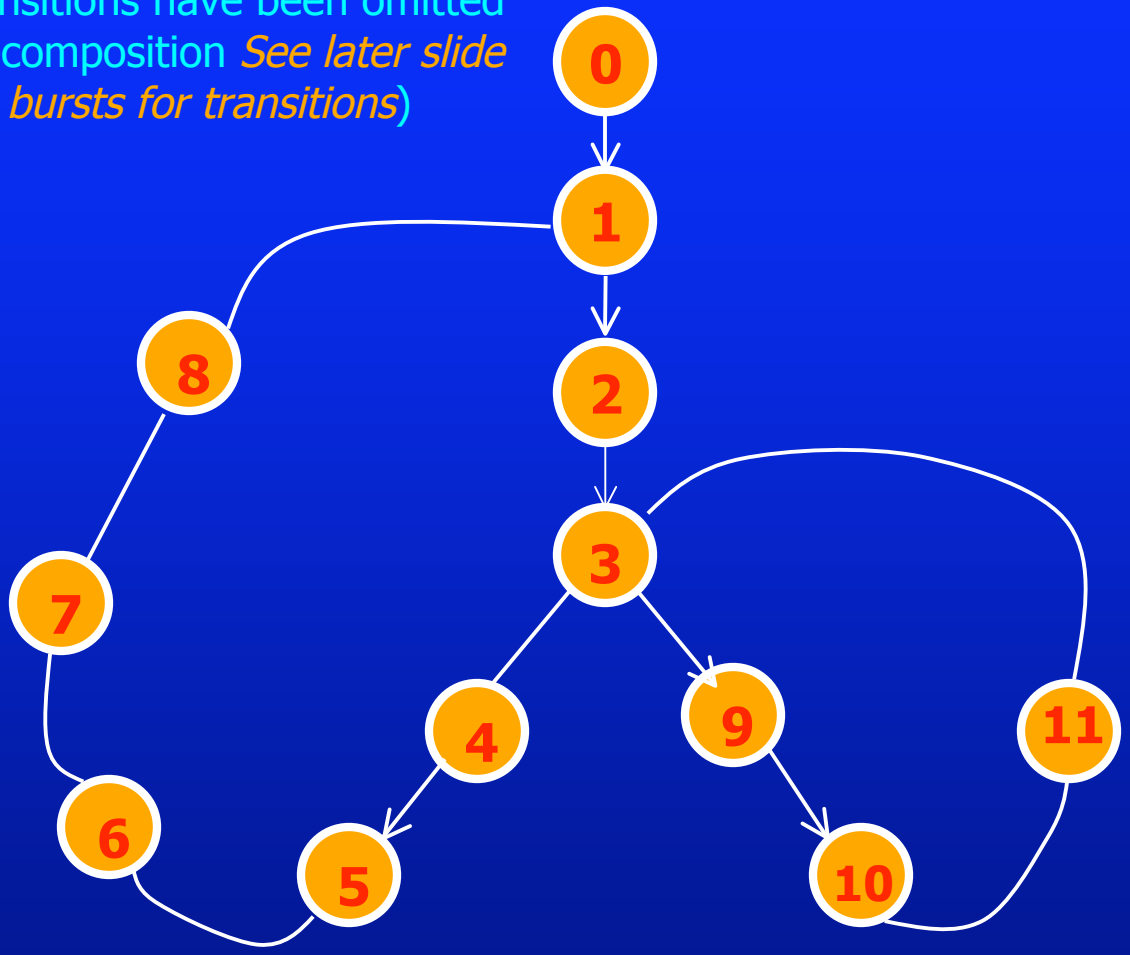


Example #1: HP-IR Latch and Enable Details



Example #2: RF-Control

(input and output transitions have been omitted to show structural decomposition *See later slide for input/output bursts for transitions*)



Example #2: RF-Control

Running BM_DECOMP

Step #0. Getting Started ...

(a) go back into "bm_decomp/test-demo" directory:

> cd ..

(b) create a new subdirectory, and go to it:

> mkdir ex2

> cd ex2

(c) copy BM spec into the 'test-demo/ex2' subdirectory:

> cp ../../tutorial/RF-CONTROL/rf-control.bms .

Example #2: RF-Control

Step #1. Show BM Specification

(a) Look at "BMS" text file:

>more rf-control.bms

```
name RF_control
Input       RFFrameReq          0
Input       SOFEventOK          0
Input       EOFEventOK          0
Input       CtrEoTSAck          0
Input       SCEotSAck           0
Input       HIFCommitAck        0

Output      ControlResetAck     0
Output      RFFrameAck          0
Output      IntSDReq            0
Output      SCEoTSReq           0
Output      HIFCommitReq        0

 [… continued on right
       column ==> ]
```

```
0  1      RFFrameReq+ | IntSDReq+ ControlResetAck-
1  2      SOFEventOK+ | IntSDReq-
2  3      SOFEventOK-  | IntSDReq+
3  4      EOFEventOK+ | HIFCommitReq+ IntSDReq-
4  5      EOFEventOK- HIFCommitAck+ |
                    HIFCommitReq+    SCEoTSReq+
5  6      HIFCommitAck- SCEoTSAck+ | SCEoTSReq-
6  7      SCEoTSAck- | RFFrameAck+
7  8      RFFrameReq- | RFFrameAck-
8  1      RFFrameReq+ | IntSDReq+
3  9      CtrEoTSAck+ | IntSDReq-
9  10     CtrEoTSAck- | SCEoTSReq+
10 11     SCEoTSAck+ | SCEoTSReq-
11 3      SCEoTSAck- | IntSDReq+
```

Example #2: RF-Control

Running BM_DECOMP

Step #2a. Run BM_DECOMP:

> bm_decomp rf-control.bms

Step #2b. Automatically synthesize specs using Minimalist-basic:

(a). When prompted:

"Would you like to synthesize your bm_decomp results with Minimalist(Y/N)?"

> Y

(b). When prompted:

"Please select a mode to synthesize the decomposed specs:"

> 1

NOTE: See menu of options when running the tool; option 1 means running "Minimalist-basic" script

Example #2: RF-Control

Running BM_DECOMP

Step #3. Display It:

(a) Text: Results Summary

> [see displayed text output]

(b) decomposed BM Specs

> more rf-control_Master.bms [for top-level Master]

> more rf-control_MachineX.bms [where X is a number]

(c) Auxiliary Hardware Details

> more rf-control.auxhw

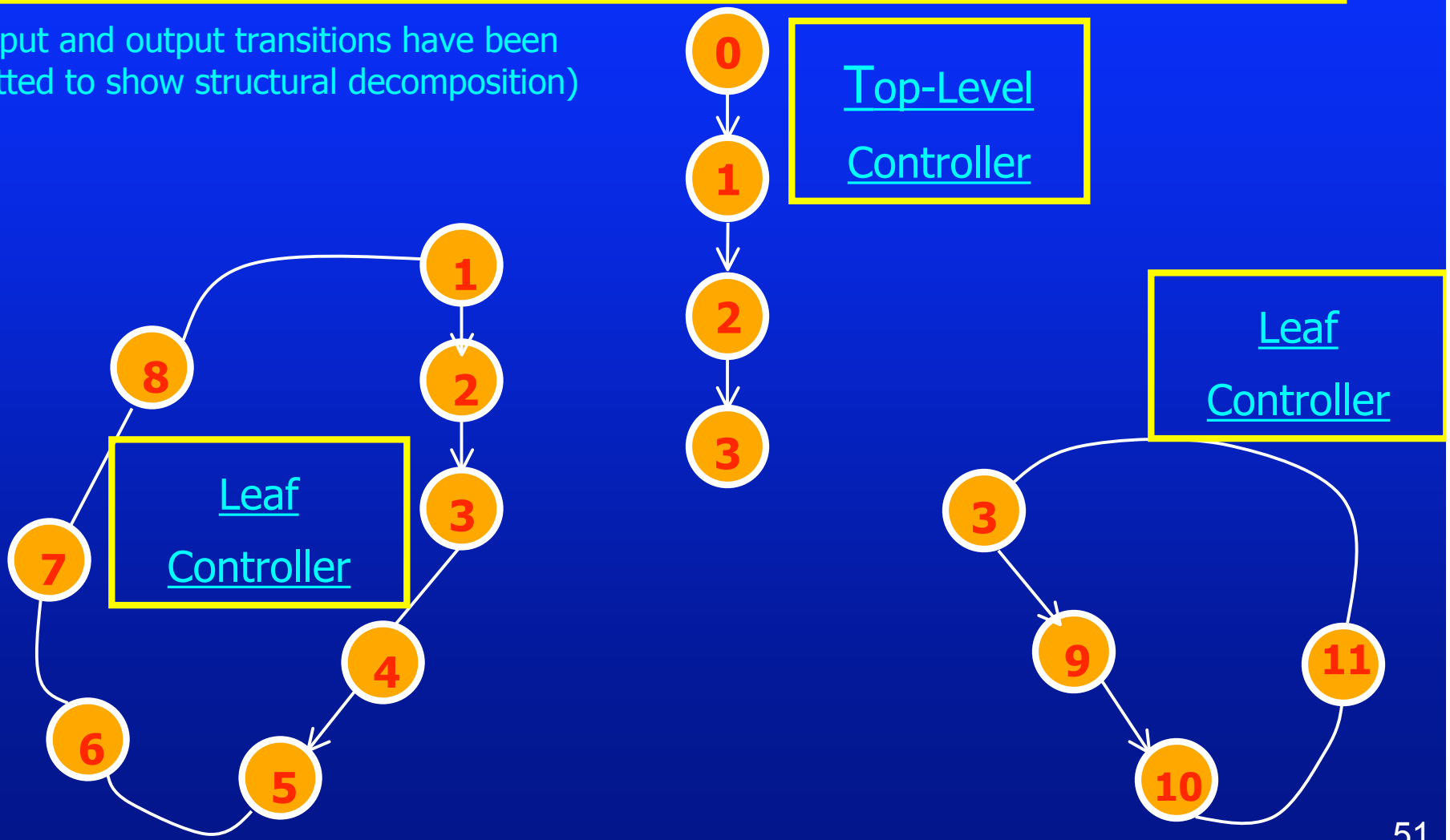
When done: compare your results with specs found in…

…/bm_decomp/tutorial/RF-CONTROL/results/

Example #2: RF-Control

3 BM Machines: 1 Top-Level Master & 2 Leaf Controllers

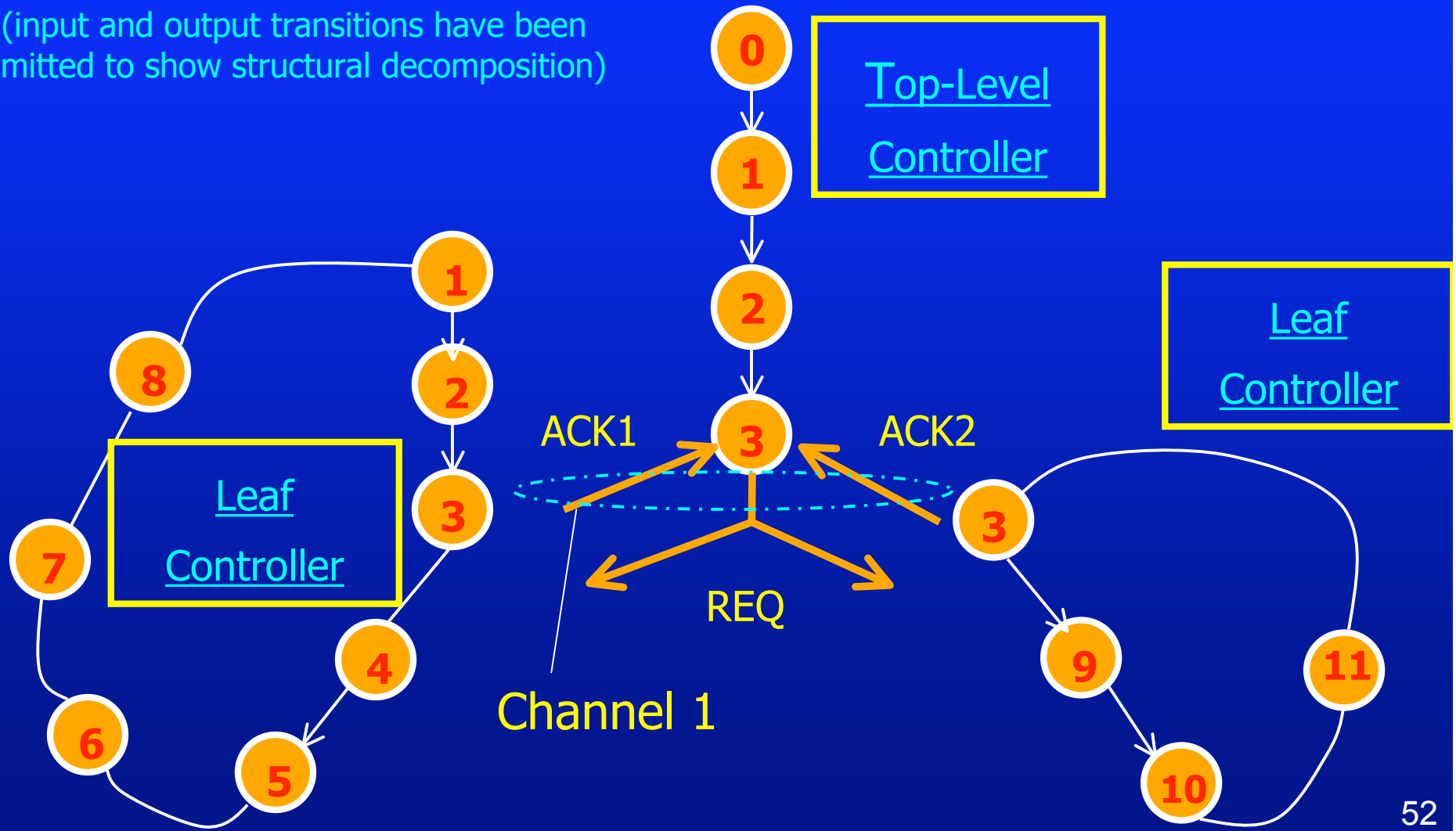
(input and output transitions have been omitted to show structural decomposition)



Example #2: RF-Control

1 Symbolic Communication Channel (between parent and leaf controllers)

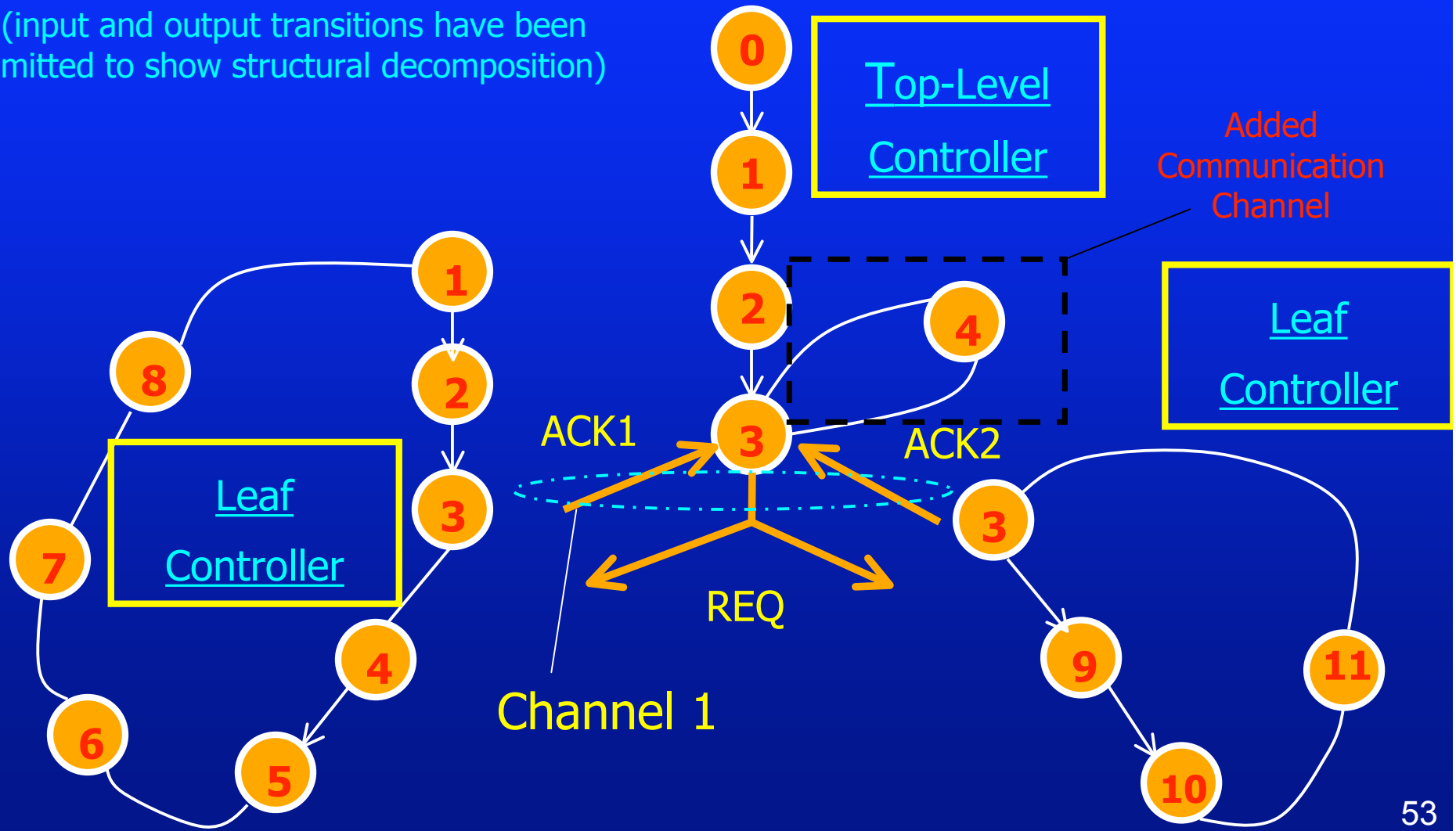
(input and output transitions have been omitted to show structural decomposition)



Example #2: RF-Control

Additional Communication Transitions (for symbolic channel)

(input and output transitions have been omitted to show structural decomposition)



Example #2: RF-Control (rf-control.auxhw)

Auxiliary File Generated by BM_DECOMP (BM Machines)

The results of bm_decomp are as follows:

Original BM Controller: rf-control.bms

# of Decomp. BM Controllers: 3

Names of Decomp. BM Controllers: rf-control_Master.bms
rf-control_Machine1.bms
rf-control_Machine2.bms

Master Machine

Primary Input 1: RFFrameReq

LATCH REQUIRED: YES

LATCH ENABLE LOGIC:

Permitting Unit: Does not exist

Prohibiting Unit: ReqZero' AckOneToZero' AckTwoToZero'

Master_enable = ReqZero' AckOneToZero' AckTwoToZero`

Example #2: RF-Control (rf-control.auxhw)

Auxiliary File Generated by BM_DECOMP (BM Machines)

Master Machine (cont'd)

Primary Input 2: SOFEventOK

LATCH REQUIRED: YES

LATCH ENABLE LOGIC:

Permitting Unit: Does not exist

Prohibiting Unit: ReqZero' AckOneToZero' AckTwoToZero'

Master_enable = ReqZero' AckOneToZero' AckTwoToZero`

Machine 1

Primary Input 1: RFFrameReq

LATCH REQUIRED: YES

LATCH ENABLE LOGIC:

Permitting Unit: AckOneToZero

Prohibiting Unit: Does not exist

Machine1_enable = AckOneToZero

Example #2: RF-Control (rf-control.auxhw)

Auxiliary File Generated by BM_DECOMP (BM Machines)

Machine 1 (cont'd)

Primary Input 2: SOFEventOK

LATCH REQUIRED: YES

LATCH ENABLE LOGIC:

Permitting Unit: AckOneToZero

Prohibiting Unit: Does not exist

Machine1_enable = AckOneToZero

Primary Input 3: EOFEventOK

LATCH REQUIRED: NO

LATCH REMOVED BY OPTIMIZATION

Primary Input 4: SCEoTSAck

LATCH REQUIRED: YES

LATCH ENABLE LOGIC:

Permitting Unit: AckOneToZero

Prohibiting Unit: Does not exist

Machine1_enable = AckOneToZero

Example #2: RF-Control (rf-control.auxhw)

Auxiliary File Generated by BM_DECOMP (BM Machines)

Machine 1 (cont'd)

Primary Input 5: HIFCommitAck

LATCH REQUIRED: NO

Latch removed by optimization

Machine 2

Primary Input 1: CtrEoTSAck

LATCH REQUIRED: NO

Latch removed by optimization

Primary Input 2: SCEoTSAck

LATCH REQUIRED: YES

Permitting Unit: AckTwoToZero

Prohibiting Unit: Does not exist

Machine2_enable = AckTwoToZero

Example #2: RF-Control (rf-control.auxhw)

Auxiliary File Generated by BM_DECOMP (Output Generators)

Primary Output Generators

- Primary Output 1 = ITEventReq
  - Generator: Single Wire (Input from Machine Master)
  - Inputs: MasterToControlResetAck
- Primary Output 2 = RFFrameAck
  - Generator : Single Wire (Input from Machine 1)
  - Inputs: OneToRFFrameAck
- Primary Output 3 = IntSDReq
  - Generator : AND3 gate (Inputs from Machines Master, 1, 2)
  - Inputs: MasterToIntSDReq, OneToIntSDReq, TwoToIntSDReq

Example #2: RF-Control (rf-control.auxhw)

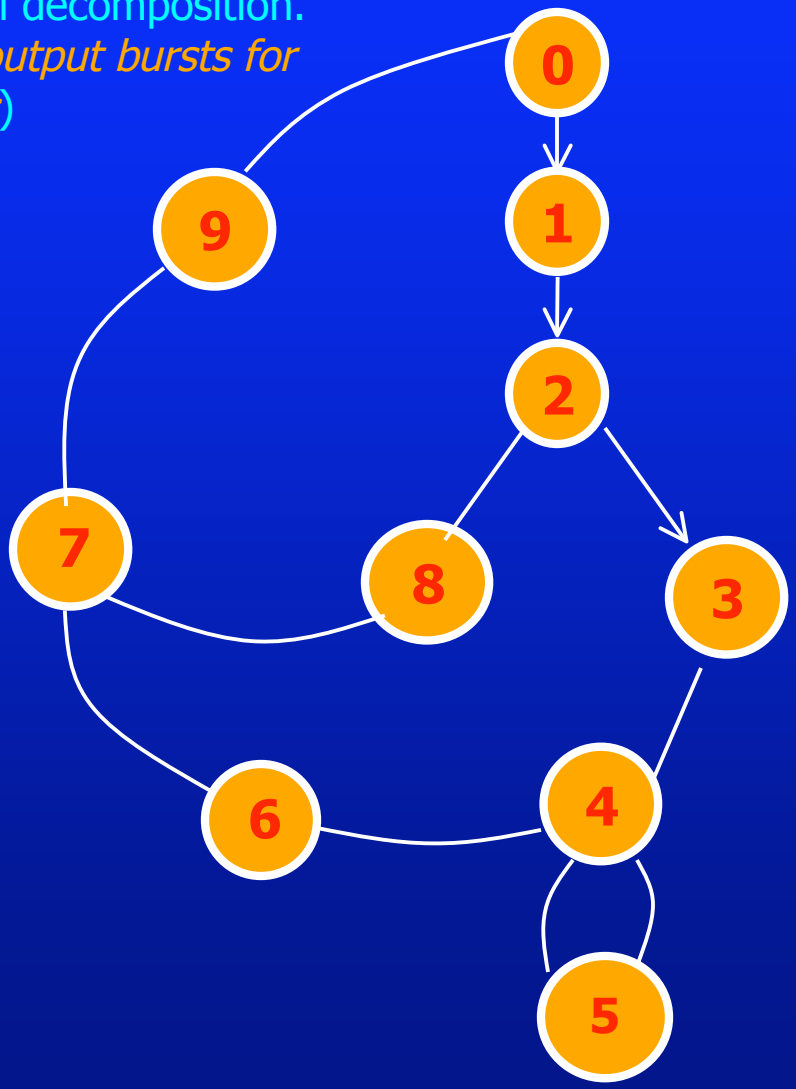
Auxiliary File Generated by BM_DECOMP (Output Generators)

Primary Output Generators

- Primary Output 4 = SCEoTSReq
  - Generator : OR2 gate (Inputs from Machines 1, 2)
  - Inputs: OneToSCEoTSReq, TwoToSCEoTSReq
- Primary Output 5 = HIFCommitReq
  - Generator : Single Wire (Input from Machine 1)
  - Inputs: OneToHIFCommitReq

Example #3: pscsi-tsend

(input and output transitions have been omitted to show structural decomposition. *See later slide for input/output bursts for transitions*)



Example #3: pscsi-tsend

Running BM_DECOMP

Step #0. Getting Started ...

(a) go back into "bm_decomp/test-demo" directory:

> cd ..

(b) create a new subdirectory, and go to it:

> mkdir ex3

> cd ex3

(c) copy BM spec into the 'test-demo/ex3' subdirectory:

> cp ../../tutorial/PSCSI-TSEND/pscsi-tsend.bms .

Example #3: pscsi-tsend

Running BM_DECOMP

Step #1. Show BM Specification

(a) Look at "BMS" text file:

>more pscsi-tsend.bms

```
; name pscsi-tsend

input   StartDMASend 0
input   DWAckNormN 1
input   DWAckLastN 1
input   AckInN 1

output  DRQ 0
output  ReqOutN 1
output  EndDMAInt 1
```

[… continued on right column ==>]

```
0  1   StartDMASend+   |  EndDMAInt+
1  2   StartDMASend-   |  DRQ+
2  3   DWAckNormN-     |  DRQ-
2  8   DWAckLastN-     |  DRQ-
3  4   DWAckNormN-     |   ReqOutN – DRQ+
4  6   DWAckLastN- AckInN- | ReqOutN+ DRQ-
4  5   DWAckNormN- AckInN- | ReqOutN+ DRQ-
5  4   DWAckNormN+ AckInN+ | ReqOutN-
     DRQ+
6  7   DWAckLastN+ AckInN+ | ReqOutN-
8  7   DWAckLastN+     |   ReqOutN-
7  9   AckInN-         |   ReqOutN+
9   0   AckInN+        |   EndDMAInt+
```

Example #3: pscsi-tsend

Running BM_DECOMP

Step #2a. Run BM_DECOMP:

> bm_decomp pscsi-tsend.bms

Step #2b. Synthesize specs using Minimalist-area with fedback

(a). When prompted:

"Would you like to synthesize your bm_decomp results with Minimalist(Y/N)?"

> Y

(b). When prompted:

"Please select a mode to synthesize the decomposed specs:"

> 4

(c). When prompted:

"Enter run-type -- multi-output or output-disjoint:"

> multi-output

Example #3: pscsi-tsend

Running BM_DECOMP

Step #3. Display It:

(a) Text: Results Summary

> [see displayed text output]

(b) decomposed BM Specs

> more pscsi-tsend_Master.bms [for top-level Master]

> more pscsi-tsend_MachineX.bms [where X is a number]

(c) Auxiliary Hardware Details

> more pscsi-tsend.auxhw

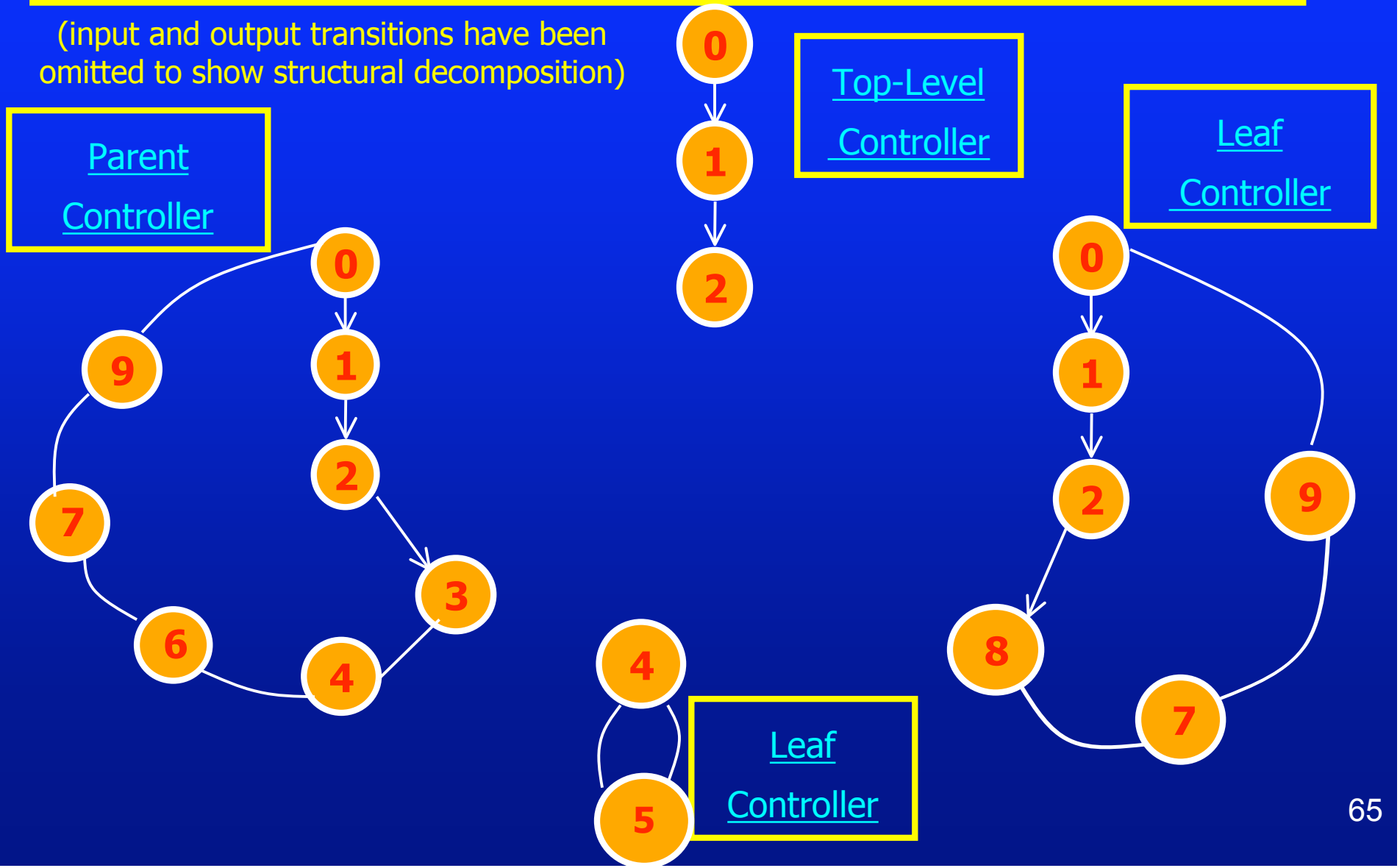
When done: compare your results with specs found in…

…/bm_decomp/tutorial/PSCSI-TSEND/results/

Example #3: pscsi-tsend

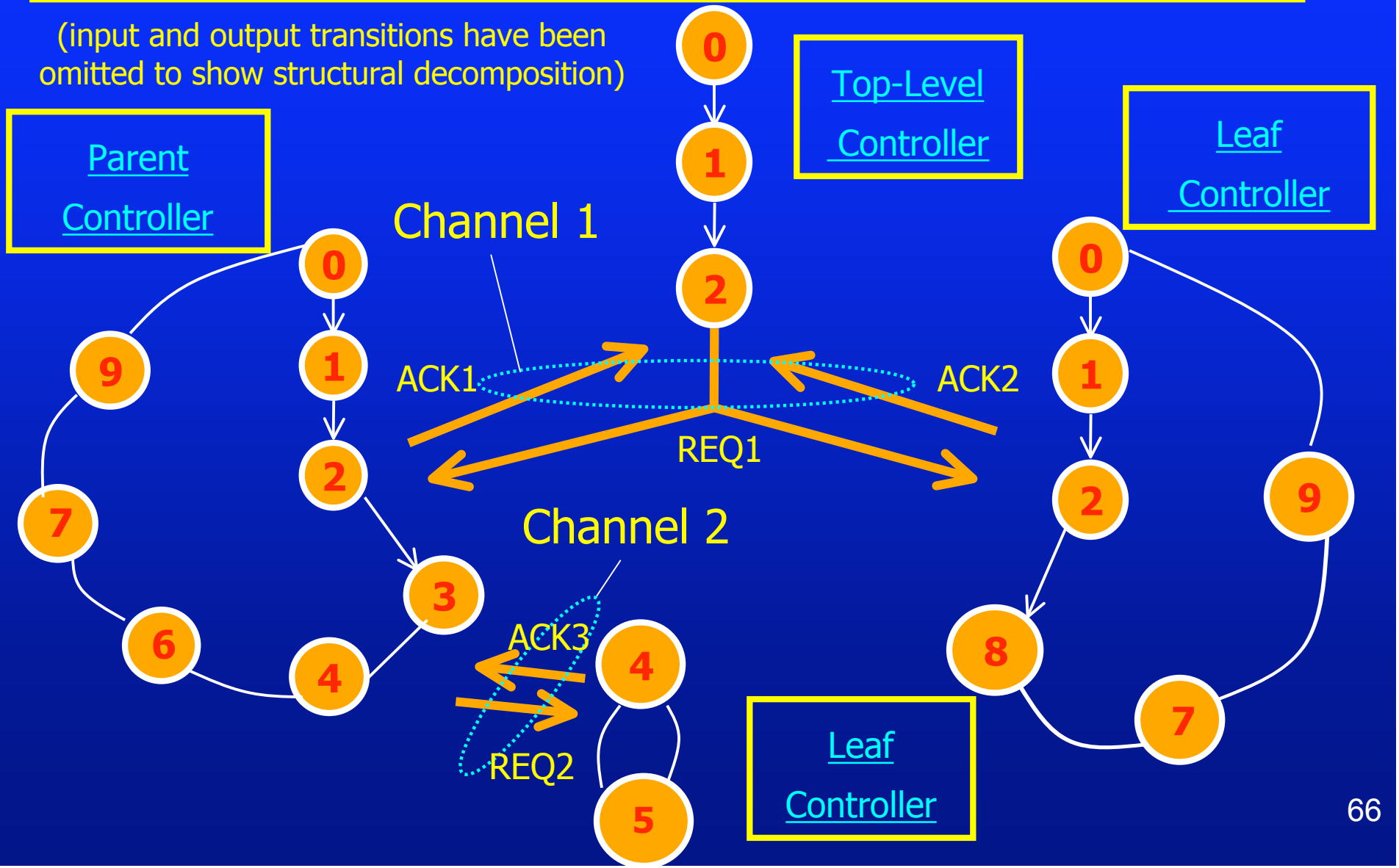
3 BM Machines: 1 Top-Level Master, 1 Parent, 2 Leaf Controllers

(input and output transitions have been omitted to show structural decomposition)



Example #3: pscsi-tsend

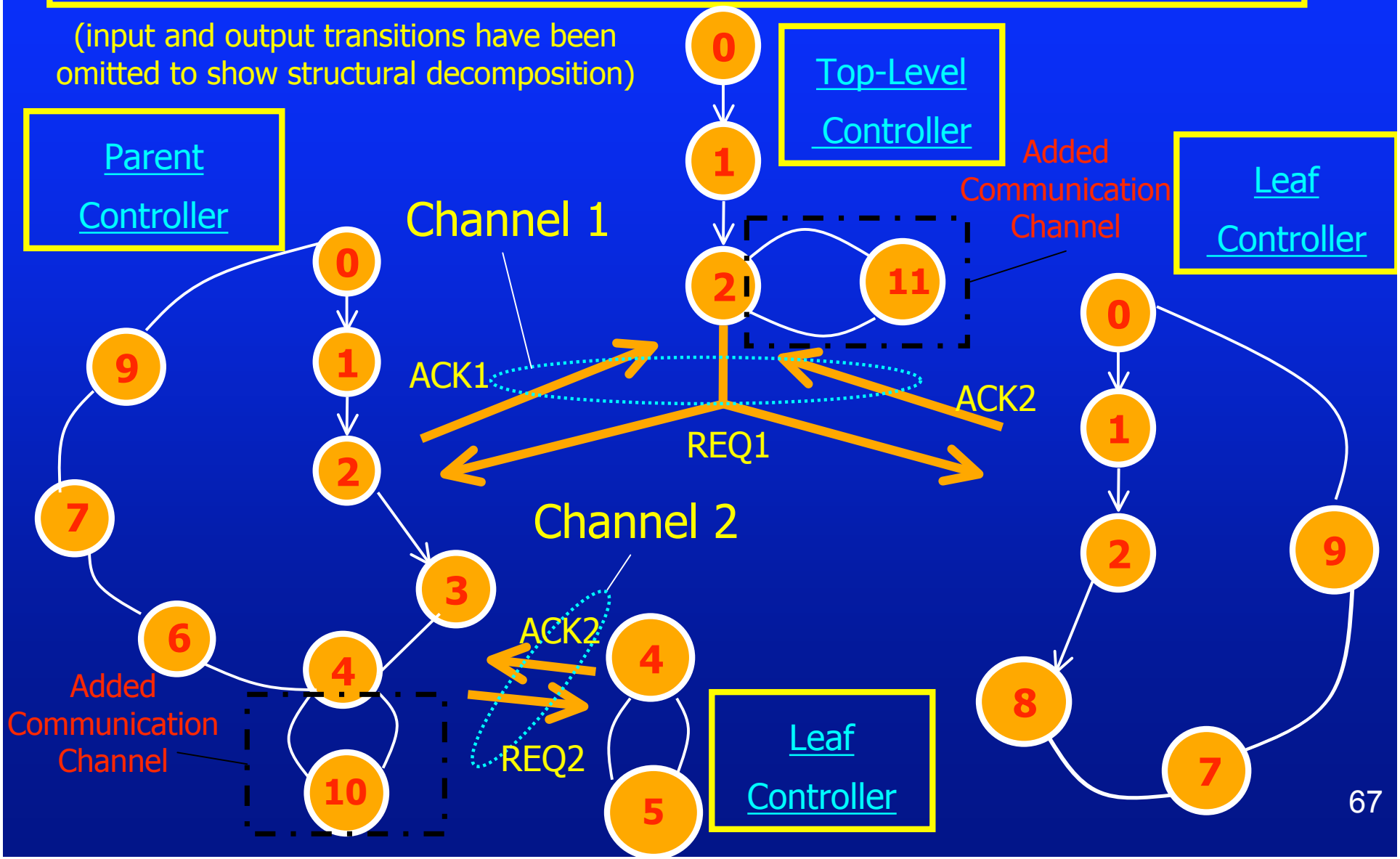
2 Symbolic Communication Channels (between parents & leaf controllers)



Example #3: pscsi-tsend

Additional Communication Transitions (for symbolic channels)

(input and output transitions have been omitted to show structural decomposition)



Example #3: pscsi-tsend (pscsi-tsend.auxhw)

Auxiliary File Generated by BM_DECOMP (BM Machines)

The results of bm_decomp are as follows:

Original BM Controller:	pscsi-tsend.bms
# of Decomp. BM Controllers:	4
Names of Decomp. BM Controllers:	pscsi-tsend_Master.bms
	pscsi-tsend_Machine1.bms
	pscsi-tsend_Machine2.bms
	pscsi-tsend_Machine3.bms

Master Machine

Primary Input 1: StartDMASend

LATCH REQUIRED: YES

LATCH ENABLE LOGIC:

Permitting Unit: Does not exist

Prohibiting Unit: ReqZero' AckOneToZero' AckThreeToZero'

Master_enable = ReqZero' AckOneToZero' AckThreeToZero'

Example #3: pscsi-tsend (pscsi-tsend.auxhw)

Auxiliary File Generated by BM_DECOMP (BM Machines)

Machine 1

Primary Input 1: StartDMASend

LATCH REQUIRED: YES

LATCH ENABLE LOGIC:

Permitting Unit: AckOneToZero

Prohibiting Unit: Does Not Exist

Machine1_enable = AckOneToZero

Primary Input 2: DWAckLastN

LATCH REQUIRED: YES

LATCH ENABLE LOGIC:

Permitting Unit: AckOneToZero + ReqZero

Prohibiting Unit: Does Not Exist

Machine1_enable = AckOneToZero + ReqZero

Primary Input 3: AckInN

LATCH REQUIRED: YES

LATCH ENABLE LOGIC:

Permitting Unit: AckOneToZero

Prohibiting Unit: Does Not Exist

Machine1_enable = AckOneToZero

Example #3: pscsi-tsend (pscsi-tsend.auxhw)

Auxiliary File Generated by BM_DECOMP (BM Machines)

Machine 2

Primary Input 1: DWAckNormN

LATCH REQUIRED: NO

* LATCH REMOVED BY OPTIMIZATION*

Primary Input 2: AckInN

LATCH REQUIRED: YES

LATCH ENABLE LOGIC:

Permitting Unit: AckTwoToThree + ReqOne

Prohibiting Unit: Does Not Exist

Machine2_enable = AckTwoToThree + ReqOne

Machine 3

Primary Input 1: StartDMASend

LATCH REQUIRED: YES

LATCH ENABLE LOGIC:

Permitting Unit: AckThreeToZero

Prohibiting Unit: (ReqOne + GlobalAckThree)'

Machine3_enable = (AckThreeToZero) (ReqOne + GlobalAckThree)'

Example #3: pscsi-tsend (pscsi-tsend.auxhw)

Auxiliary File Generated by BM_DECOMP (BM Machines)

Machine 3 (cont'd)

Primary Input 2: DWAckLastN

LATCH REQUIRED: YES

LATCH ENABLE LOGIC:

Permitting Unit: AckThreeToZero

Prohibiting Unit: GlobalAckThree'

Machine3_enable = (AckThreeToZero) (GlobalAckThree')

Primary Input 3: DWAckNormN

LATCH REQUIRED: NO

Latch removed by optimization

Primary Input 4: AckInN

LATCH REQUIRED: YES

LATCH ENABLE LOGIC:

Permitting Unit: AckThreeToZero

Prohibiting Unit: GlobalAckThree'

Machine3_enable = (AckThreeToZero) (GlobalAckThree')

Example #3: pscsi-tsend (pscsi-tsend.auxhw)

Auxiliary File Generated by BM_DECOMP (Output Generators)

Primary Output Generators

- Primary Output 1 = EndDMAInt
  - Generator: AND3 gate (Inputs from Machines Master, 1, 3)
  - Inputs: MasterToEndDMAInt, OneToEndDMAInt, ThreeToEndDMAInt
- Primary Output 2 = DRQ
  - Generator : AND4 gate (Inputs from Machines Master, 1, 2, 3)
  - Inputs: MasterToDRQ, OneToDRQ, TwoToDRQ, ThreeToDRQ
- Primary Output 3 = ReqOutN
  - Generator: XNOR3 gate (Inputs from Machines 1, 2, 3)
  - Inputs: OneToReqOutN, TwoToReqOutN, ThreeToReqOutN

Conclusions

bm_decomp CAD tool

- An extension of MINIMALIST
- Decomposition technique for BM and XBM controllers
- Inter-controller communication protocol
- Additional hardware
  - Optimizations proposed to remove & reduce hardware