Recent Advances in Designing Clockless Digital Systems

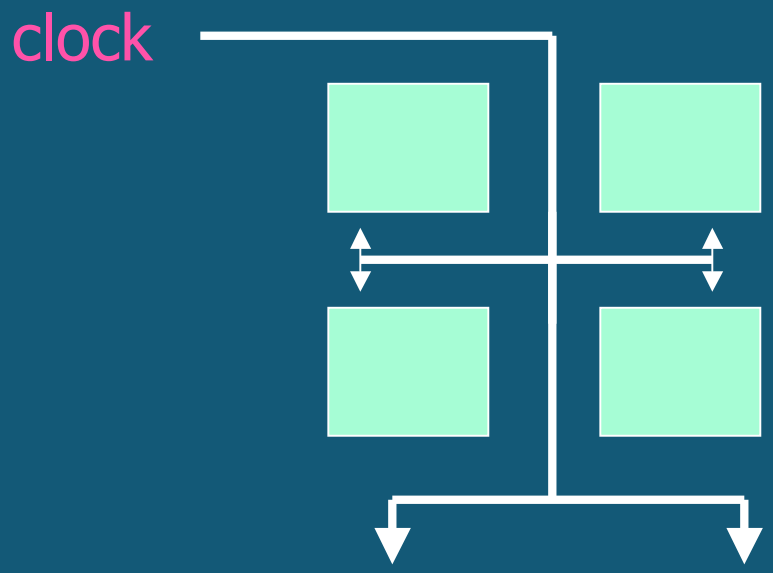
Prof. Steven M. Nowick

nowick@cs.columbia.edu

Chair, Computer Engineering Program
Department of Computer Science (and Elect. Eng.)
Columbia University
New York, NY, USA

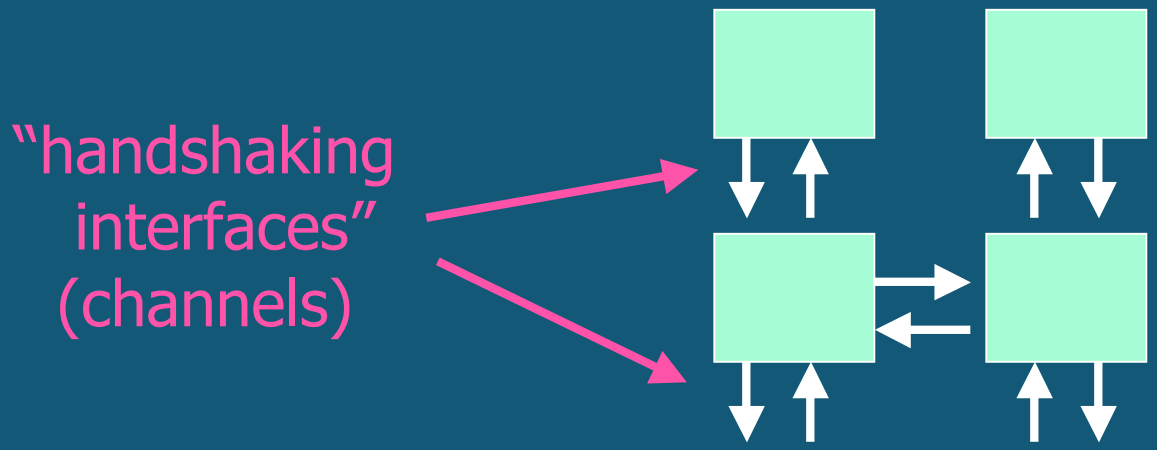
Introduction

- Synchronous vs. Asynchronous Systems?
  - Synchronous Systems: use a *global clock*
    - entire system operates *at fixed-rate*
    - uses *"centralized control"*



Introduction (cont.)

- Synchronous vs. Asynchronous Systems? (cont.)
  - Asynchronous Systems: *no global clock*
    - components can operate at *varying rates*
    - *communicate locally* via *"handshaking"*
    - uses *"distributed control"*



Trends and Challenges

Trends in Chip Design: next decade

- *"Semiconductor Industry Association (SIA) Roadmap"*

Unprecedented Challenges:

- complexity and scale (= size of systems)
- clock speeds
- power management
- reusability & scalability
- reliability
- "time-to-market"

Design becoming unmanageable using a centralized single clock (synchronous) approach….

Trends and Challenges (cont.)

1. Clock Rate:

- *1980: several MegaHertz*
- *2001: ~750 MegaHertz - 1+ GigaHertz*
- *2008: 5-6 GigaHertz (and sometimes falling!)*

Design Challenge:

- *"clock skew":* clock must be near-simultaneous across entire chip

Trends and Challenges (cont.)

2. Chip Size and Density:

Total #Transistors per Chip: *60-80% increase/year*

- *~1970: 4 thousand* (Intel 4004 microprocessor)
- *today: 50-200+ million*
- *2008 and beyond: 1 billion+*

Design Challenges:

- system complexity, design time, clock distribution
- *clock to require 10-20 cycles to reach across chip*

Trends and Challenges (cont.)

3. Power Consumption

- Low power: ever-increasing demand
  - consumer electronics: battery-powered
  - high-end processors: avoid expensive fans, packaging

Design Challenge:

- *clock inherently consumes power continuously*
- "power-down" techniques: add complexity, only partly effective

Trends and Challenges (cont.)

4. Time-to-Market, Design Re-Use, Scalability

Increasing pressure for faster *"time-to-market".* Need:

- reusable components: "plug-and-play" design
- flexible interfacing: under varied conditions, voltage scaling
- scalable design: easy system upgrades

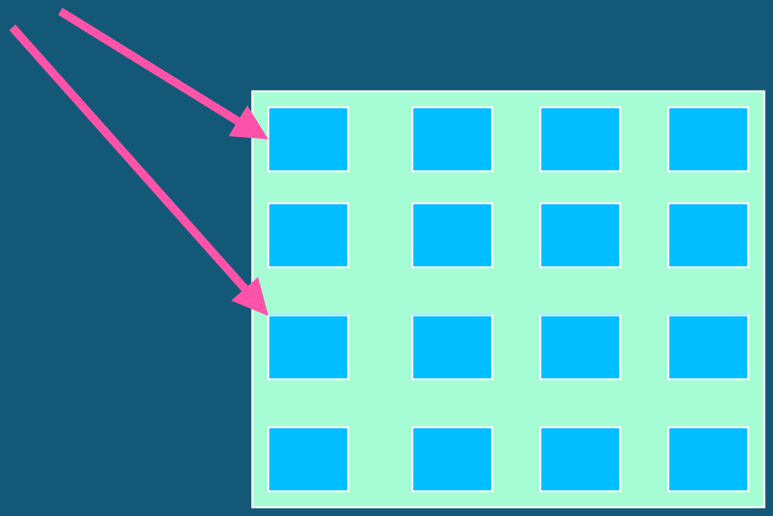
Design Challenge: mismatch with central fixed-rate clock

Trends and Challenges (cont.)

5. Future Trends: "Mixed Timing" Domains

Chips themselves becoming *distributed systems….*

- contain many sub-regions, *operating at different speeds:*



Design Challenge: breakdown of single centralized clock control

Asynchronous Design: Potential Advantages

Several Potential Advantages:

- Lower Power
  - no clock
    - ➔ components use power only "on demand"
    - ➔ *avoid global clock distribution*
    - ➔ effectively provides automatic clock gating at arbitrary granularity
- Robustness, Scalability
  - no global timing
    - ➔ "mix-and-match" variable-speed components
    - ➔ supports dynamic voltage scaling
  - composable/modular design style ➔ "object-oriented"
- Higher Performance
  - systems not limited to "worst-case" clock rate
- "Demand- (Data-) Driven" Operation
  - provides instantaneous wake-up from standby mode

Asynchronous Design: Recent Industrial Developments

1. Philips Semiconductors:

- Wide commercial use: 300 million async chips for consumer electronics:
  pagers, cell phones, smart cards, digital passports, automotive
- Benefits (vs. sync):
  - *3-4x lower power (and lower energy consumption/ops)*
  - *much lower "electromagnetic interference"* (EMI)
  - instant startup from stand-by mode (no PLL's)
- Complete CAD tool flows:
  - "Tangram": Philips (mid-90's to early 2000's)
  - "Haste": Handshake Solutions (incubated spinoff) (early 2000's to present)
- Synthesis strategy: *syntax-directed compilation*
  - *starting point: concurrent HDL ("Tangram", "Haste")*
  - *2-step synthesis:*
    - front-end: HDL spec => intermediate netlist of concurrent components
    - back-end: each component => standard cell (… then physical design)
  - *+: fast, 'transparent', easy-to-use*
  - *-: few optimizations, low/moderate-performance only*

Asynchronous Design: Recent Industrial Developments

2. Intel:

- experimental Pentium instruction-length decoder = "RAPPID" (1990's)
- *3-4x faster* than synchronous subsystem
- *~2x lower power*

3. Sun Labs:

- commercial: high-speed FIFO's in recent "Ultra's" (memory access)

4. IBM Research:

- experimental: high-speed pipelines, FIR filters, mixed-timing systems

5. Recent Async Startups:

- Fulcrum Microsystems (California): *Ethernet routing chips*
- Camgian Systems: *very low-power/robust designs (sensors, etc.)*
- Handshake Solutions (Netherlands): *incubated by Philips, tools + design*
- Silistrix (UK): *interconnect for low-end heterogenous/mixed-timing systems*
- Achronix: *FPGA's for bit-sliced fine-grained pipelined systems (fixed style)*

Asynchronous CAD Tools: Recent Developments

DARPA's "CLASS" Program (2003-2007):

- Major clockless initiative ($14M): to make async commercially viable

Goals:

- CAD tool: produce viable commercial-grade async tool flow
- demonstration: a complex Boeing ASIC chip

Participants:

- Lead (PI): Boeing
- Industrial participants:
  - Philips (via async incubated startup, "Handshake Solutions")
  - Theseus Logic, Codetronix
- Academic participants:
  - Columbia (Nowick), UNC, UW, Yale, OSU

Target: cover wide "design space" – very robust to high-speed circuits

Asynchronous Design: Challenges

Critical Design Issues:

- components must *communicate cleanly:* *'hazard-free' design*
- *highly-concurrent designs:* harder to verify!

Lack of Automated "Computer-Aided Design" Tools:

- most commercial "CAD" tools targeted to synchronous

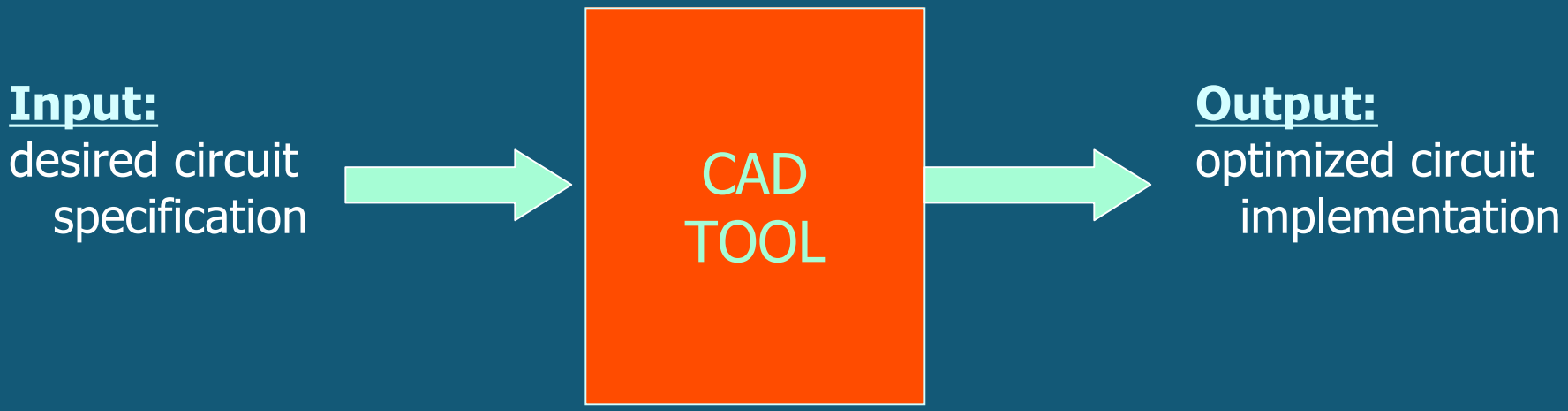
... but recent industrial advances -- Philips' Handshake Solutions:

uses Synopsys/Magma/Cadence physical design tools

What Are CAD Tools?

Software programs to aid digital designers =
"computer-aided design" tools

- automatically *synthesize* and *optimize* digital circuits



Asynchronous Design Challenge

Lack of Existing Asynchronous Design Tools:

- Most commercial "CAD" tools targeted to synchronous
- Synchronous CAD tools:
  - major drivers of growth in microelectronics industry
- Asynchronous "chicken-and-egg" problem:
  - few CAD tools ←→ less commercial use of async design
  - especially lacking: tools for designing/optmzng. large systems

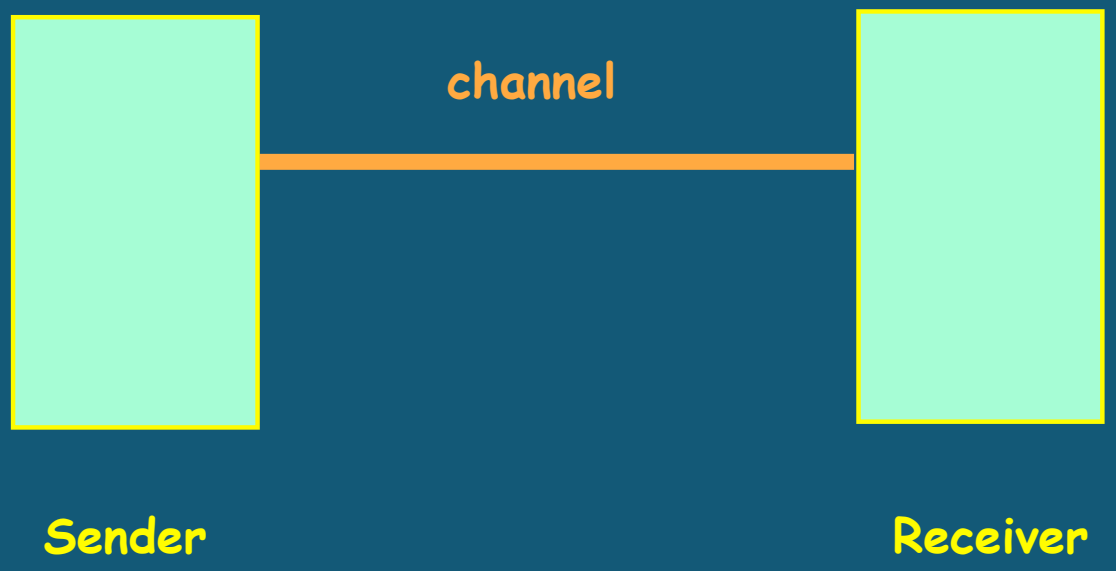
Asynchronous Basics

Large variety of asynchronous design styles

- Address different points in "design-space" spectrum
- Example targets:
  - highly-robust:
    - providing near "delay-insensitive (DI)" operation
  - ultra-low power (or energy):
    - "on-demand" operation, instant wakeup
  - ease-of-design/moderate performance
    - e.g. Philips' style
  - very high-speed: async pipelines (with localized timing constraints)
    - … comparable to high-end synchronous
    - with added benefits: support variable-timing I/O rates, function blocks
  - support for heterogeneity: mixed sync/async systems
    - "GALS-style" (globally-async/locally-sync)

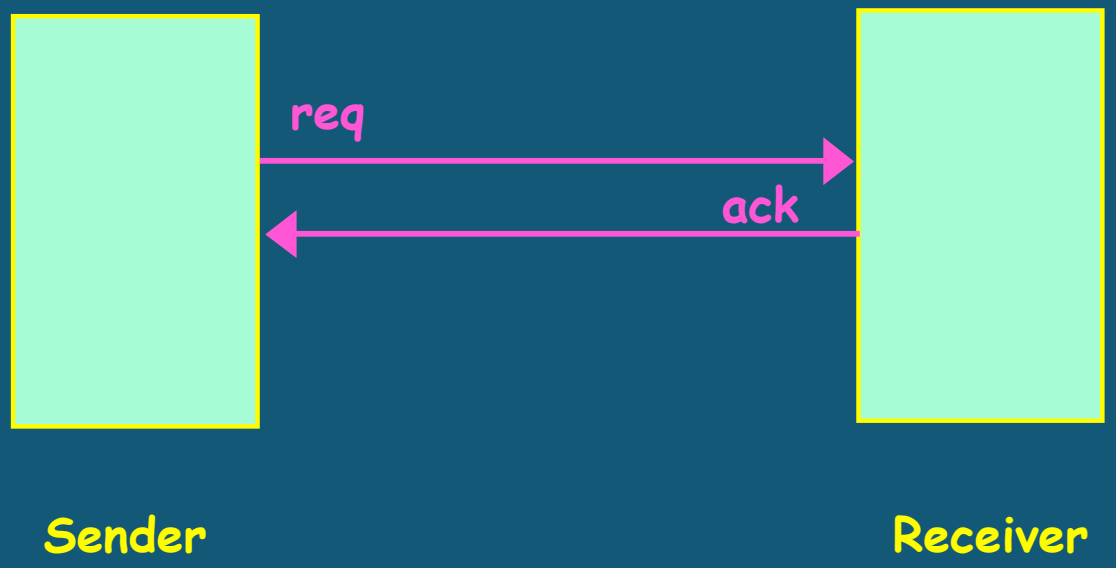
Overview: Asynchronous Communication

Components usually communicate & synchronize on channels

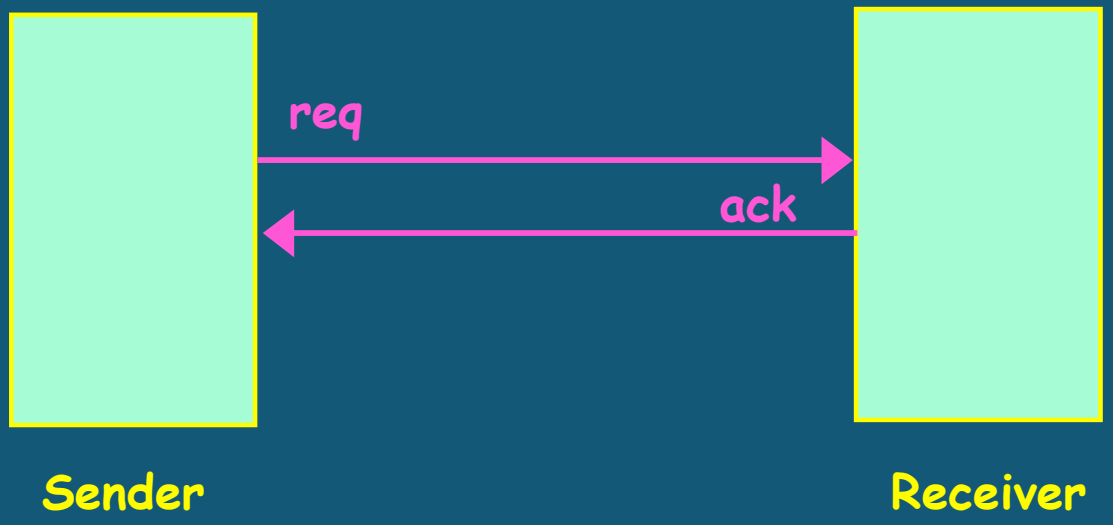


Overview: Signalling Protocols

Communication channel: usually instantiated as 2 wires

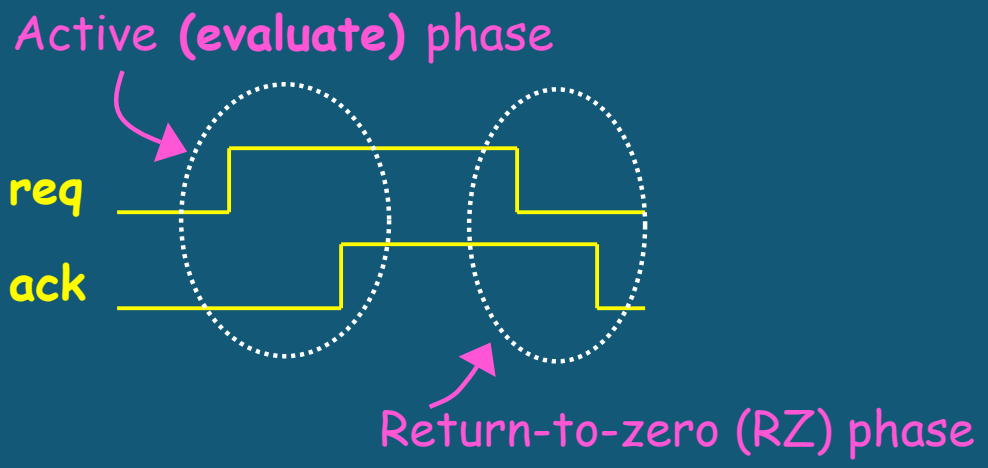


Overview: Signalling Protocols

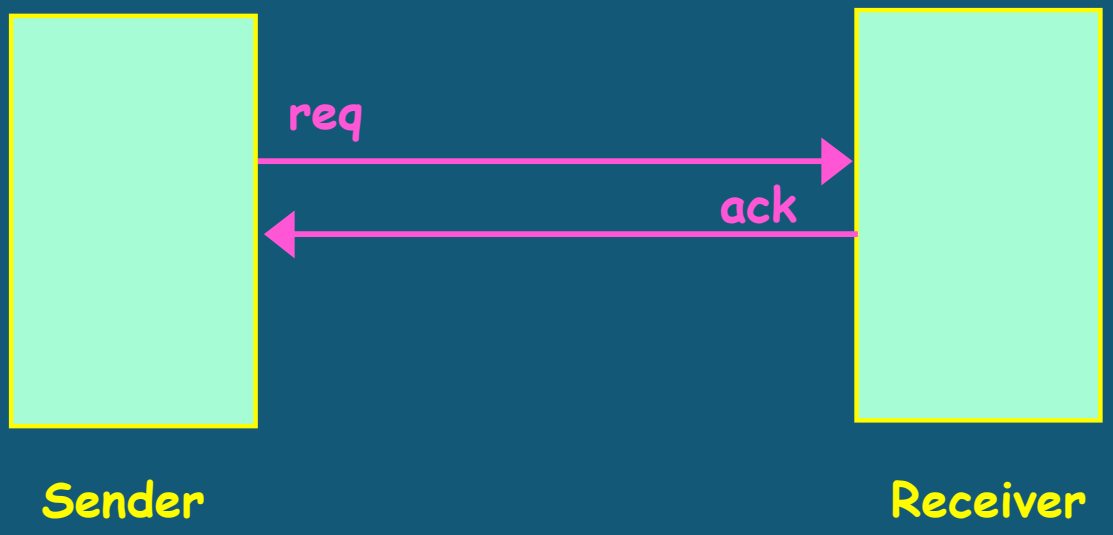


4-Phase Handshaking

One transaction
(return-to-zero [RZ]):

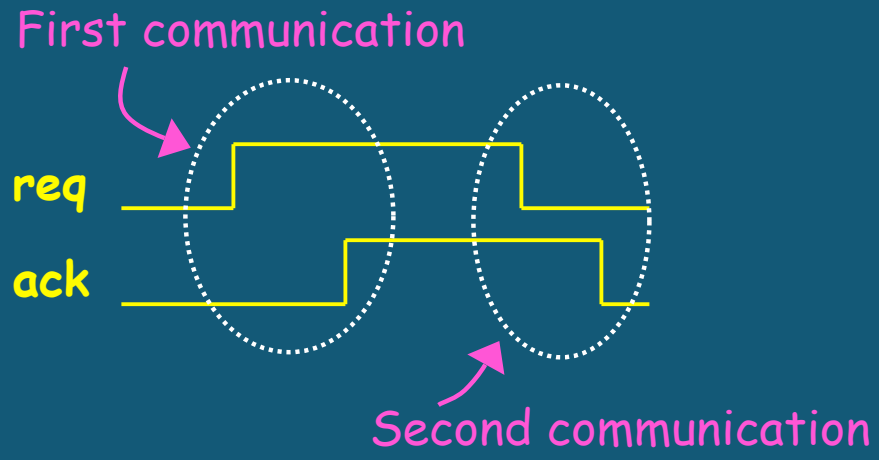


Overview: Signalling Protocols



2-Phase Handshaking = "Transition-Signalling"

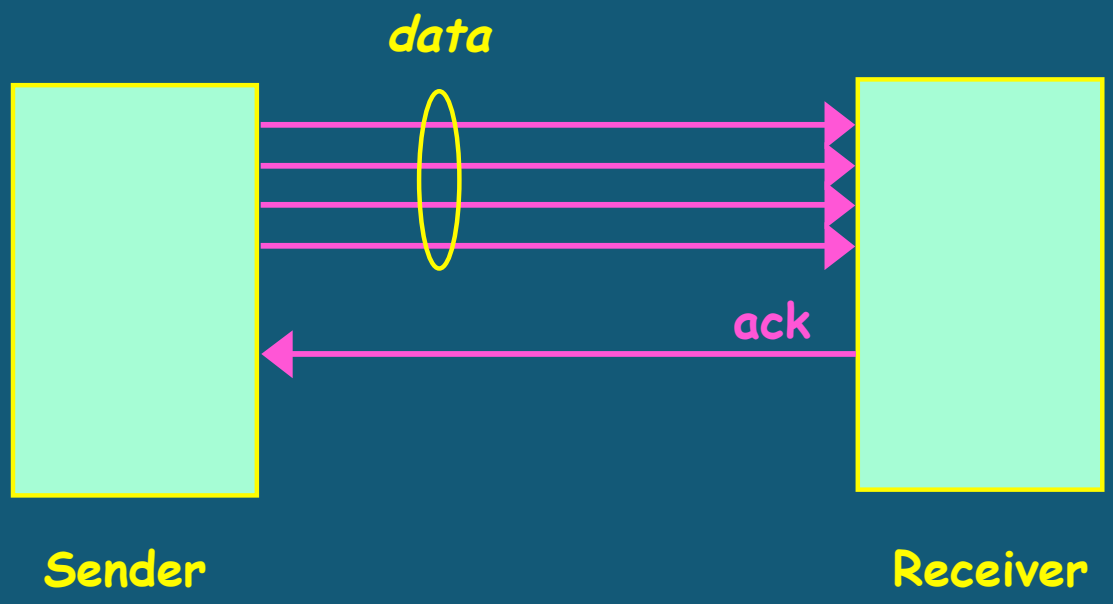
Two transactions
(non-return-to-zero [NRZ]):



Overview: How to Communicate Data?

Data channel: replace "req" by (encoded) data bits

- … still use 2-phase or 4-phase protocol



Overview: How to Encode Data?

A variety of asynchronous data encoding styles

- Two key classes: (i) "DI" (delay-insensitive) or (ii) "timing-dependent"
- … each can use *either* a 2-phase or 4-phase protocol

DI Codes: provides timing-robustness (to arbitrary bit skew, arrival times, etc.)

- 4-phase (RZ) protocols:
  - dual-rail (1-of-2): *widely used!*
  - 1-of-4 (or m-of-n)
- 2-phase (NRZ) protocols:
  - transition-signaling (1-of-2)
  - LEDR (1-of-2) ["level-encoded dual-rail"] [Dean/Horowitz/Dill, Advanced Research in VLSI '91]
  - LETS (1-of-4) ["level-encoded transition-signalling"] [McGee/Agyekum/Mohamed/Nowick IEEE Async Symp. '08]

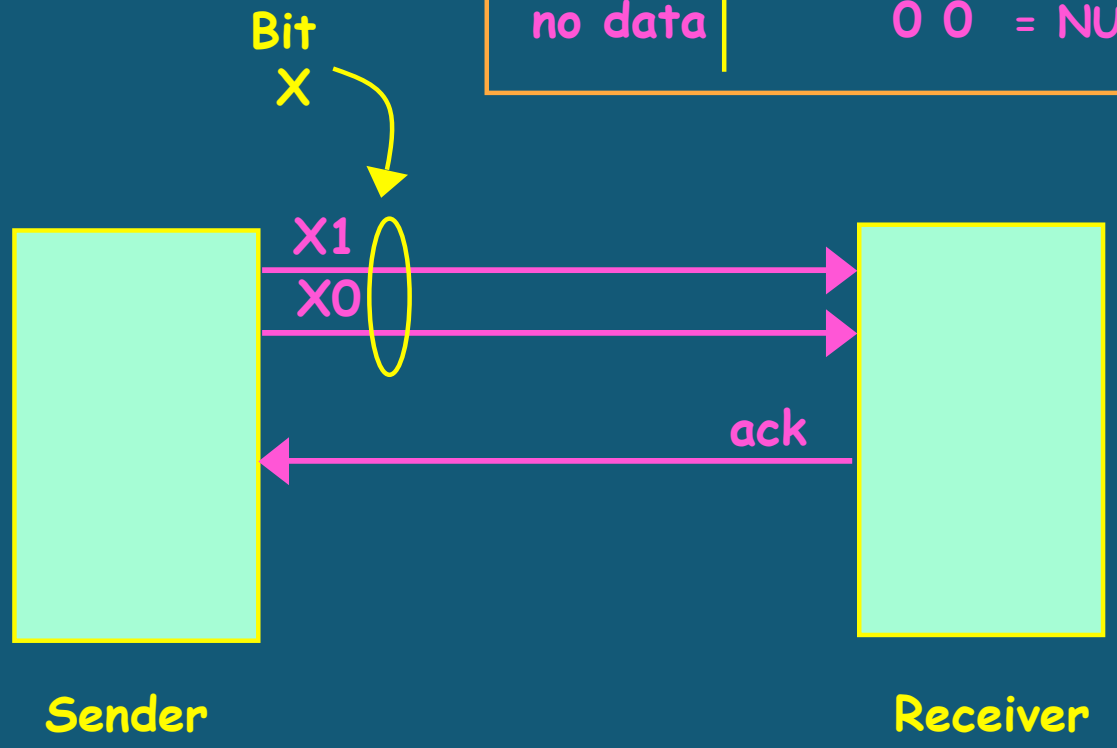
Timing-Dependent Codes: use localized timing assumptions

- Single-rail "bundled data": *widely used! = sync encoding + matched delay*
- Other: "pulse-mode", etc.

Overview: How to Encode Data?

"dual-rail": 4-Phase (RZ)

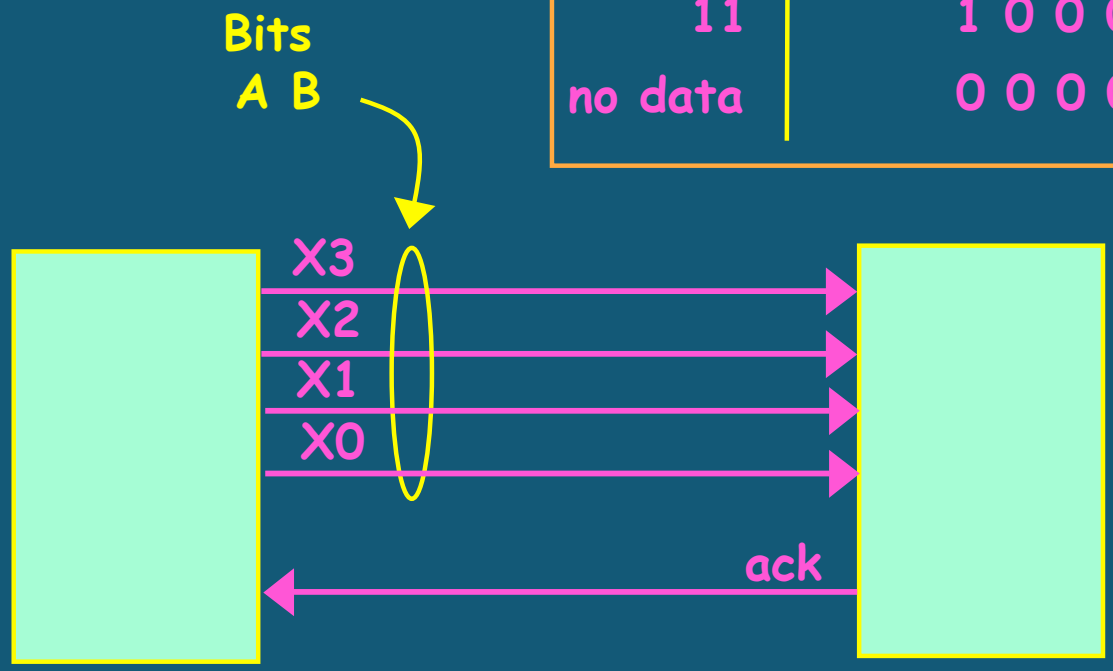
Bit X	Dual-rail encoding X1 X0
0	0 1
1	1 0
no data	0 0 = NULL (spacer)



Overview: How to Encode Data?

"1-of-4": 4-Phase (RZ)

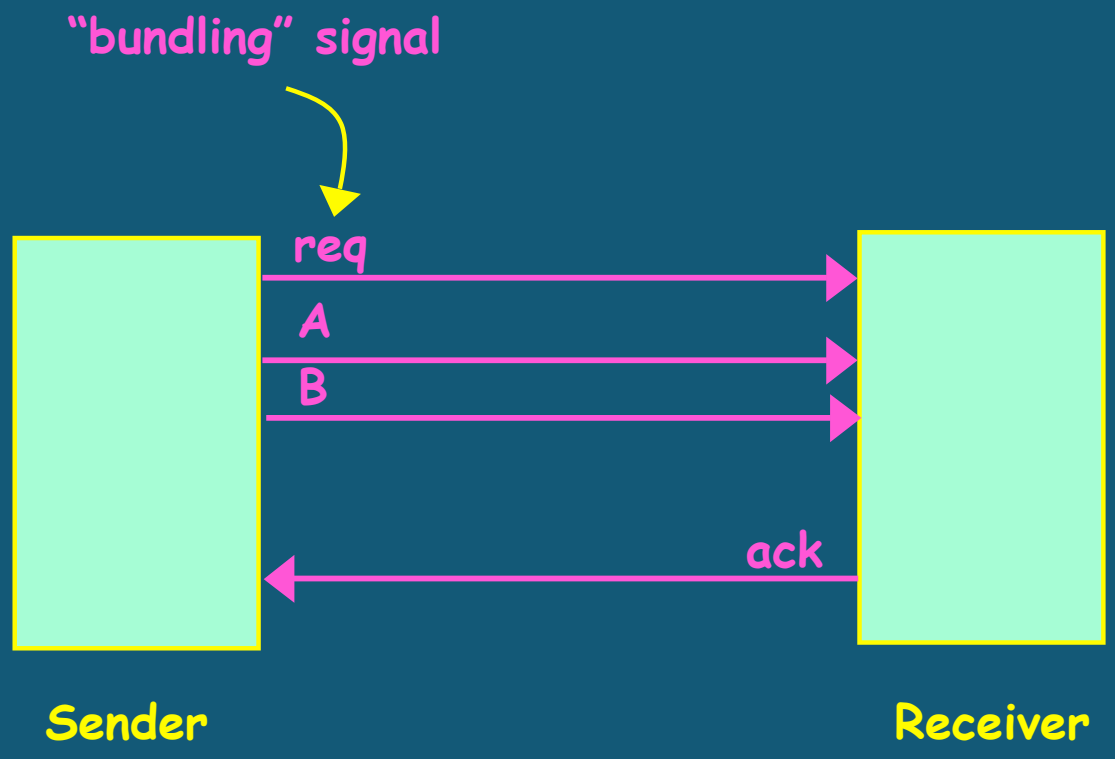
Bits A B	Dual-rail encoding X3 X2 X1 X0
00	0 0 0 1
01	0 0 1 0
10	0 1 0 0
11	1 0 0 0
no data	0 0 0 0 = NULL (spacer)



Overview: How to Encode Data?

Single-Rail "Bundled Data": 4-Phase (RZ)

Uses synchronous (single-rail) data
+ local worst-case "model delay"



Signalling Protocols + Data Encoding: Tradeoffs

DI Codes: provides timing-robustness

- 4-phase (RZ) protocols: -: poorer system throughput + power (2 roundtrips), +: easy function block design
  - dual-rail (1-of-2): *worse power (# rail transitions)*
  - 1-of-4 (or m-of-n) *better power (# rail transitions)*
- 2-phase (NRZ) protocols: +: better system throughput + power (1 roundtrip), -: difficult to design function blocks
  - transition-signaling (1-of-2) *worse power (# rail transitions)*
  - LEDR (1-of-2) *better power (# rail transitions)*

[Dean/Horowitz/Dill, Advanced Research in VLSI '91]

  - LETS (1-of-4) *best power (# rail transitions)*

[McGee/Agyekum/Mohamed/Nowick IEEE Async Symp. '08]

Timing-Dependent Codes: good power + ease of function design/poor robustness

- Single-rail "bundled data": *widely used! = sync encoding + matched delay*
- Other: "pulse-mode", etc.

Async Protocols: Evaluation Summary

Robust/High-Throughput Global Communication:

- High throughput + low power: 2-phase (NRZ) protocols (LETS)

Efficient Local Computation (easy-to-design function blocks):

- Ease-of-design + low area + low power:
  - Timing Robust (DI): 4-phase (RZ) protocols (dual-rail, 1-of-4)
  - Non-DI: single-rail bundled data (2-/4-phase)

Our recent research: efficient protocol converters

- Global communication: use 2-phase (LEDR, LETS)
- Local computation: use 4-phase (bundled, dual-rail, 1-of-4)

[McGee/Agyekum/Mohamed/Nowick IEEE Async Symp. '08]

An Asynchronous CAD Framework: Philips

For large async systems:

- **Tangram:** Philips Semiconductors (since mid-1980's)
  -- developed in research labs (van Berkel, et al.)
  -- commercial use in product divisions (several countries)
- **Haste:** Handshake Solutions (incubated Philips spinoff)
  -- commercial use

Target: low-/medium-performance consumer electronics

Starting point: high-level behavioral system specification

- use concurrent program language (based on CSP)
- features: block-structured, algorithmic, models concurrency

End point: VLSI circuit implementation (layout)

Asynchronous CAD Frameworks

Commercial applications:

- **Tangram:** microcontroller chips, error correctors, …
  - in several commercial Philips products:

    ==> smartcards, pagers, cell phones, automotive, digital passports
- **Haste:** entire ARM processors, … (offered by ARM Ltd.)

Many sophisticated tool features:

- profilers, early estimation tools (power, delay), testing
- Benefits: rapid development, ease-of-design

History: based on "Macromodules Project" (Clark/Molnar, Wash. U., 1960's)

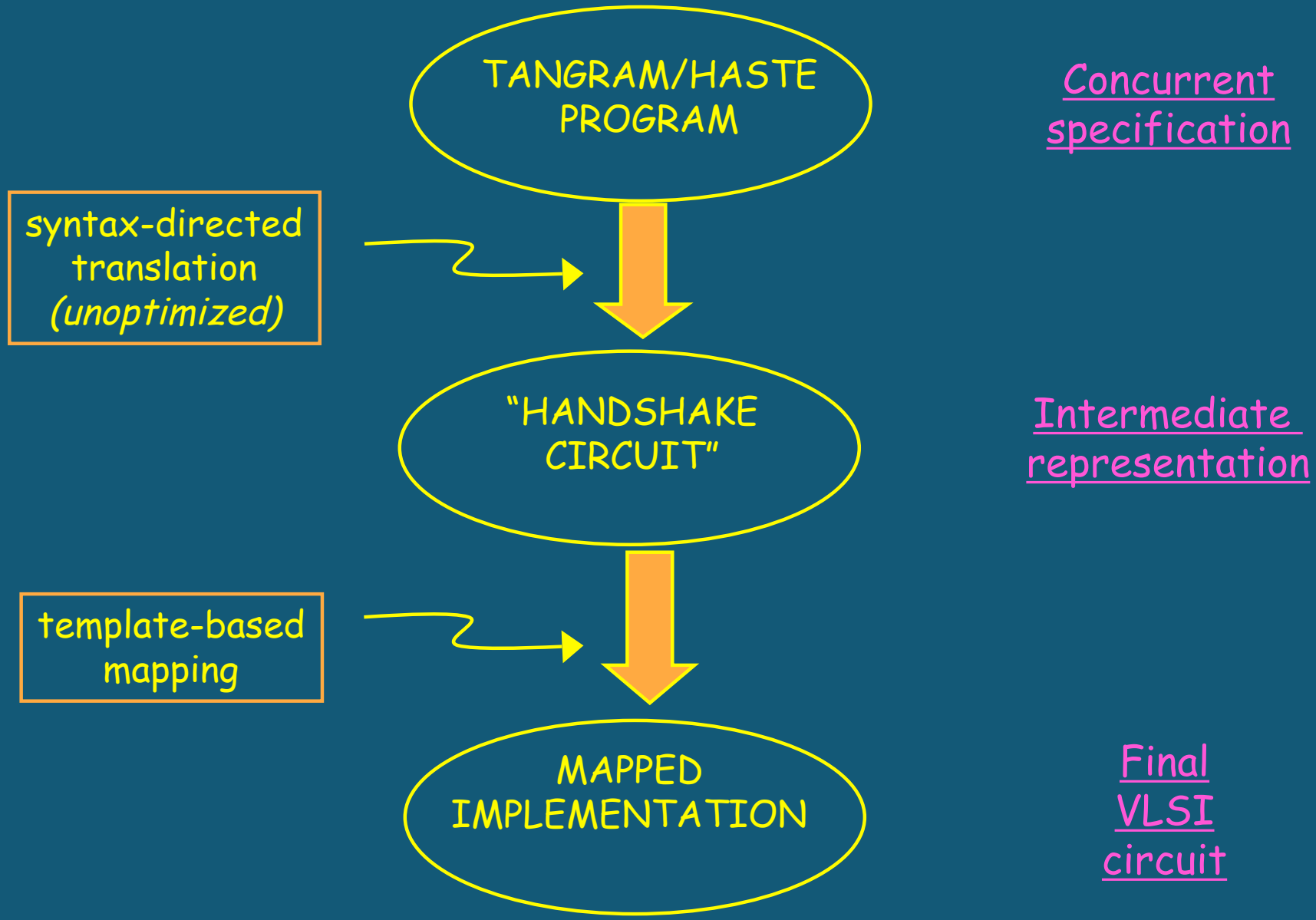
Tangram/Haste CAD Frameworks

2 main synthesis steps

- Syntax-directed translation:
  - start with concurrent "program" = system specification
  - translate to intermediate network of handshake components
- Template-based mapping:
  - map each handshake component directly into library modules

- **Advantages:**
  - Can synthesize large systems
  - Good runtime ⇒ syntax-directed compilation
  - "Transparency": final circuit is predictable, matches spec!
- **Disadvantages:**
  - Few optimizations!: circuits often have poor performance

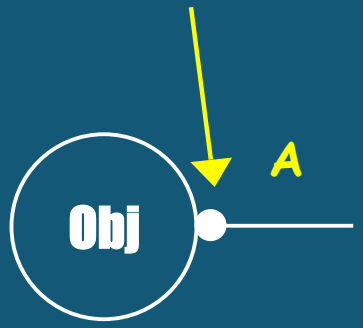
Basic Automated Compiler Flow: Tangram



Background: Handshake Components

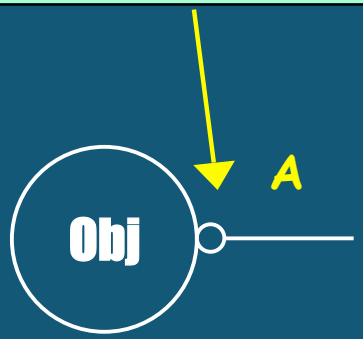
#1. Active Port: *initiates communication*

Handshake component:

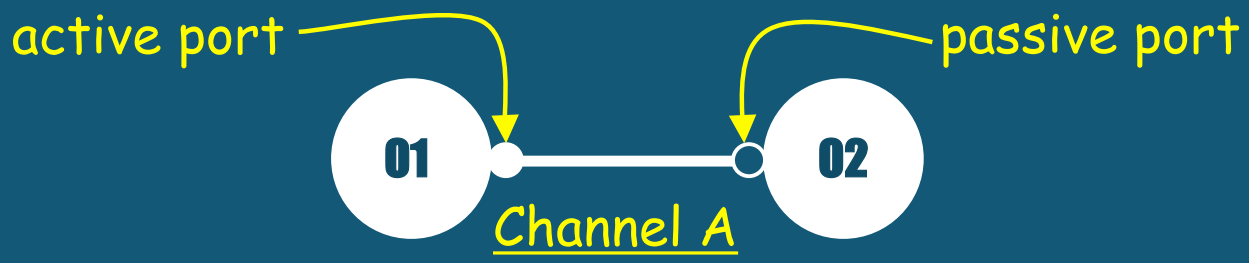


#2. Passive Port: *responds to communication*

Handshake component:



Background: Channel-Based Communication



Components communicate using *"4-phase handshaking"*

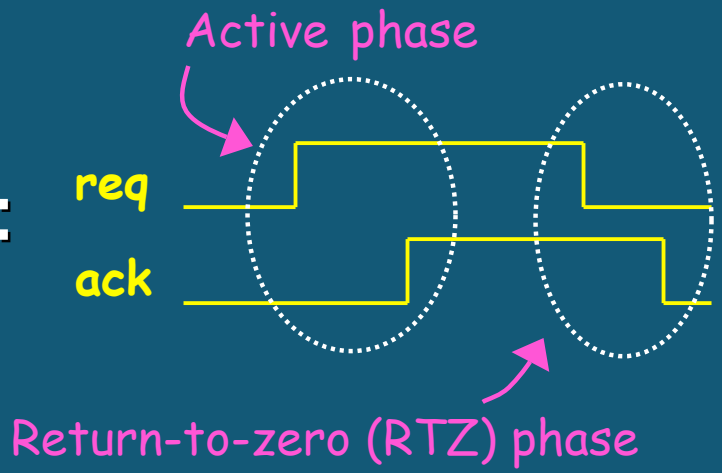
- **O1:** *initiates* communication
- **O2:** *completes* communication

Channel implmtn. => use 2 wires:

req => start operation

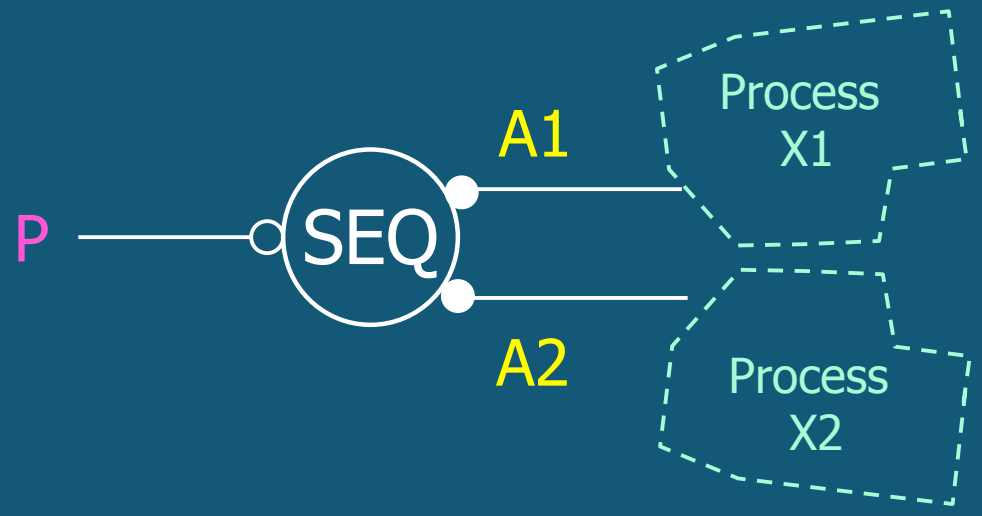
ack => operation done

(... can be extended to handle data)



Basic Handshake Components: Sequencer

2-Way Sequencer: *activated* on channel P;
then *activates 2 processes in sequence* on channels A1 and A2



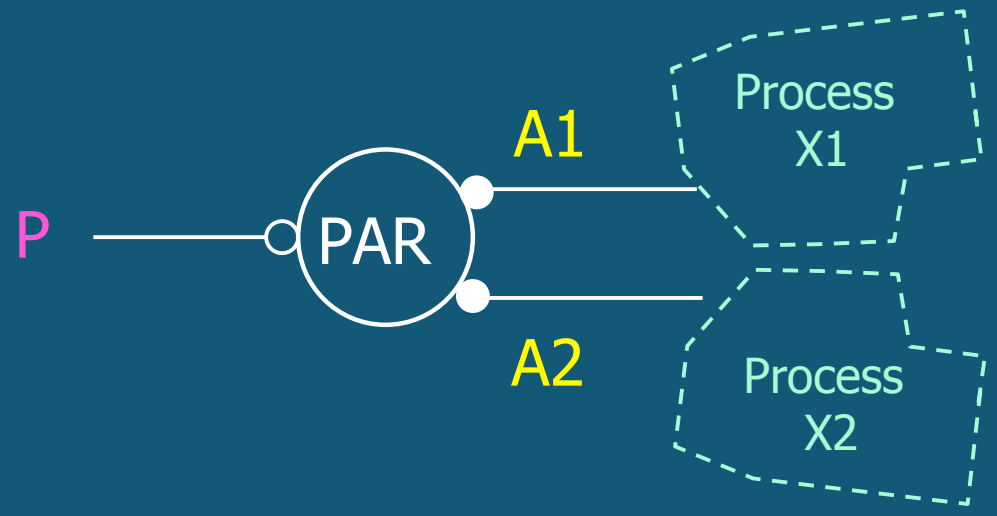
Operation
X1; X2

Goal: activate two sequential processes (i.e.operations)

Basic Handshake Components: PAR Component

PAR Component: *activated* on channel P;

then *activates 2 processes in parallel* on channels A1 and A2

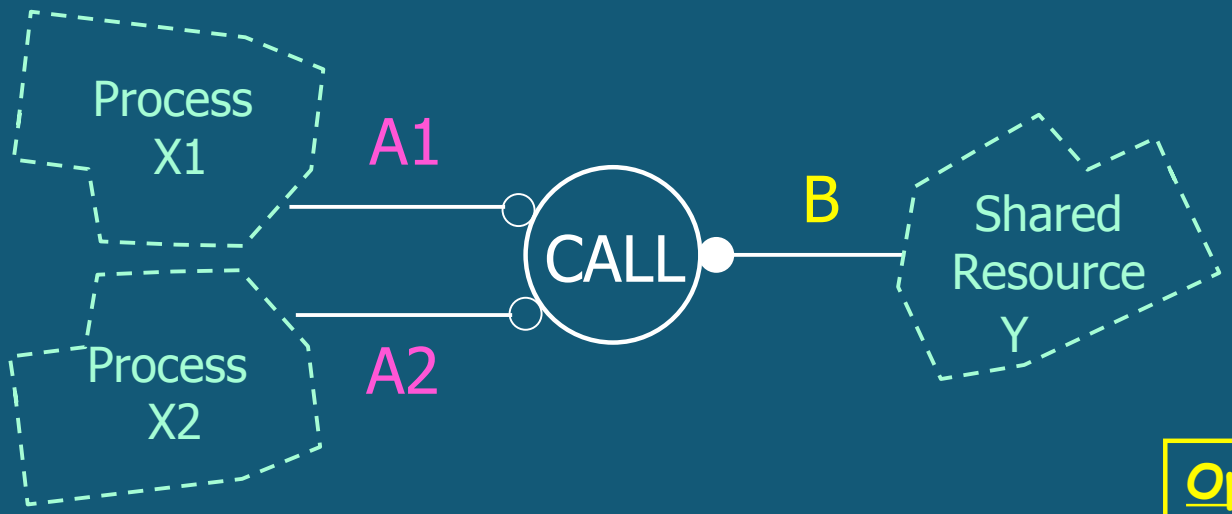


Operation
X1 || X2

Goal: activate two parallel processes

Basic Handshake Components: MIXER (multiplexer)

2-Way "MIXER": *activated* on *either* channel A1 *or* A2;
then *activates process* on channel B



Operation

Xi ==> Y

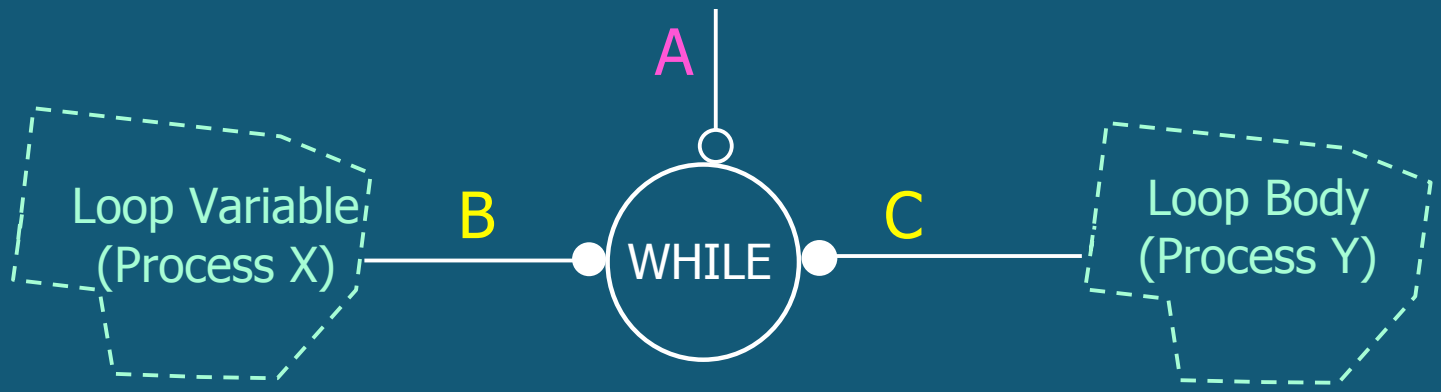
X2 ==> Y

...

Goal: facilitate resource sharing between 2 mutually-exclusive processes

Basic Handshake Components: WHILE Module

WHILE Module: *activated* on channel A;
repeat {*while loop variable TRUE* on channel B,
activate loop body on channel C}



TEST LOOP VARIABLE

EXECUTE LOOP BODY

Goal: control "while loop" operation

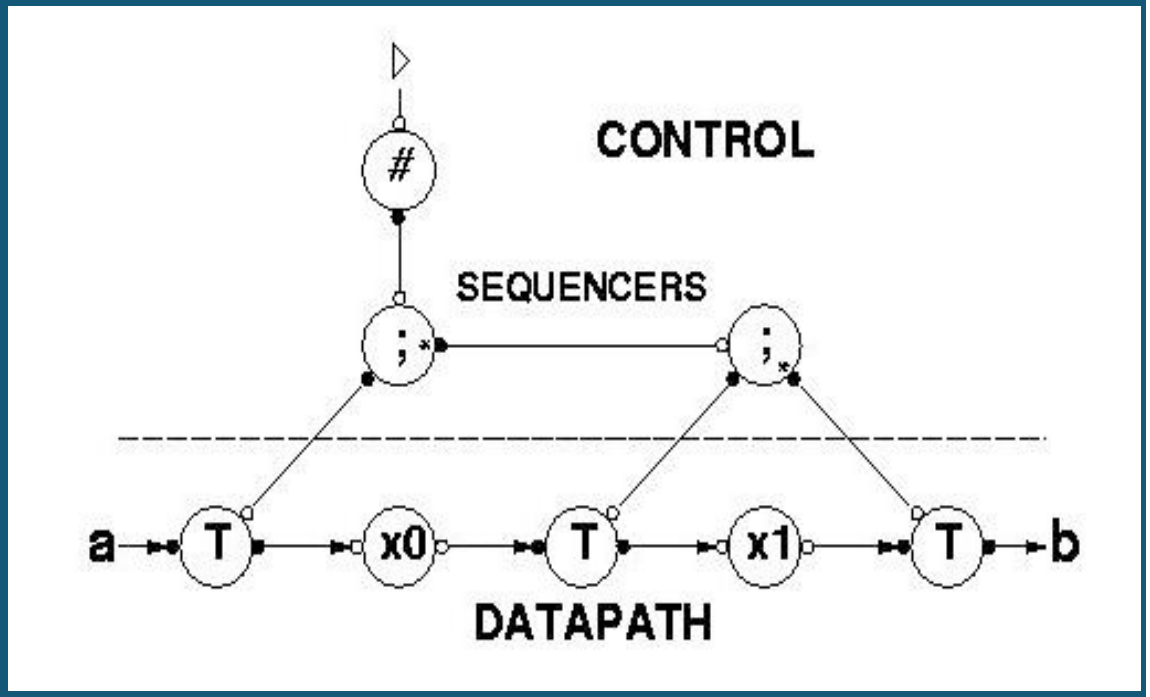
Operation
WHILE (X)
DO Y;

Synthesizing a System: a Small Example

2-Place "Ripple Register" (= FIFO)

Tangram Program ⇨ Intermediate "Handshake Circuit"

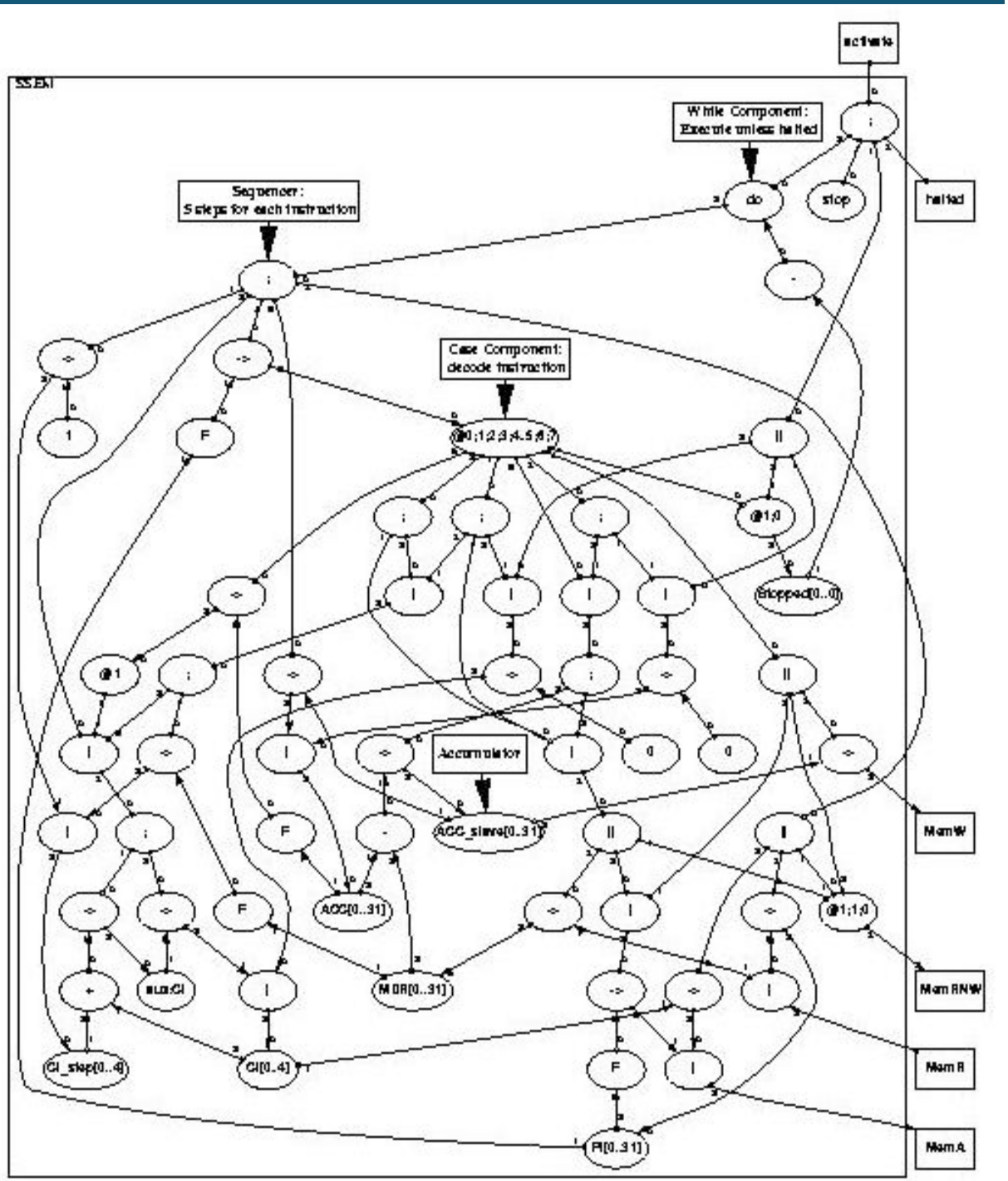
```
proc (a?T  & b!T)
  begin
        x0, x1:   var T
  | forever do
        b! x1;
        x1 := x0;
        a? x0
  od
  end
```



syntax-directed translation (unoptimized)

A Larger Example

Intermediate "Handshake Circuit"



Overview: My Research Areas

- CAD Tools/Algorithms for Asynchronous Controllers (FSM's)
  - "MINIMALIST" Package: for synthesis + optimization
- Mixed-Timing Interface Circuits:
  - for interfacing sync/sync and sync/async systems
- High-Speed Asynchronous Pipelines:
  - for static or dynamic logic

CAD Tools for Async Controllers

MINIMALIST: developed at Columbia University [1994-]

- extensible CAD package for synthesis of asynchronous controllers
- integrates synthesis, optimization and verification tools
- used in 80+ sites/17+ countries (was taught in IIT Bombay)
- URL: http://www.cs.columbia.edu/~nowick/asynctools
- … new release: expected early 2007 *(or contact me)*

Features:

- Scripts vs. custom commands
- Verilog back-end
- Automatic verifier
- Graphical interfaces
- … many optimization modes

Recent application: space measurement chip

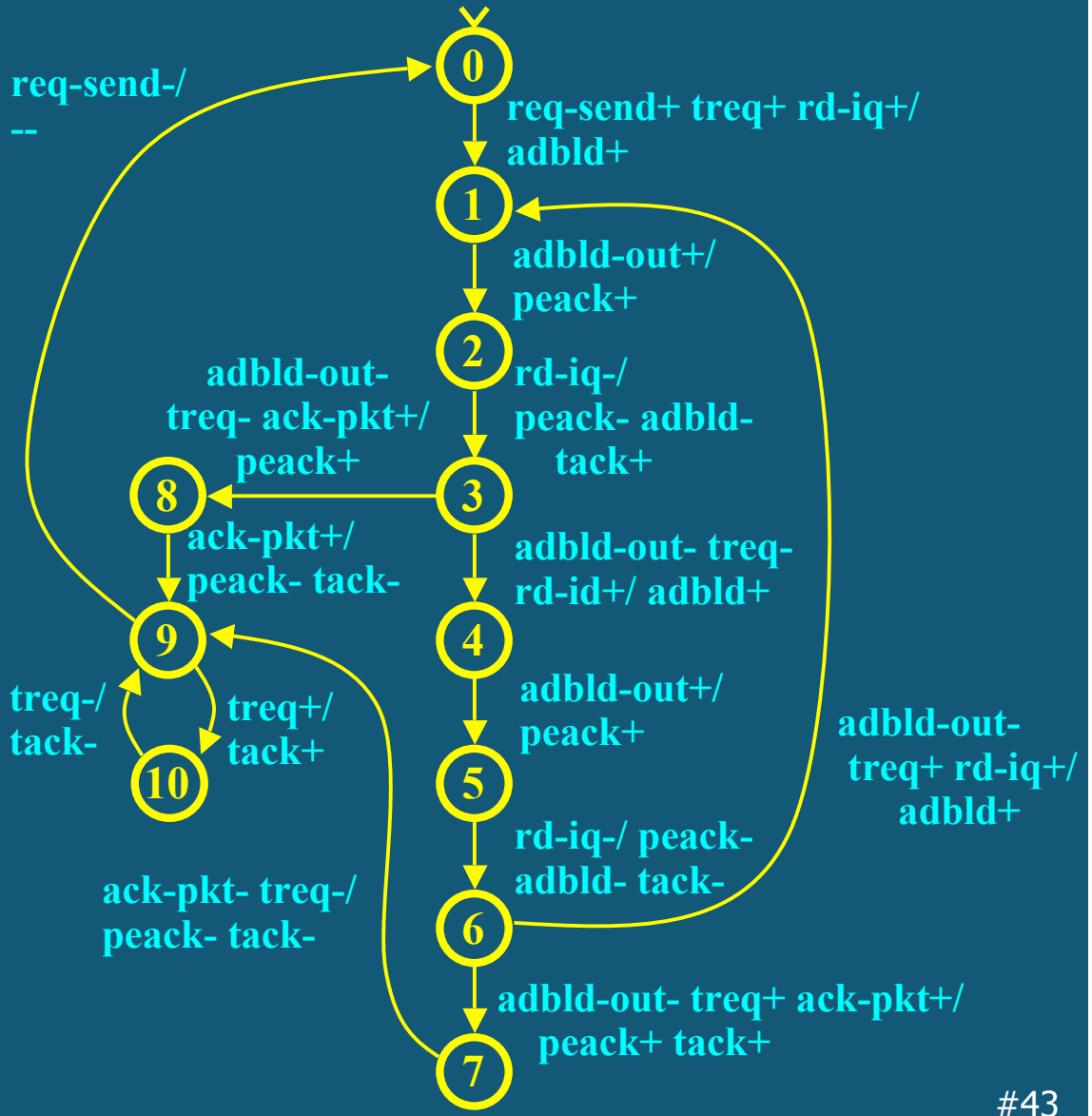
- joint funded project: NASA/Columbia (2006-2007)
- fabricated experimental chip: taped out (Oct. 06)

Key goal: *facilitate design-space exploration*

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

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

Outputs:
tack
peack
adbld

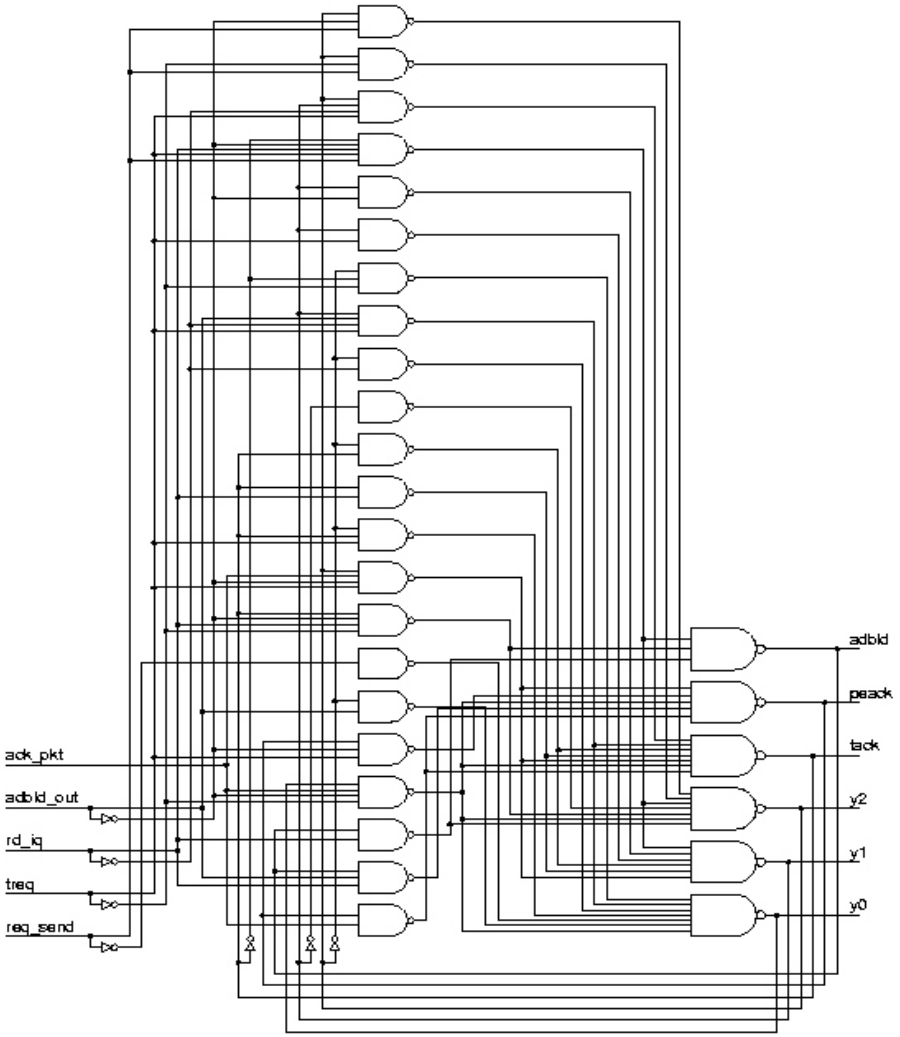


From HP Labs
"Mayfly" Project:
B.Coates, A.Davis, K.Stevens,
"The Post Office Experience: Designing a Large Asynchronous Chip",
INTEGRATION: the VLSI Journal, vol. 15:3, pp. 341-66 (Oct. 1993)

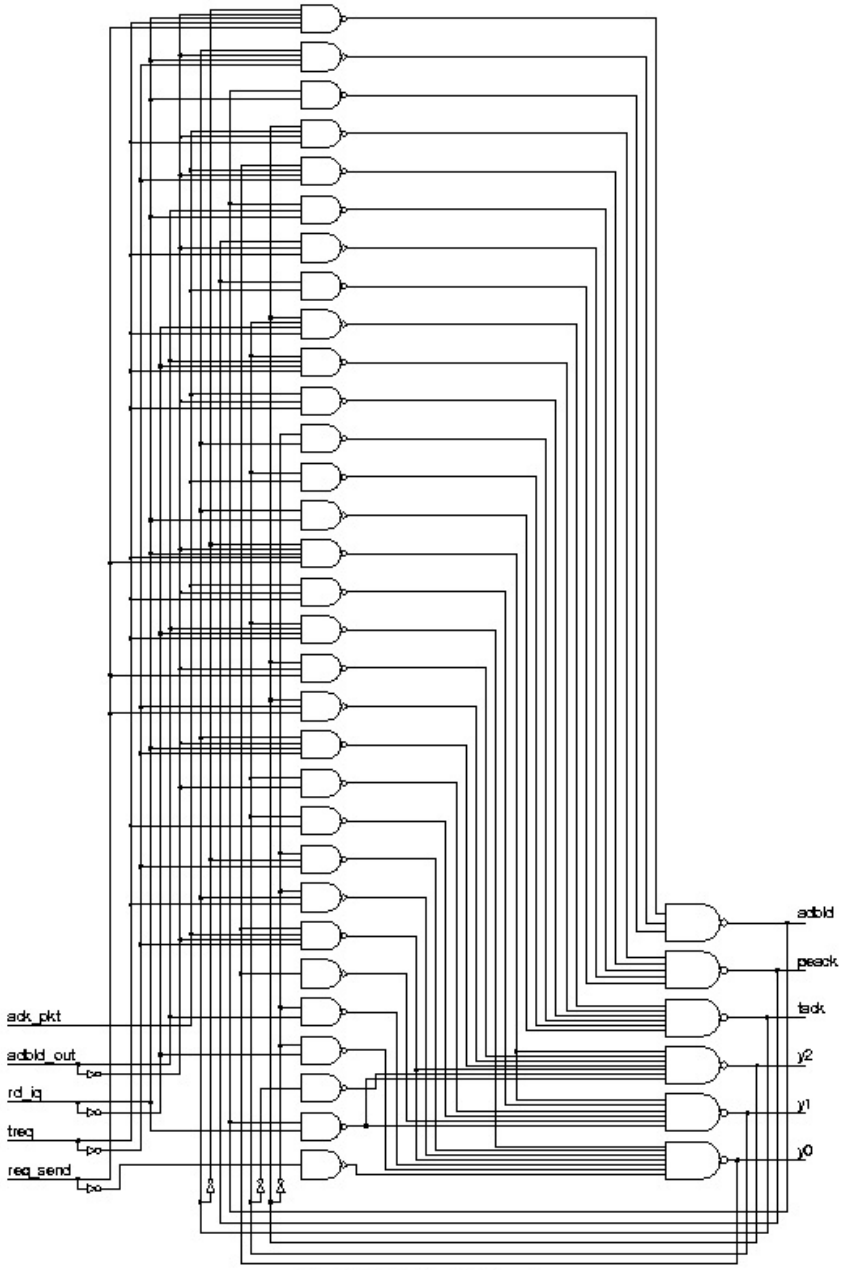
EXAMPLE (cont.):

Design-Space Exploration using MINIMALIST:

optimizing for area vs. speed



PE_SEND_IFC-FL

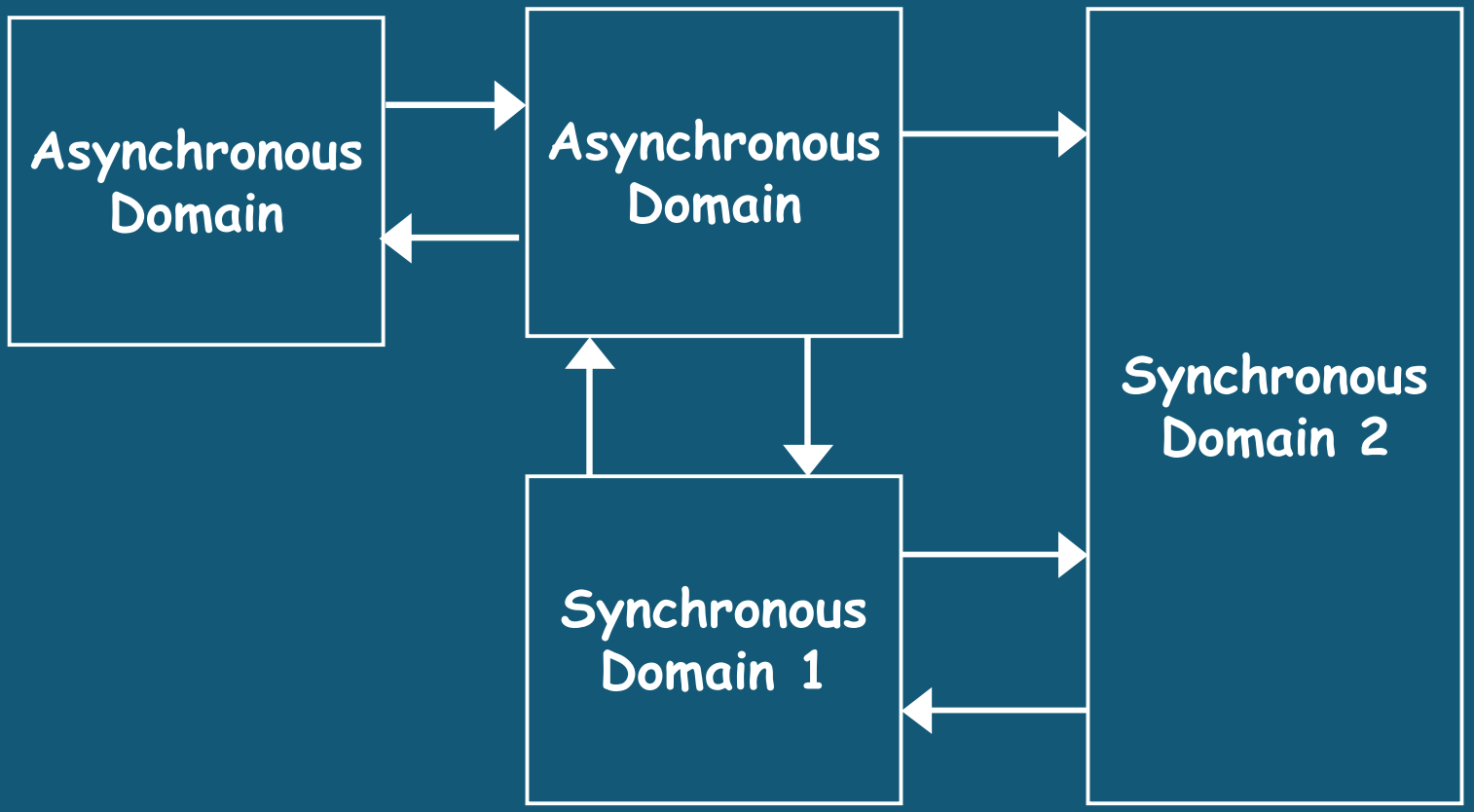


PE_SEND_IFC-Fs

Overview: My Research Areas

- CAD Tools/Algorithms for Asynchronous Controllers (FSM's)
  - "MINIMALIST" Package: for synthesis + optimization
- Mixed-Timing Interface Circuits:
  - for interfacing sync/sync and sync/async systems
- High-Speed Asynchronous Pipelines:
  - for static or dynamic logic

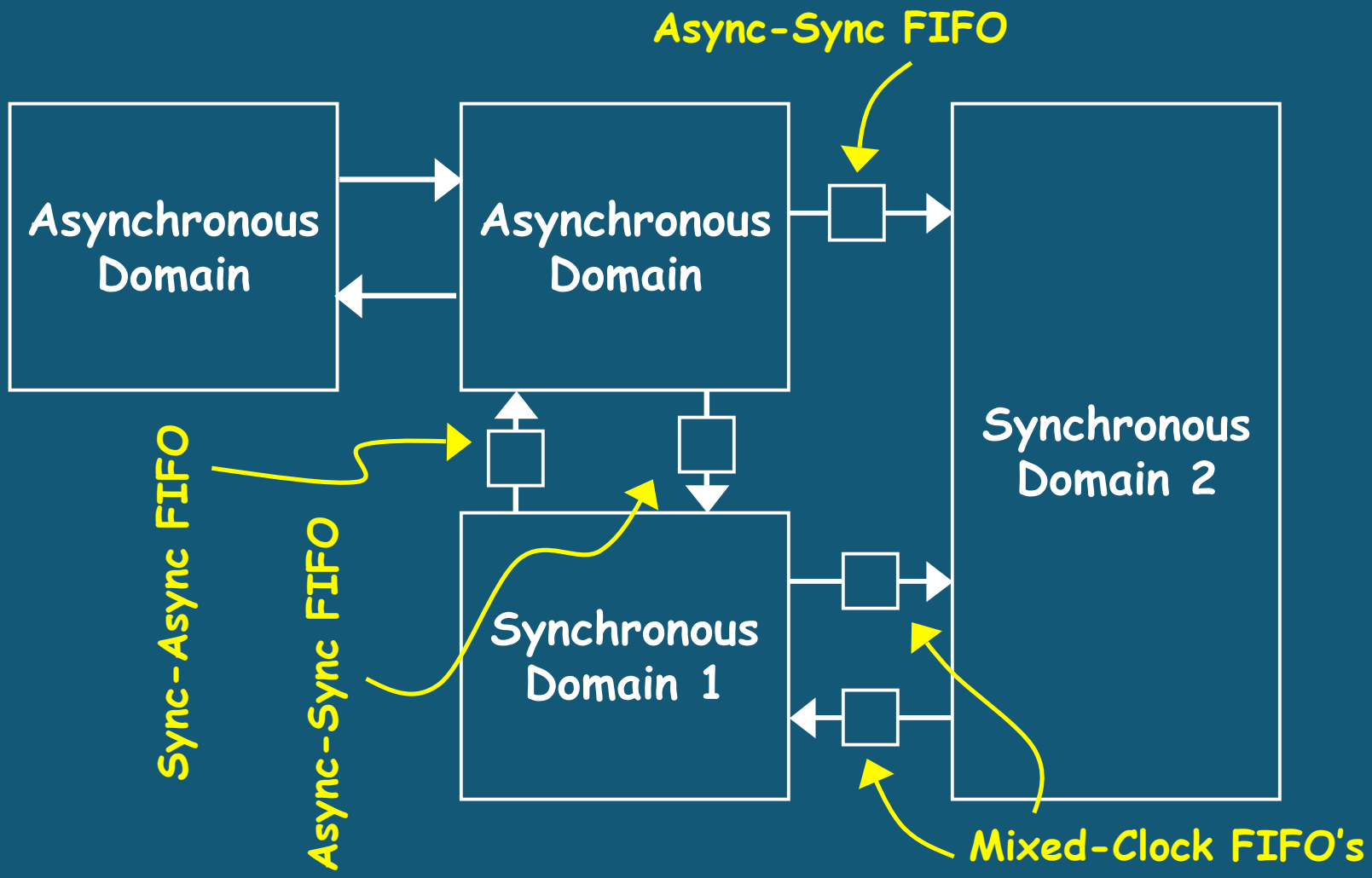
Mixed-Timing Interfaces: Challenge



Goal: provide low-latency communication between "timing domains"

Challenge: avoid synchronization errors

Mixed-Timing Interfaces: Solution



Solution: insert mixed-timing FIFO's ⇒ provide safe data transfer

... developed complete family of mixed-timing interface circuits

[Chelcea/Nowick, IEEE Design Automation Conf. (2001); IEEE Trans. on VLSI Systems v. 12:8, Aug. 2004]

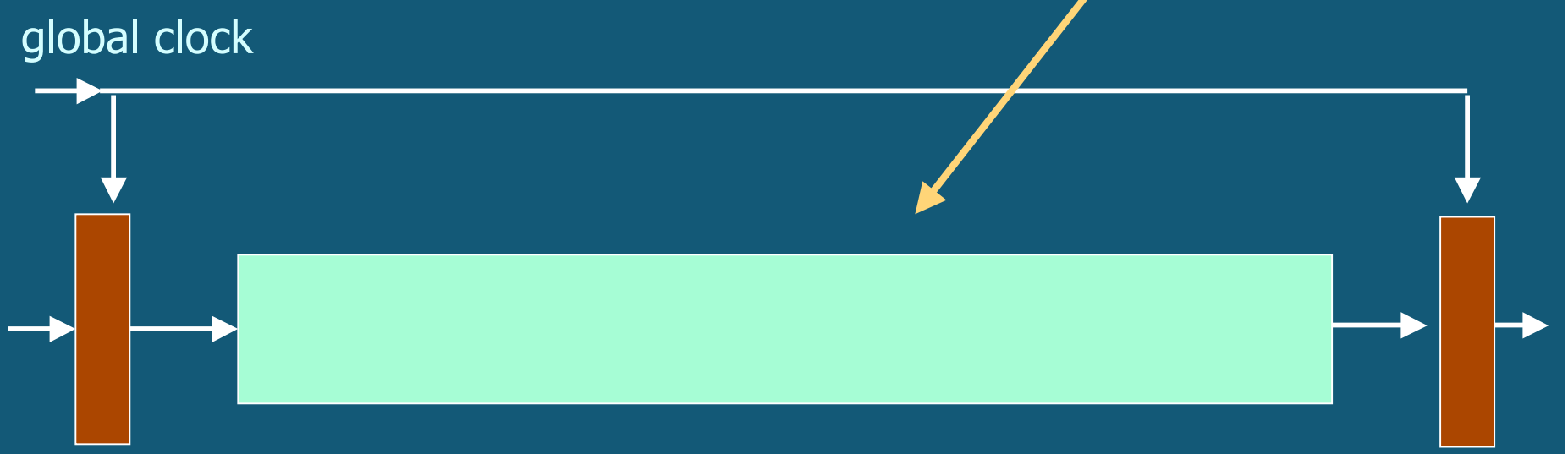
Overview: My Research Areas

- CAD Tools/Algorithms for Asynchronous Controllers (FSM's)
  - "MINIMALIST" Package: for synthesis + optimization
- Mixed-Timing Interface Circuits:
  - for interfacing sync/sync and sync/async systems
- High-Speed Asynchronous Pipelines:
  - for static or dynamic logic

High-Speed Asynchronous Pipelines

NON-PIPELINED COMPUTATION:

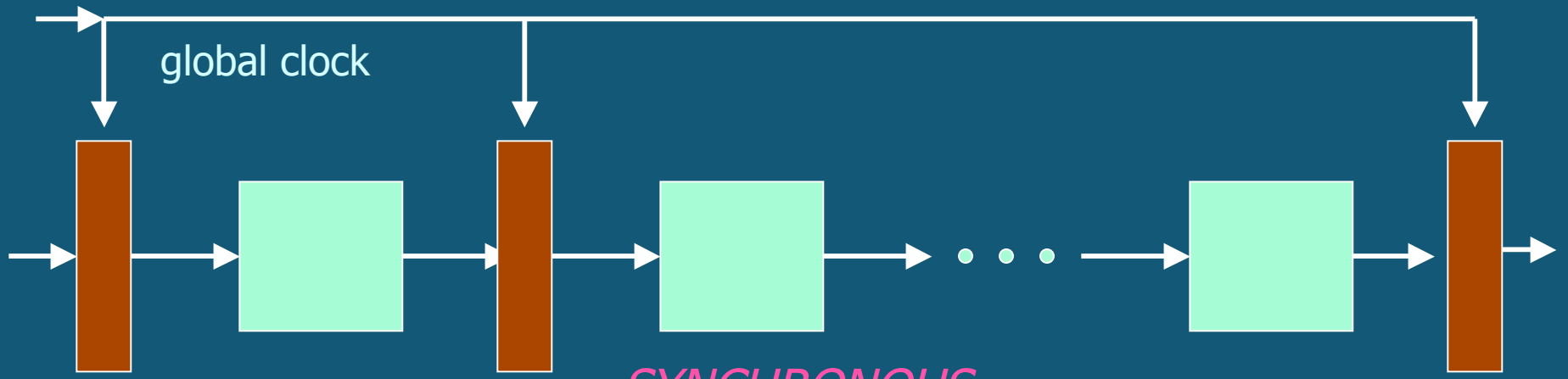
"datapath component" = adder, multiplier, etc.



SYNCHRONOUS

High-Speed Asynchronous Pipelines

"PIPELINED COMPUTATION": like an assembly line



SYNCHRONOUS

no global clock

ASYNCHRONOUS

High-Speed Asynchronous Pipelines

Goal: fast + flexible async datapath components

- speed: comparable to fastest existing synchronous designs
- additional benefits:
  - dynamically adapt to variable-speed interfaces
    - *handles dynamic voltage scaling*
  - "elastic" processing of data in pipeline
  - no requirement of equal-delay stages
  - no high-speed clock distribution
  - multi-GigaHertz performance

Contributions: 3 New Async Pipeline Styles [SINGH/NOWICK]

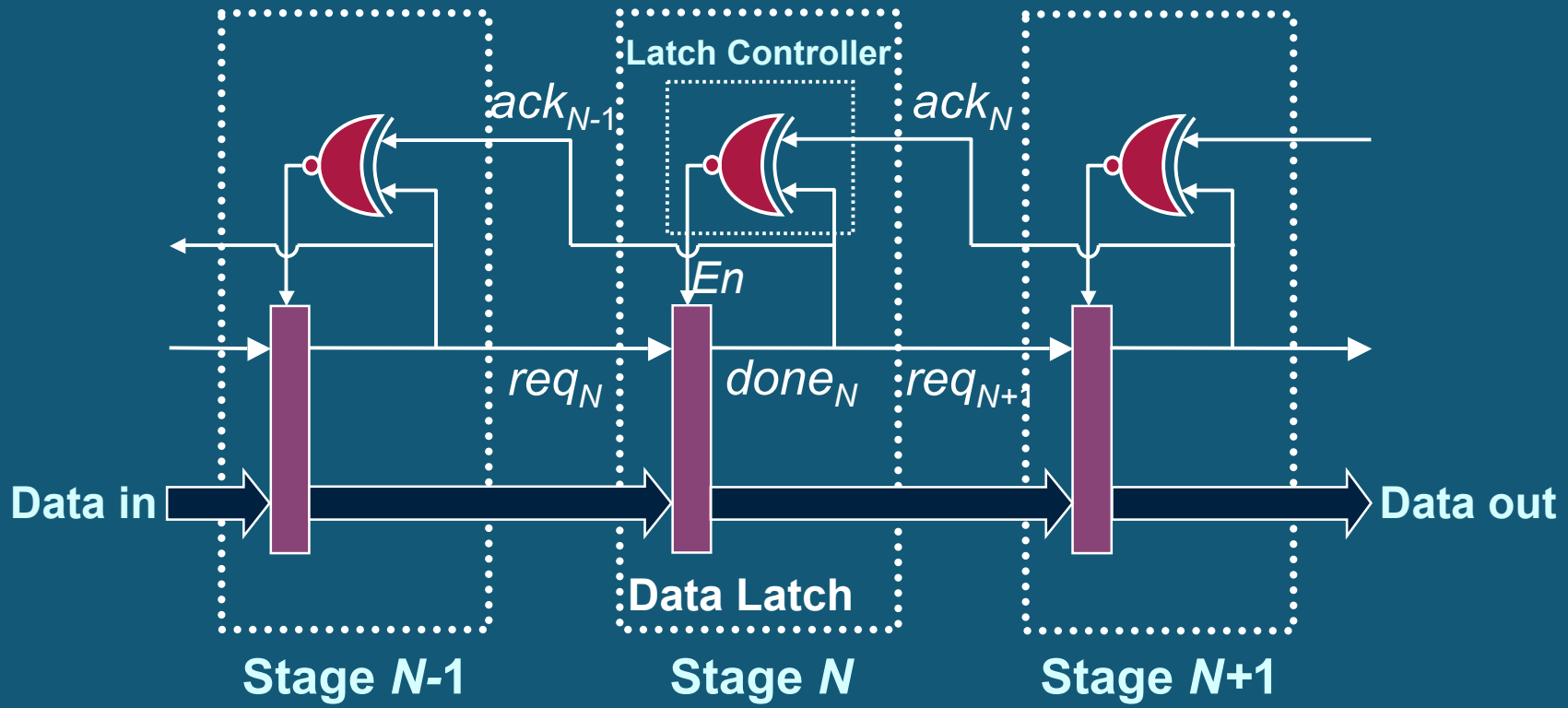
(i) MOUSETRAP:	static logic	[ICCD-01, IEEE Trans. on VLSI Systems '07]
(ii) Lookahead (LP):	dynamic logic	[Async-02, IEEE Trans. on VLSI Systems '07]
(iii) High-Capacity (HC):	dynamic logic	[Async-02, ISSCC-02, IEEE Trans. on VLSI Systems '07]

Application (IBM Research): experimental FIR filter for disk drives [ISSCC-02, Tierno et al.]

- async filter within sync wrapper
- performance: better than best comparable existing commercial synchronous design
- provides "adaptive latency" = # of clock cycles per operation

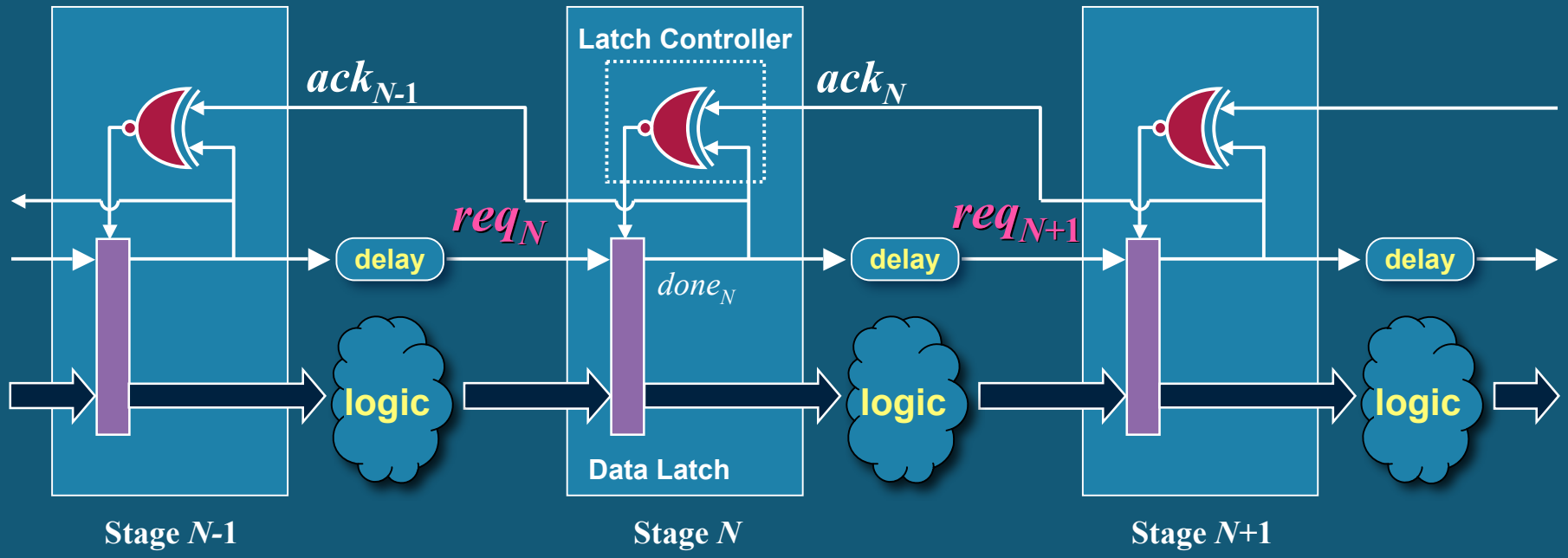
MOUSETRAP: A Basic FIFO (no computation)

Stages communicate using *transition-signaling:*



[Singh/Nowick, IEEE Int. Conf. on Computer Design (2001)]

"MOUSETRAP" Pipeline: w/computation



Function Blocks: use "synchronous" single-rail circuits (not hazard-free!)

"Bundled Data" Requirement:

- each *"req"* must arrive after data inputs valid and stable

Major Recent Research Projects

#1. With NASA (Goddard Space Center): laser space measurement circuits

- Uses our Minimalist CAD tools/circuit styles for async controllers
  - Joint chip design: Nowick + NASA manager
  - Prototype chip #1: back from fab
  - Prototype chip #2: Summer 07

#2. High-Throughput Async Interconnect: for GALS "supercomputer-on-chip"

- Collaboration with parallel architectures/algorithms group: U. of Maryland
  - Goal: very flexible, low-power interconnect = CPU's <--> caches
  - Uses our MOUSETRAP pipelines + mixed-timing interfaces
  - Funding: ~$1M NSF "team" grant (CPA, 2008)

Other Rcent Research: Asynchronous CAD Tools/Algorithms

CAD Tools/Optimizations for Very Robust Async Circuits

- Cheoljoo Jeong

- Collaboration with Orlando-based startup: Theseus Logic
  - Low-power applications
  - CAD tools: *multi-level logic optimization, technology mapping*
  - Circuit improvements: > 40% speed, >20% area reduction
  - Technology transfer: ongoing
  - "ATN-OPT" tool: download site = www1.cs.columbia.edu/~nowick/asynctools

CAD Tools for Async Controller Decomposition

- Melinda Agyekum

- Goal = improved runtime during synthesis
  - CAD tools: partitioning large/complex controllers
  - Over 1000x runtime improvement

Performance Analysis/Optimization of Concurrent Systems

Goal: fast analytical techniques + tools

- to handle large/complex asynchronous + mixed-timing systems

- using stochastic delay models (Markovian): [McGee/Nowick, *CODES-05*]
- using bounded delay models (min/max): *work in progress*

Applications: system-level analysis + optimization

- Large Async Systems:
  - Evaluate latency, throughput, critical vs. slack paths, average-case rating
  - Drive optimization: pipeline granularity, module selection
- Large Heterogeneous (mixed-clock) or "GALS" Systems:
  - Evaluate critical vs. slack paths, buffer requirements
  - Drive optimization: dynamic voltage scaling, load balancing of threads

- Peggy McGee