Advances in Designing Clockless Digital Systems

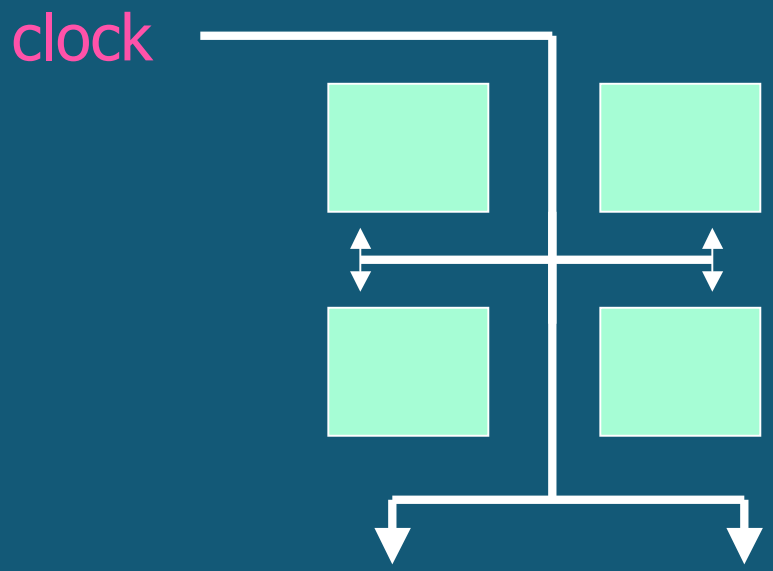
Prof. Steven M. Nowick

nowick@cs.columbia.edu

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

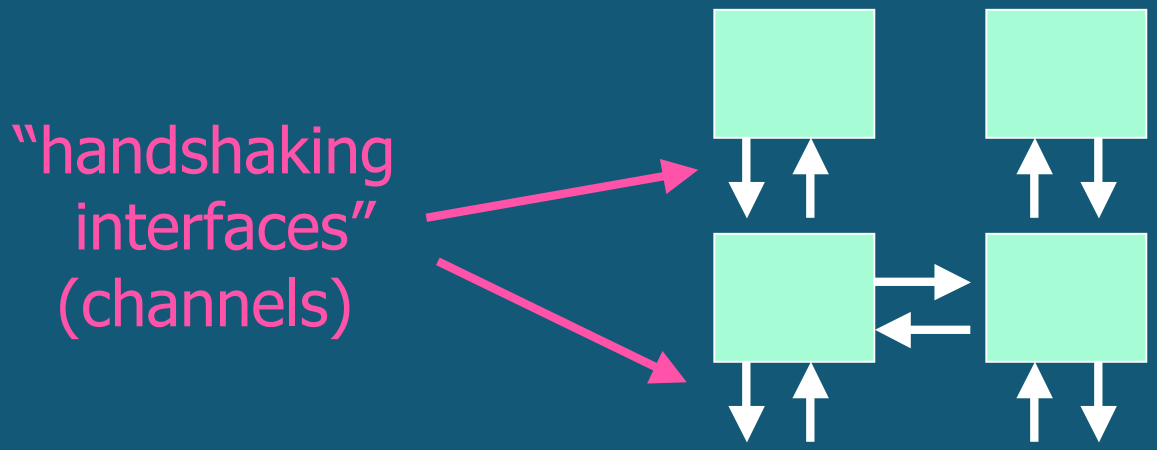
Introduction

- 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*
- *2009:* 3-6 *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*
- *2010 and beyond: 1 billion+*

Design Challenges:

- system complexity, design time, clock distribution
- *clock will 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

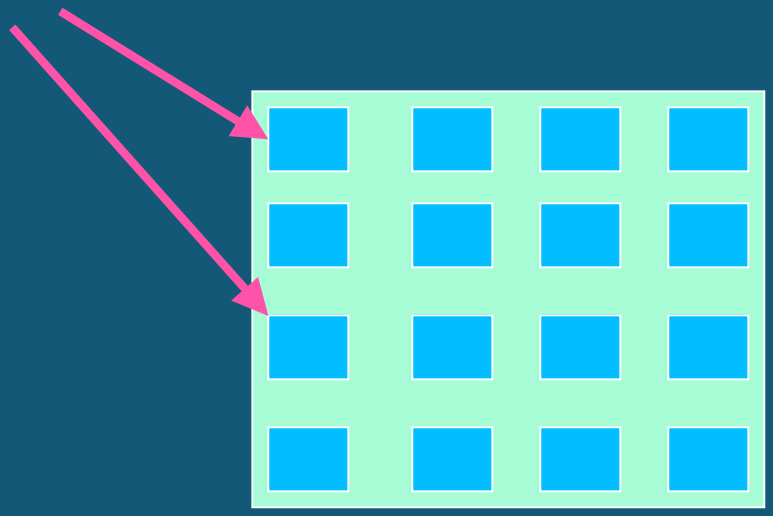
Design Challenge: mismatch with central fixed-rate clock

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 dynamic power only "on demand"
    - ➔ *no 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
  - modular design style ➔ "object-oriented"
- Higher Performance
  - not limited to "worst-case" clock rate
- "Demand- (Data-) Driven" Operation
  - instantaneous wake-up from standby mode

Asynchronous Design: Recent Industrial Developments

1. Philips Semiconductors:

- Wide commercial use: 700 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 commercial CAD tool flow:
  - "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: *high-speed FPGA's*

Asynchronous Design: Potential Targets

Large variety of asynchronous design styles

- Address different points in "design-space" spectrum…
- Example targets:
  - extreme timing-robustness:
    - 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: asynchronous pipelines (with localized timing constraints)
    - … comparable to high-end synchronous
    - with added benefits: support variable-speed I/O rates
  - support for heterogeneous systems: integrate different clock domains + async
    - "GALS-style" (globally-async/locally-sync)

Asynchronous Design: Challenges

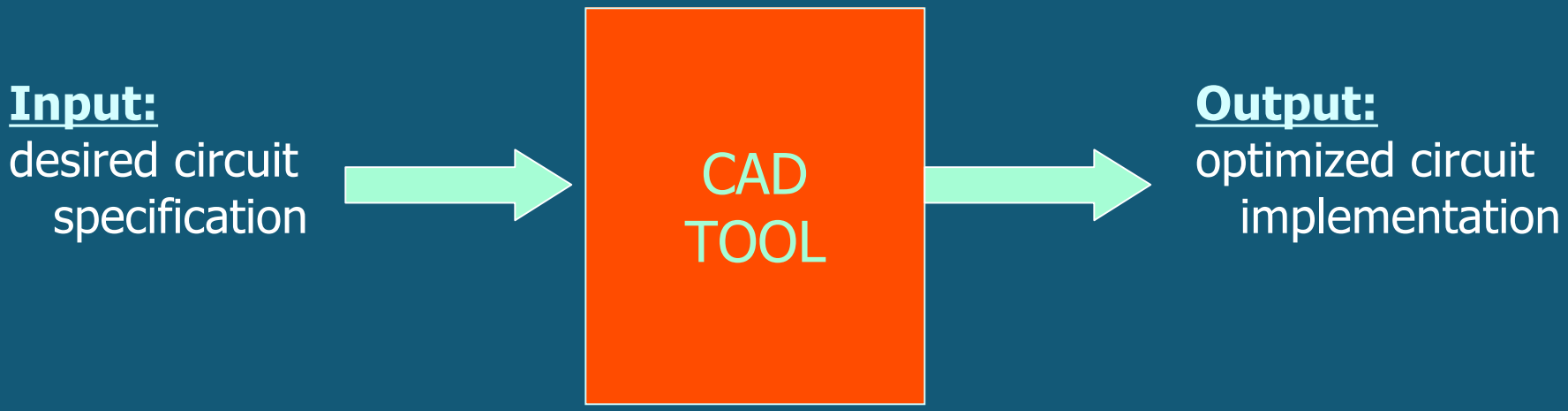
- Critical Design Issues:
  - components must *communicate cleanly:* *'hazard-free' design*
  - *highly-concurrent designs:* much harder to verify!
- Lack of Automated "Computer-Aided Design" Tools:
  - most commercial "CAD" tools targeted to synchronous

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:

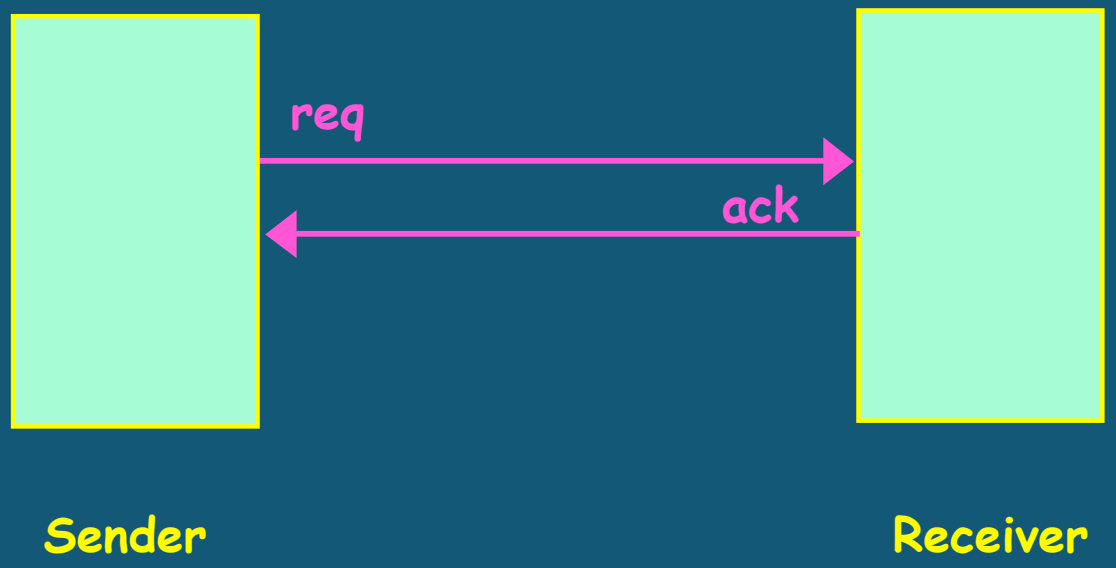
- Most commercial "CAD" tools targeted to synchronous
- Synchronous CAD tools:
  - major drivers of growth in microelectronics industry
- Asynchronous "chicken-and-egg" problem:
  - few CAD tools ←→ less commercial use of async design
  - especially lacking: tools for designing/optmzng. large systems

Overview: Asynchronous Communication

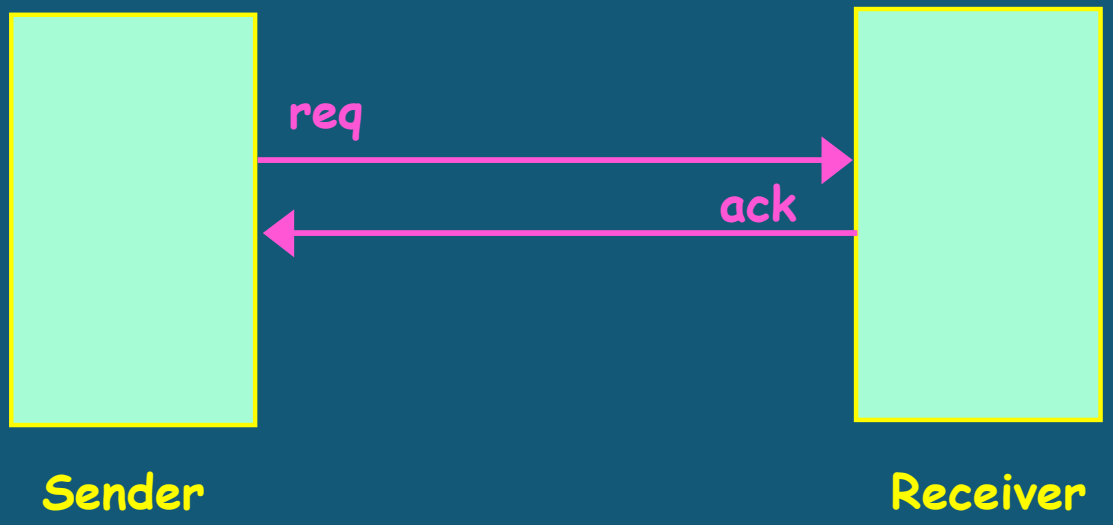
Components usually communicate & synchronize on channels

Overview: Signalling Protocols

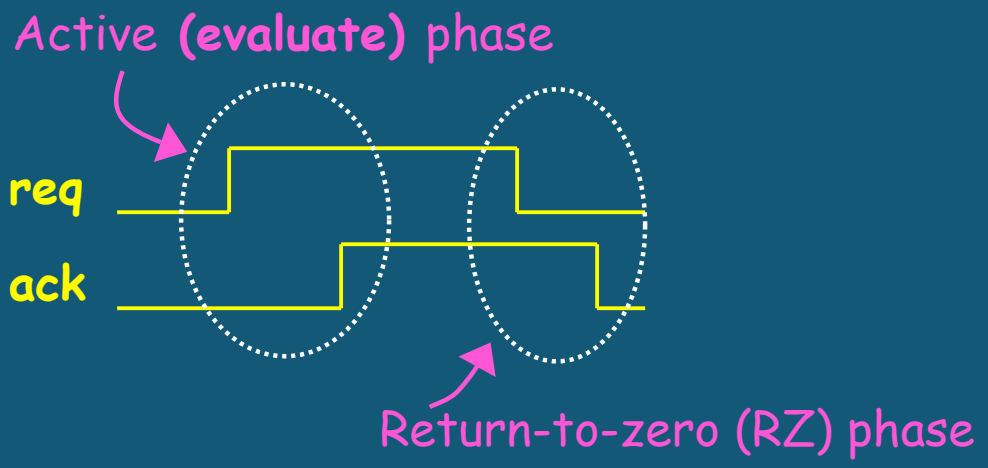
Communication channel: usually instantiated as 2 wires



Overview: Signalling Protocols

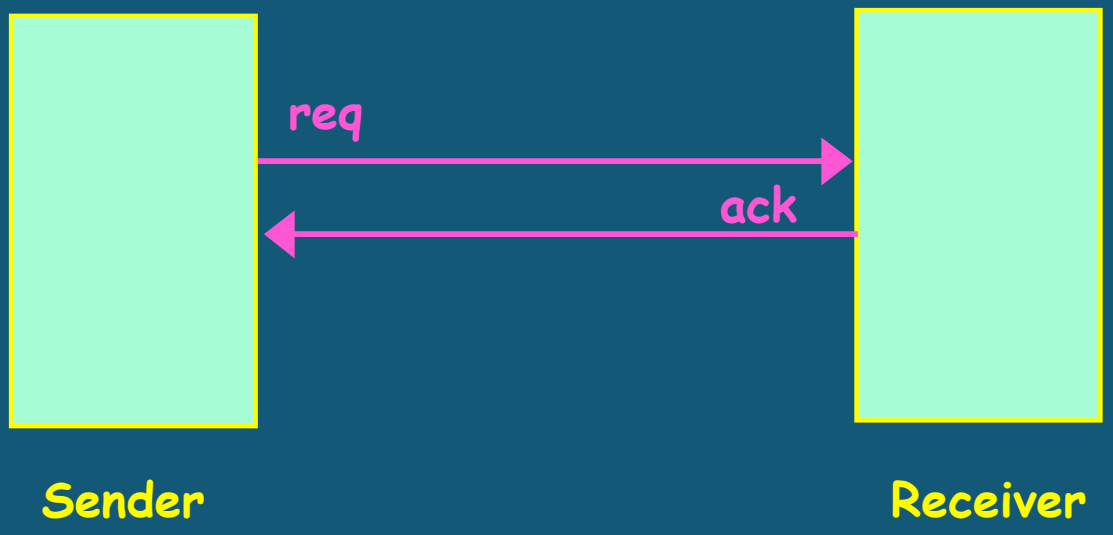


4-Phase Handshaking



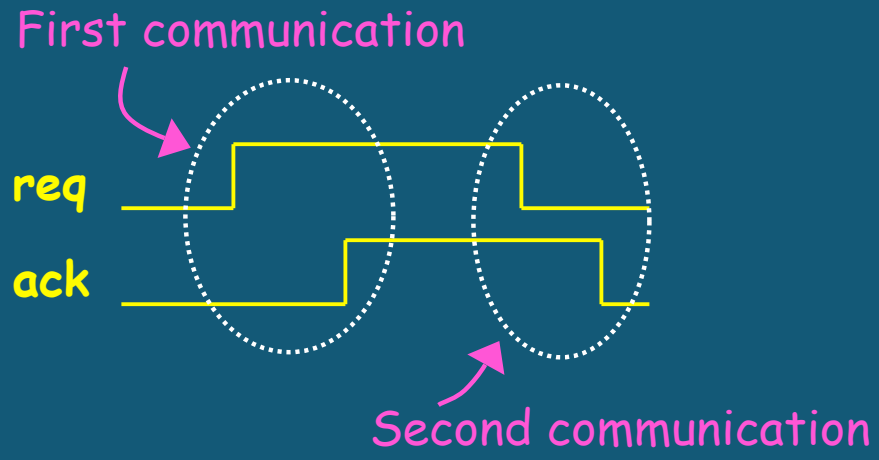
One transaction
(return-to-zero [RZ]):

Overview: Signalling Protocols



2-Phase Handshaking = "Transition-Signalling"

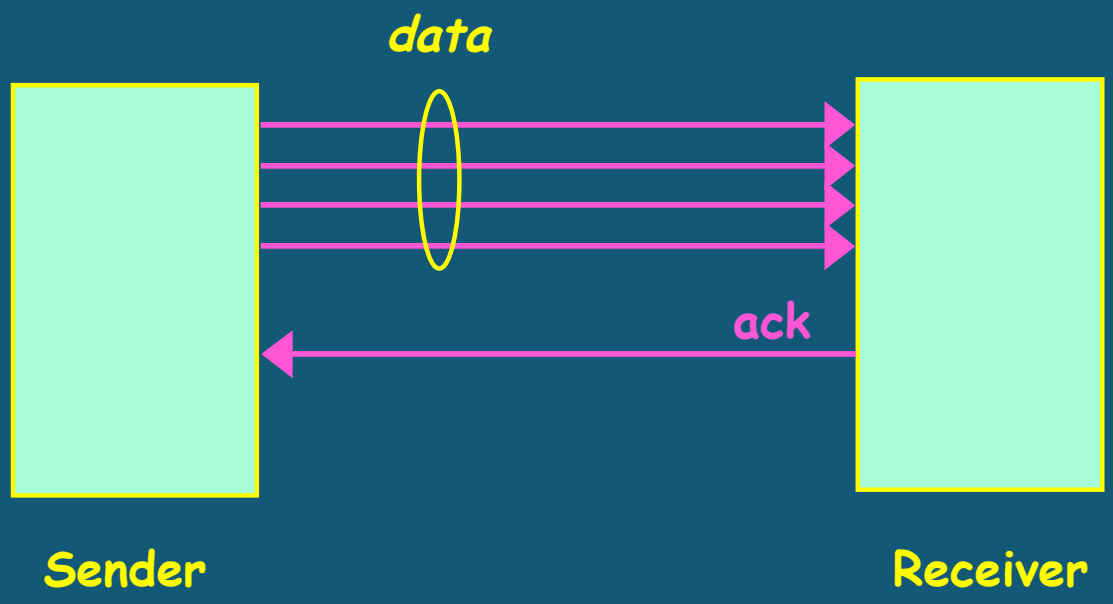
Two transactions
(non-return-to-zero [NRZ]):



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]

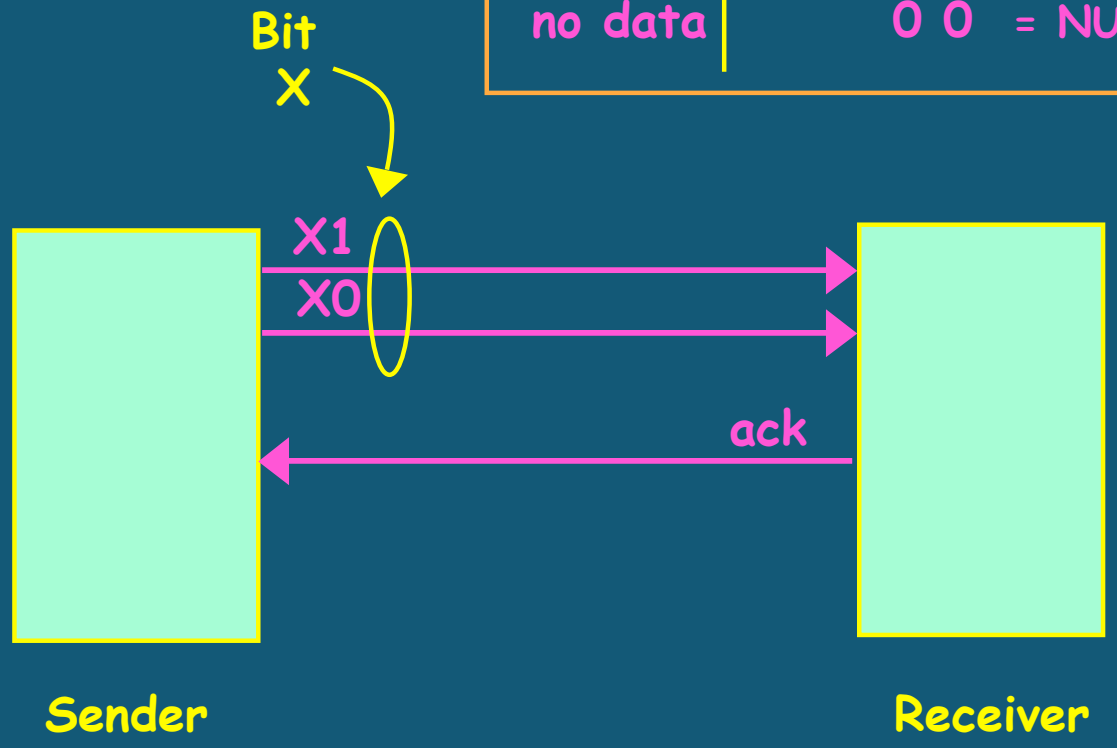
Timing-Dependent Codes: use localized timing assumptions

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

Overview: How to Encode Data?

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

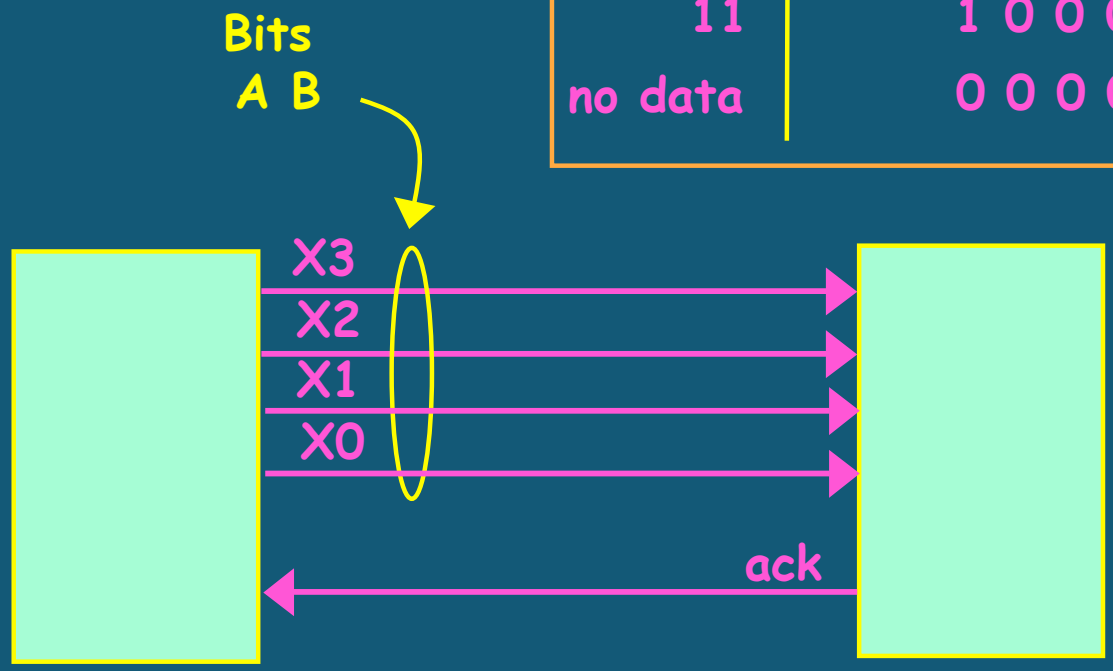
Bit X	Dual-rail encoding X1 X0
0	0 1
1	1 0
no data	0 0 = NULL (spacer)



Overview: How to Encode Data?

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

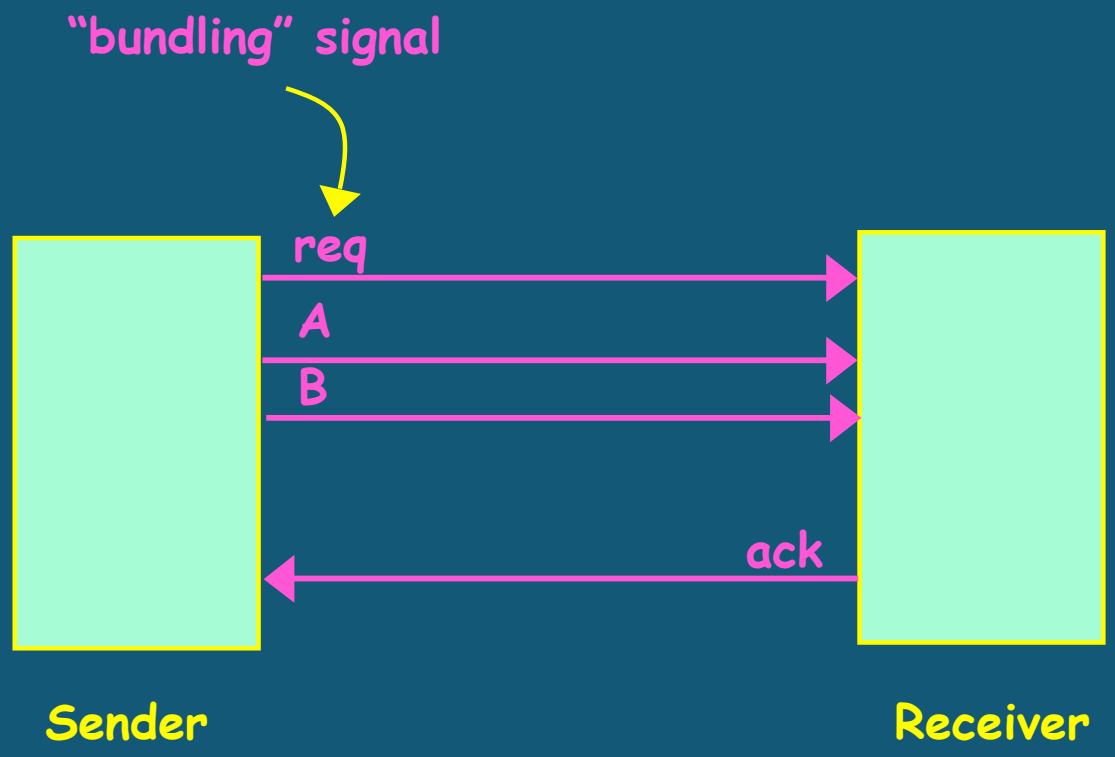
Bits A B	Dual-rail encoding X3 X2 X1 X0
00	0 0 0 1
01	0 0 1 0
10	0 1 0 0
11	1 0 0 0
no data	0 0 0 0 = NULL (spacer)



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]

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

Features:

- Automatic design scripts + custom commands
- Performance-driven multi-level logic decomposition
- Verilog back-end
- Automatic verifier
- Graphical interfaces
- … many optimization modes

Recent application: laser space measurement chip (joint with NASA Goddard)

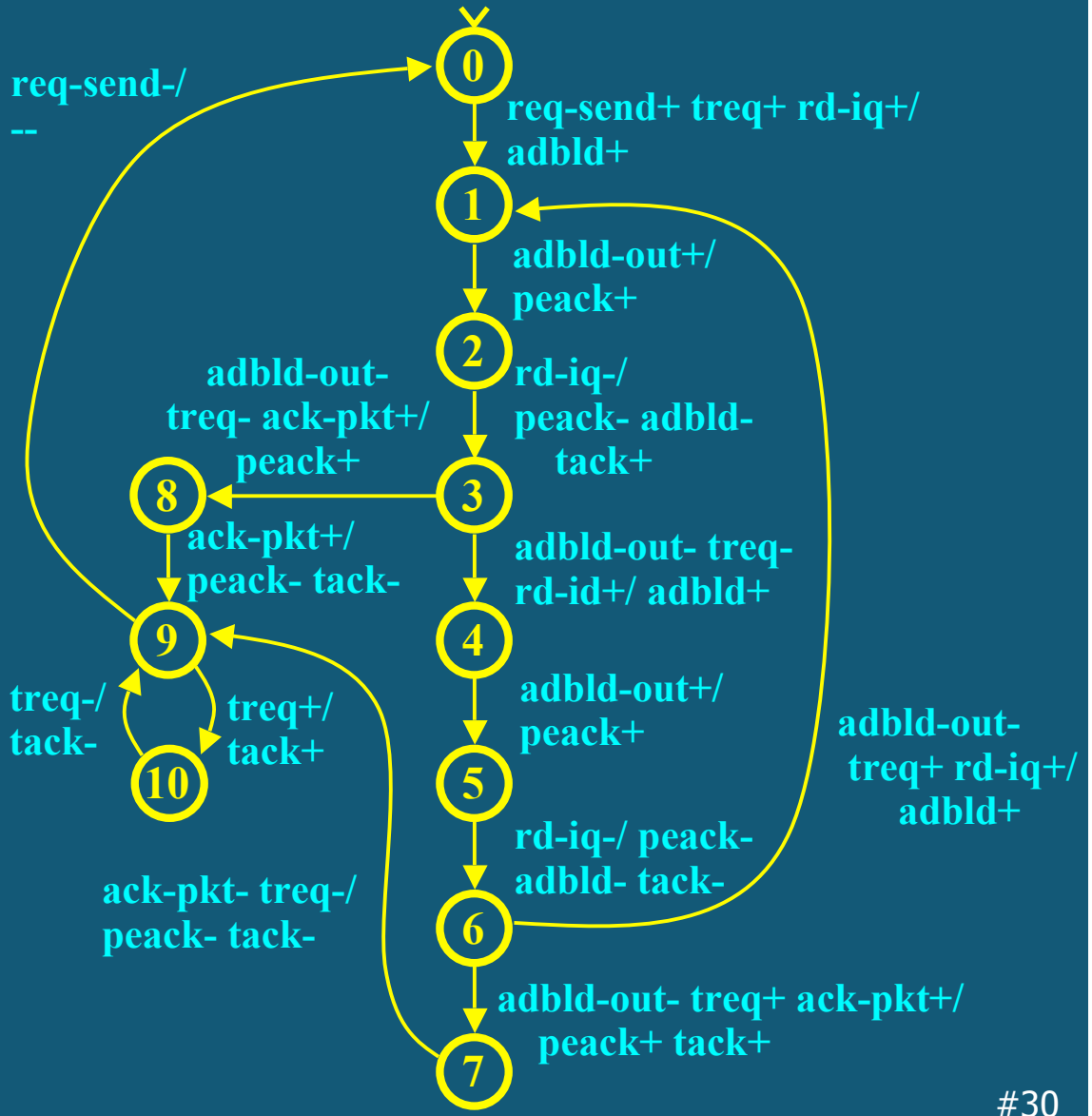
- NASA/Columbia (2006-2007)
- fabricated experimental chip: taped out (Oct. 06)

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

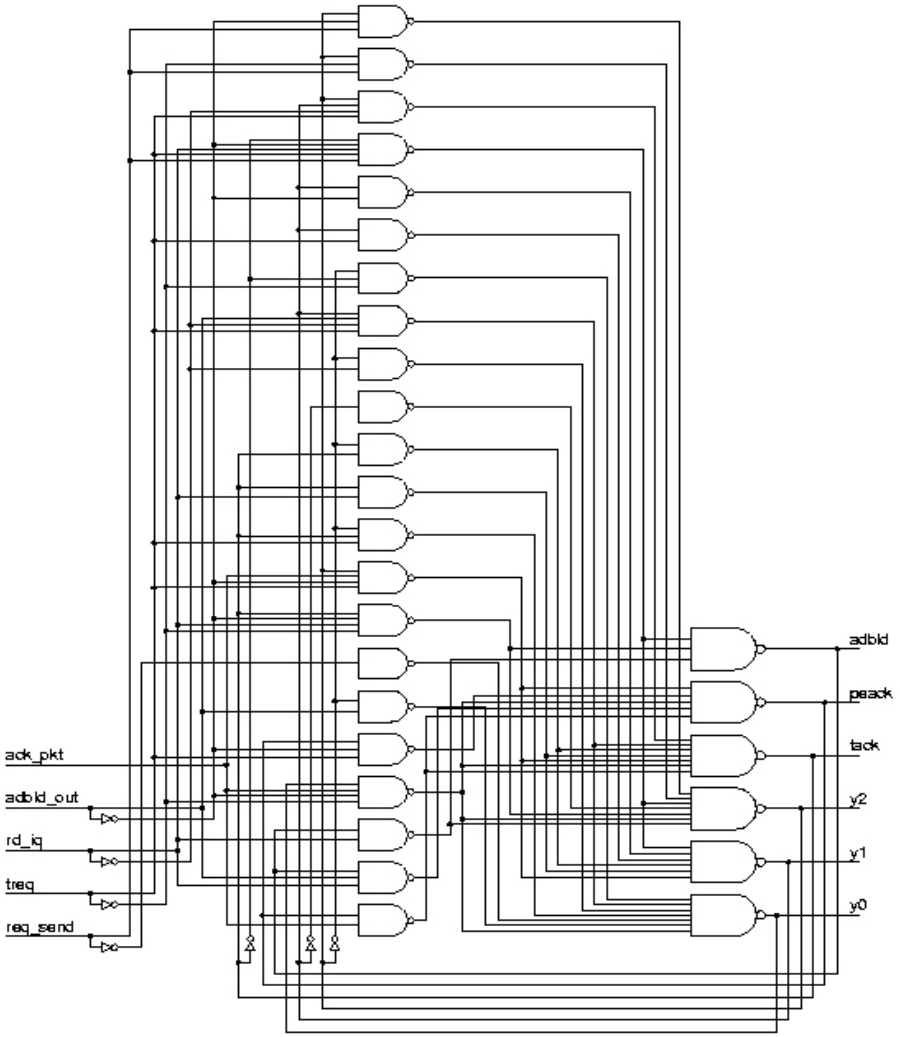


From HP Labs
"Mayfly" Project:
B.Coates, A.Davis, K.Stevens,
"The Post Office Experience: Designing a Large Asynchronous Chip",
INTEGRATION: the VLSI Journal, vol. 15:3, pp. 341-66 (Oct. 1993)

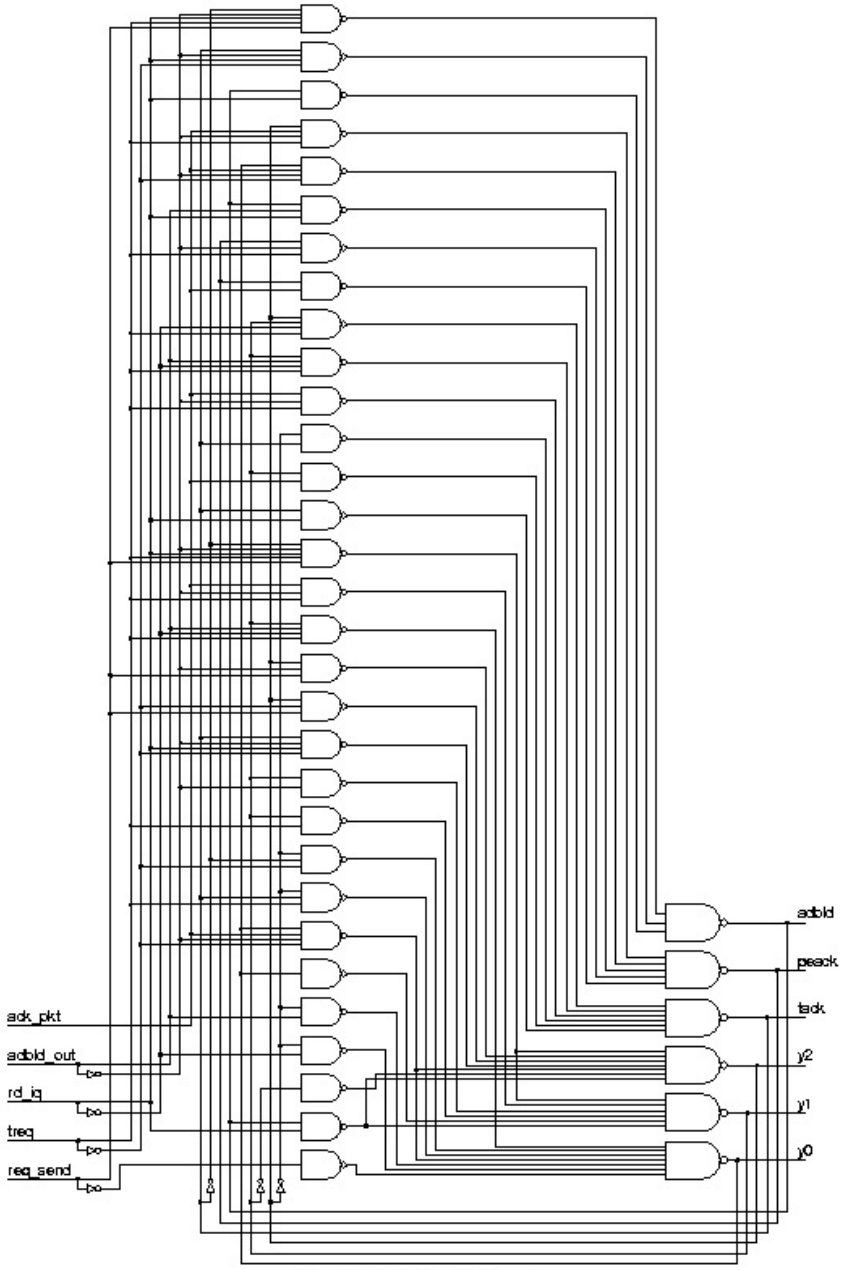
EXAMPLE (cont.):

Design-Space Exploration using MINIMALIST:

optimizing for area vs. speed



PE_SEND_IFC-FL

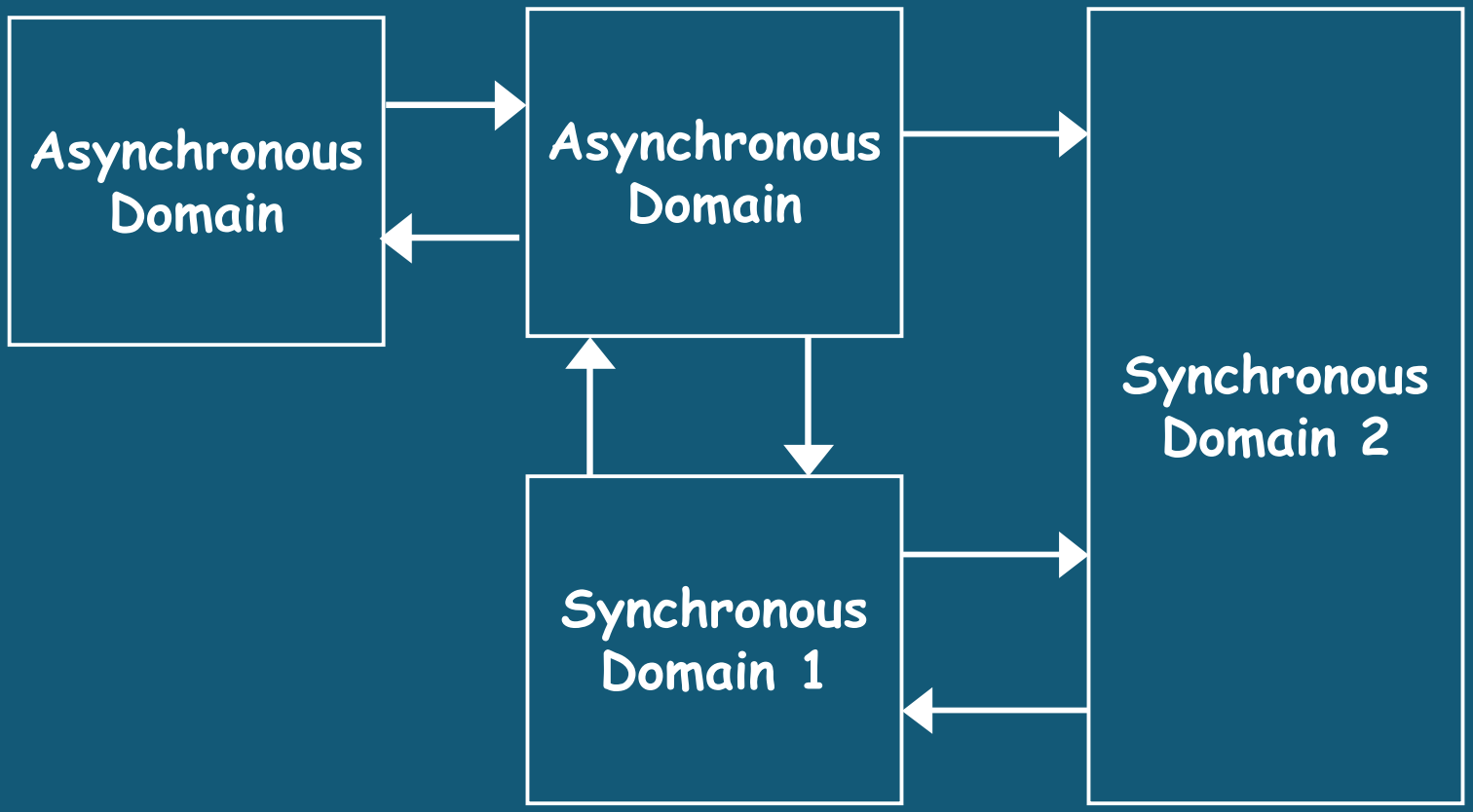


PE_SEND_IFC-Fs

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

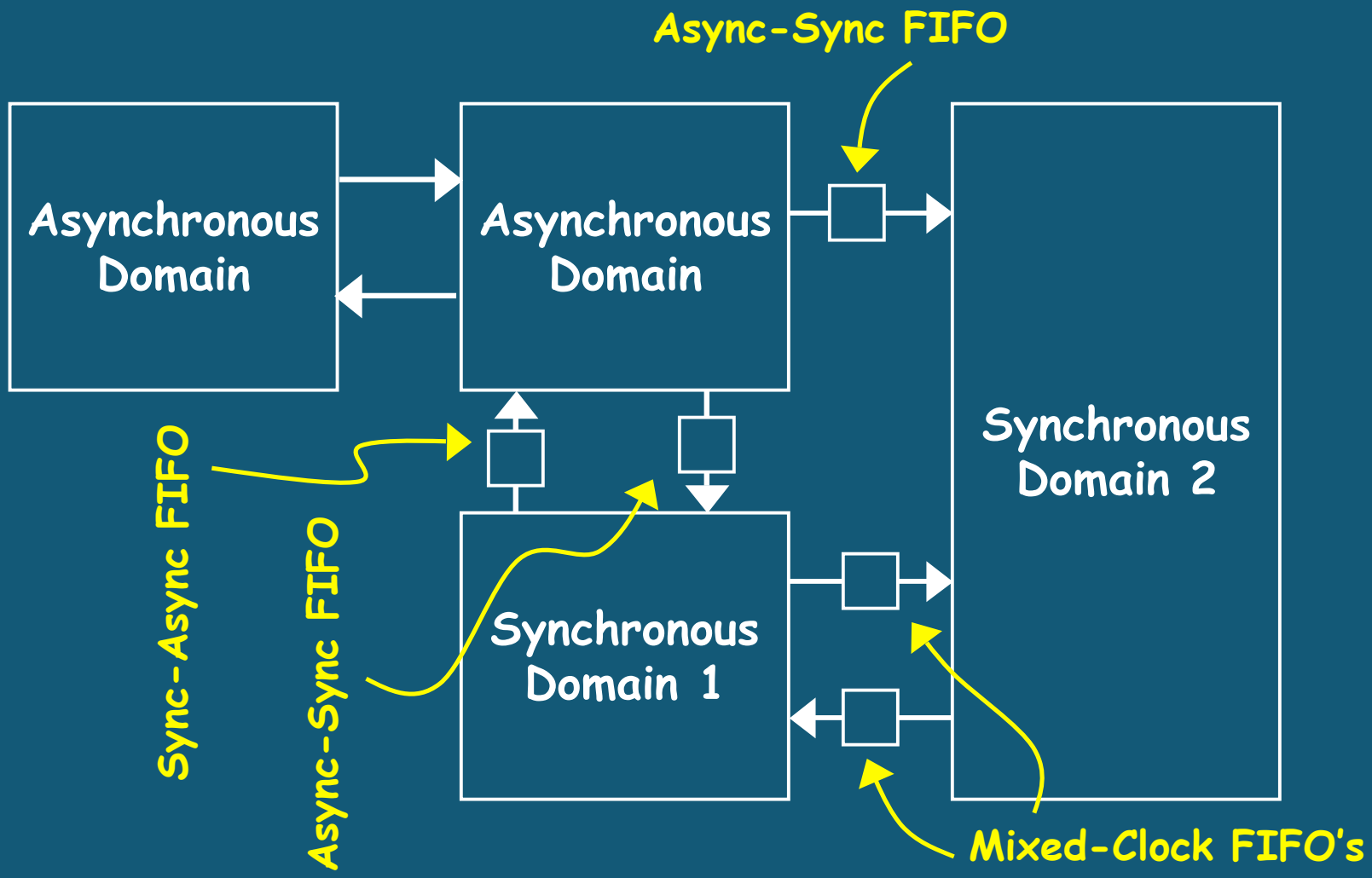
Mixed-Timing Interfaces: Challenge



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]

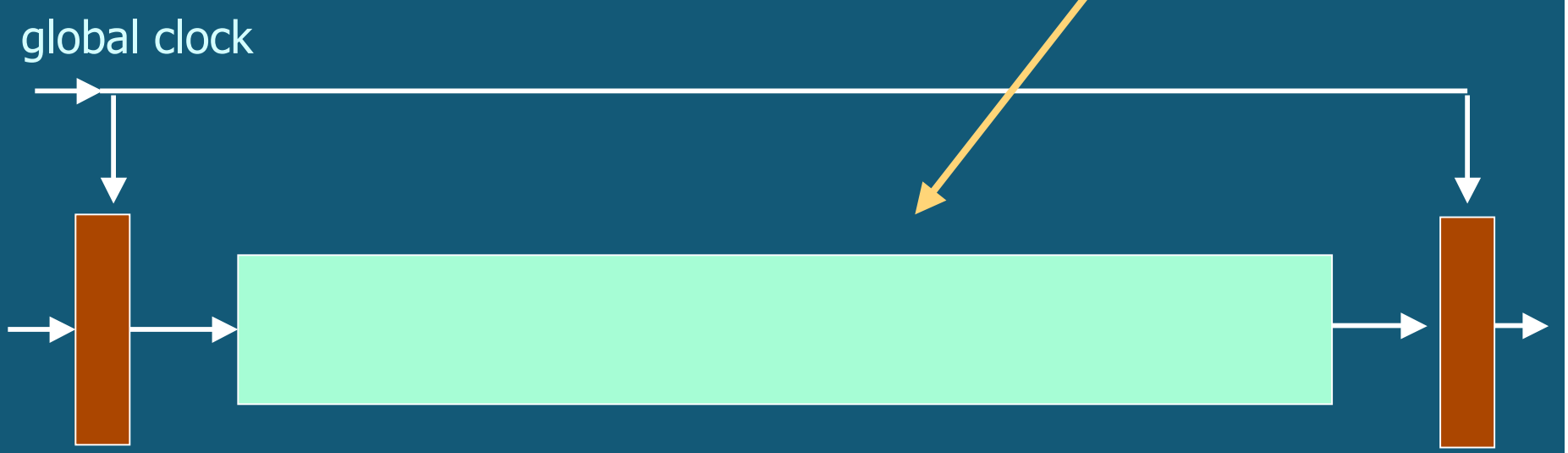
Overview: My Research Areas

- 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:

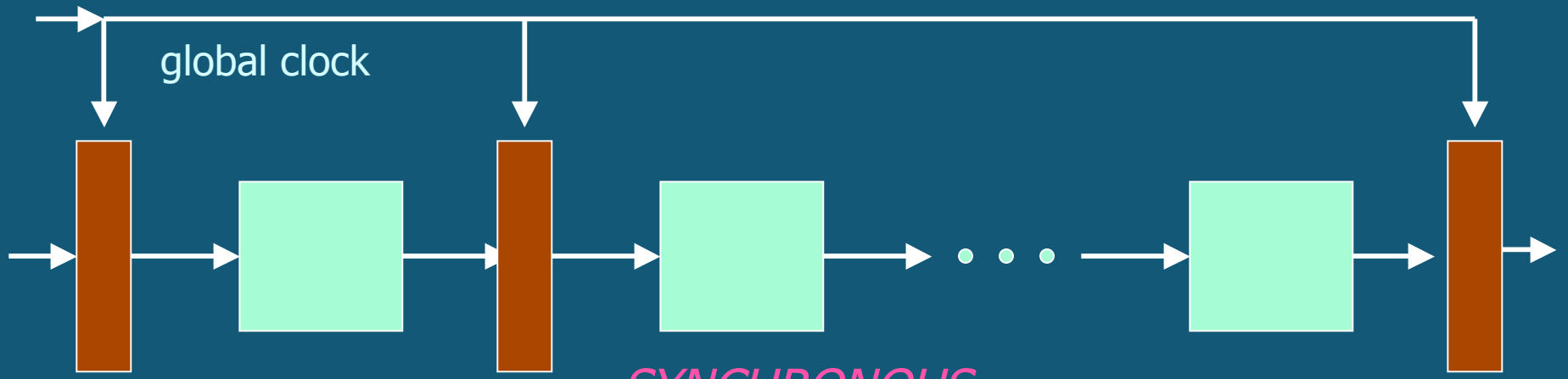
"datapath component" = adder, multiplier, etc.



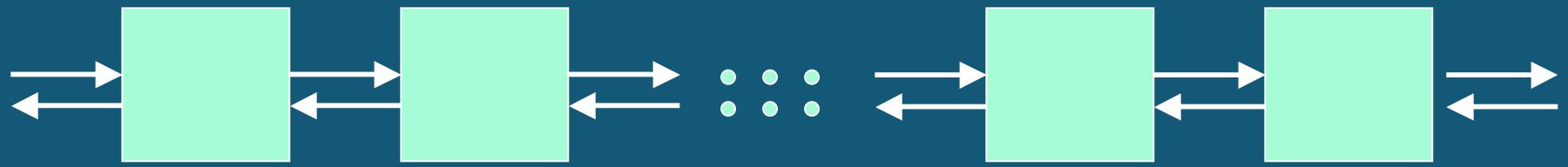
SYNCHRONOUS

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*
  - no requirement of equal-delay stages
  - no high-speed clock distribution

Contributions: 3 New Asynchronous Pipeline Styles [M. Singh/S.M. Nowick]

(i) MOUSETRAP: static logic [ICCD-01, IEEE Trans. on VLSI Systems 2007]

(ii) Lookahead (LP): dynamic logic [Async-02, IEEE Trans. on VLSI Systems 2007]

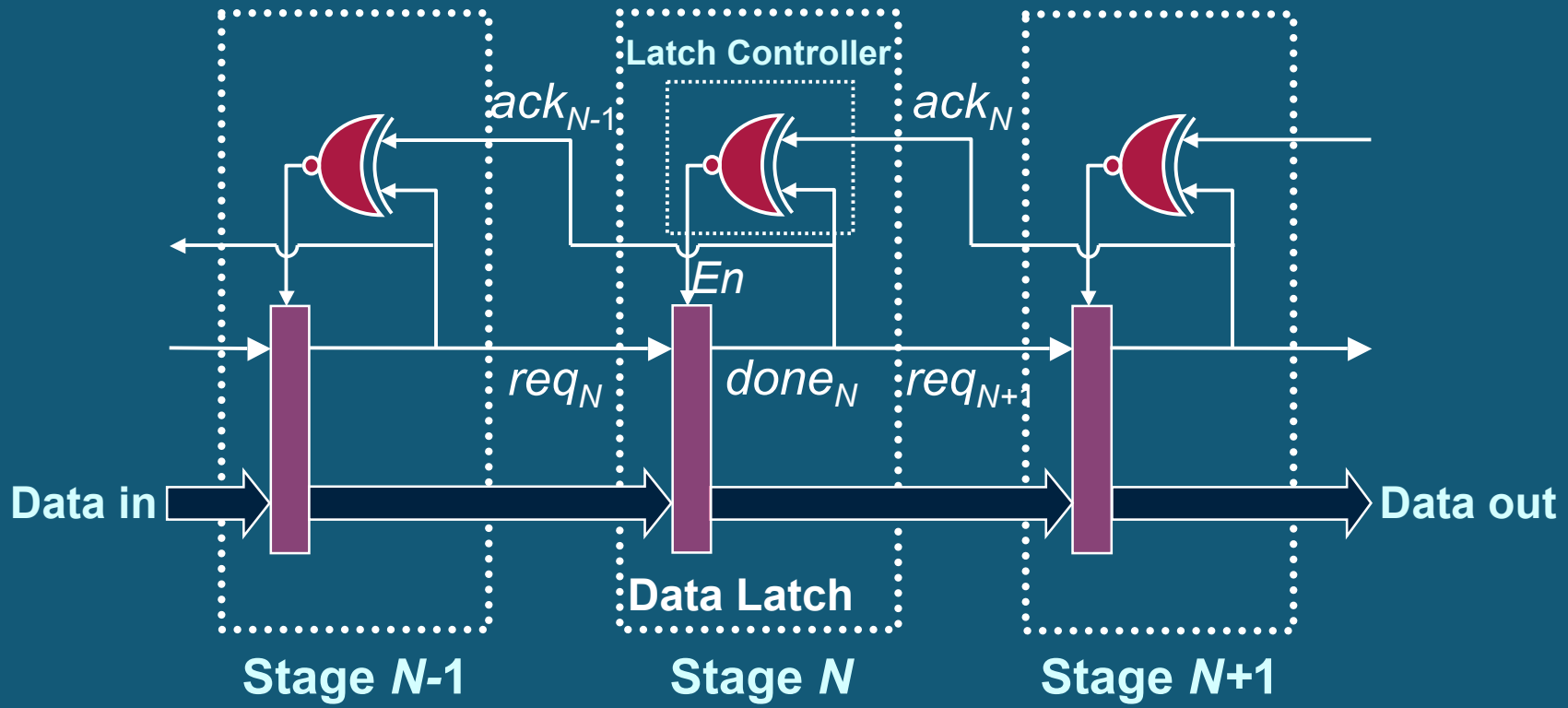
(iii) High-Capacity (HC): dynamic logic [Async-02, ISSCC-02, IEEE Trans. on VLSI Systems 2007]

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

- async filter in sync wrapper
- provides "adaptive latency" = # of clock cycles per operation
- performance: better than leading comparable commercial synchronous design (from IBM)

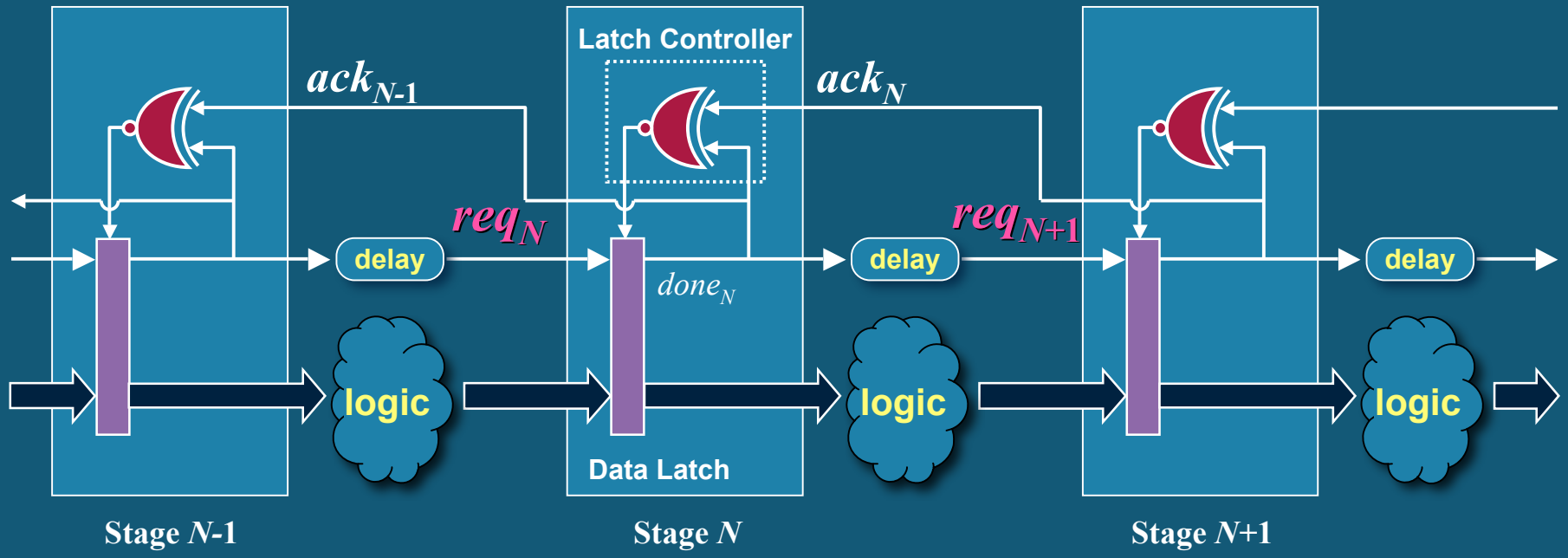
MOUSETRAP: A Basic FIFO (no computation)

Stages communicate using *transition-signaling:*



[Singh/Nowick, IEEE Int. Conf. on Computer Design (2001), IEEE Transactions on VLSI Systems (2007)]

"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

Other Research Projects

1. Asynchronous Interconnection Networks: for Shared-Memory Parallel Processors
   - Medium-scale NSF project [2008-12]: with Prof. Uzi Vishkin (University of Maryland)
   - Goal: low-power/high-performance async routing network (processors <=> memory)
     - "GALS"-style: globally-asynchronous/locally-synchronous
   - [M. Horak, S.M. Nowick, M. Carlberg, U. Vishkin, ACM NOCS-10 Symposium]

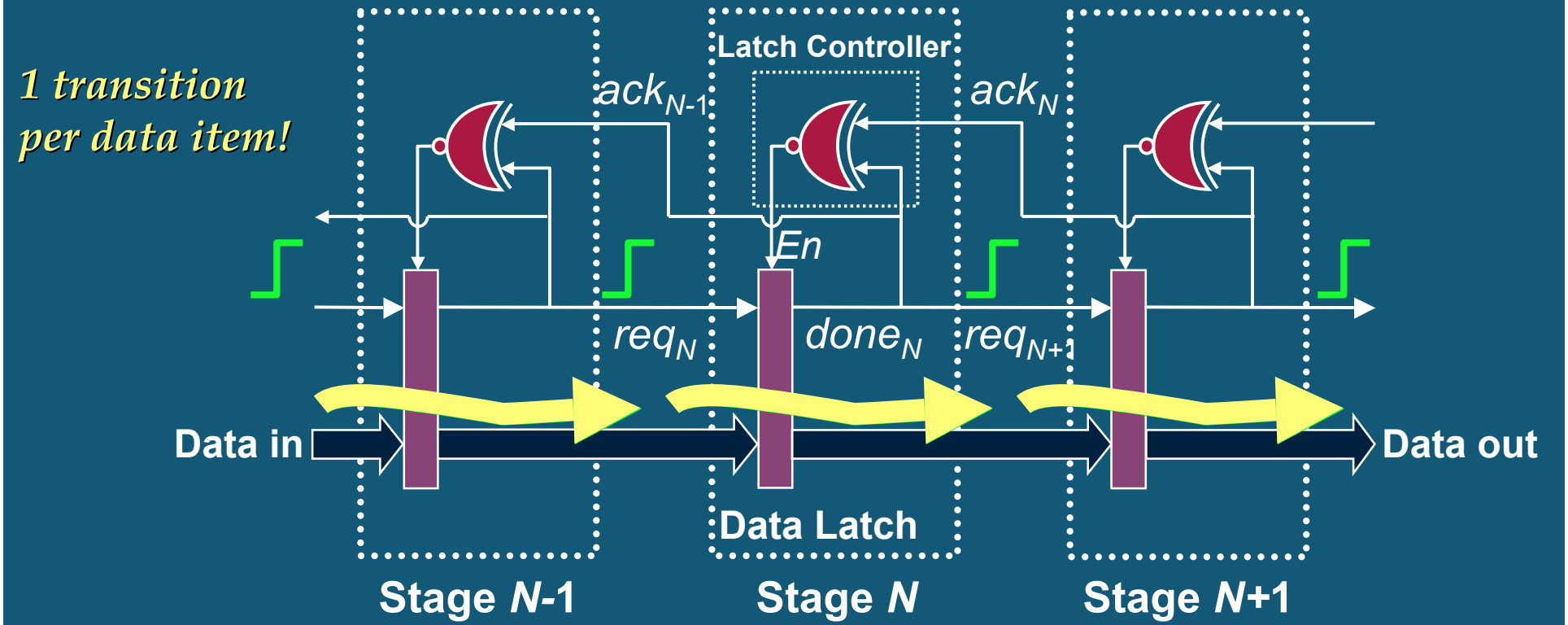
2. Continuous-Time DSP's
   - Medium-scale NSF project [2010-14]: with Prof. Yannis Tsividis (Columbia EE Dept.)
   - Idea: adaptive signal processing, based on signal rate-of-change
   - Goal: low-aliasing + low-power -- combine analog + async digital

3. Asynchronous Bus Encoding: for Timing-Robust Global Communication
   - Goal: low-power, error-correction + timing-robust ("delay-insensitive") communication
   - [M. Agyekum/S.M. Nowick, DATE-10, IWLS-10, DATE-11]

4. Variable-Latency Functional Units: "Speculative Completion"
   - Goal: high-performance components with 'data-dependent' completion
   - [S.M. Nowick et al., IEE Proceedings '96; IEEE Async-97 Symposium]

MOUSETRAP: A Basic FIFO

Stages communicate using *transition-signaling:*



One Data Item

Performance Analysis of Concurrent Systems

Goal: fast analytical techniques + tools

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

- using stochastic delay models (Markovian): [P. McGee/S.M. Nowick, *CODES-05*]
- using bounded delay models (min/max): [P. McGee/S.M. Nowick, *ICCAD-07*]

Applications: analysis + optimization

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

Introduction to MLO

- MLO is an integrated **post-processing** (i.e. backend) tool for Minimalist.
- Targeted to **multi-level logic**.
  - In contrast, Minimalist currently is targeted to two-level logic.
- Designed to work on **combinational hazard-free logic** for Burst Mode controllers.
  - Uses "hazard-non-increasing" transforms.
- Output of MLO is Verilog.
- MLO is a standalone tool running from the Linux shell **outside of Minimalist.**

Minimalist: MLO (Multi-Level Optimizer)

- Accessible on the web from:

- Initial Release
  - One version – for Linux Distributions

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

- Includes
  - Complete Tutorial
  - Documentation
  - Examples
- Tool requires Python interpreter to run:

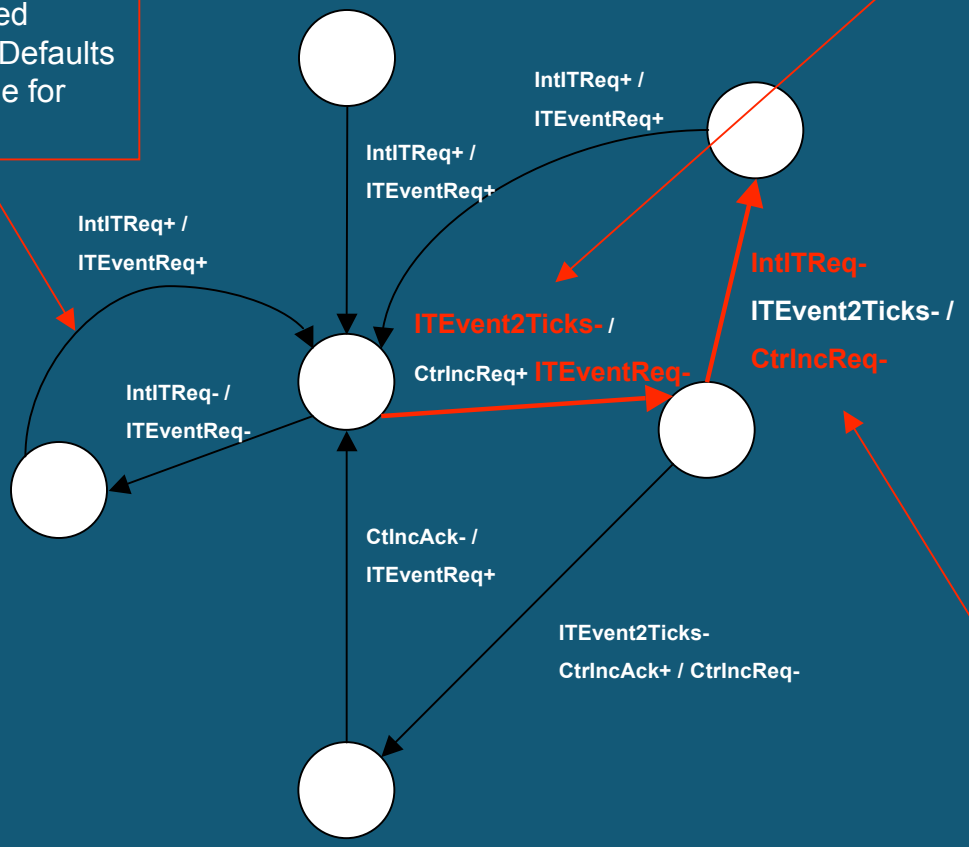
- Consult README for MLO installation information

http://www.python.org/download/

CEO Feature - User-Specified Critical Events

User-Specified Critical Arcs Highlighted in Red

Case 1: Non colorized arc. User-Specified nothing is critical. Defaults to automated mode for every output.

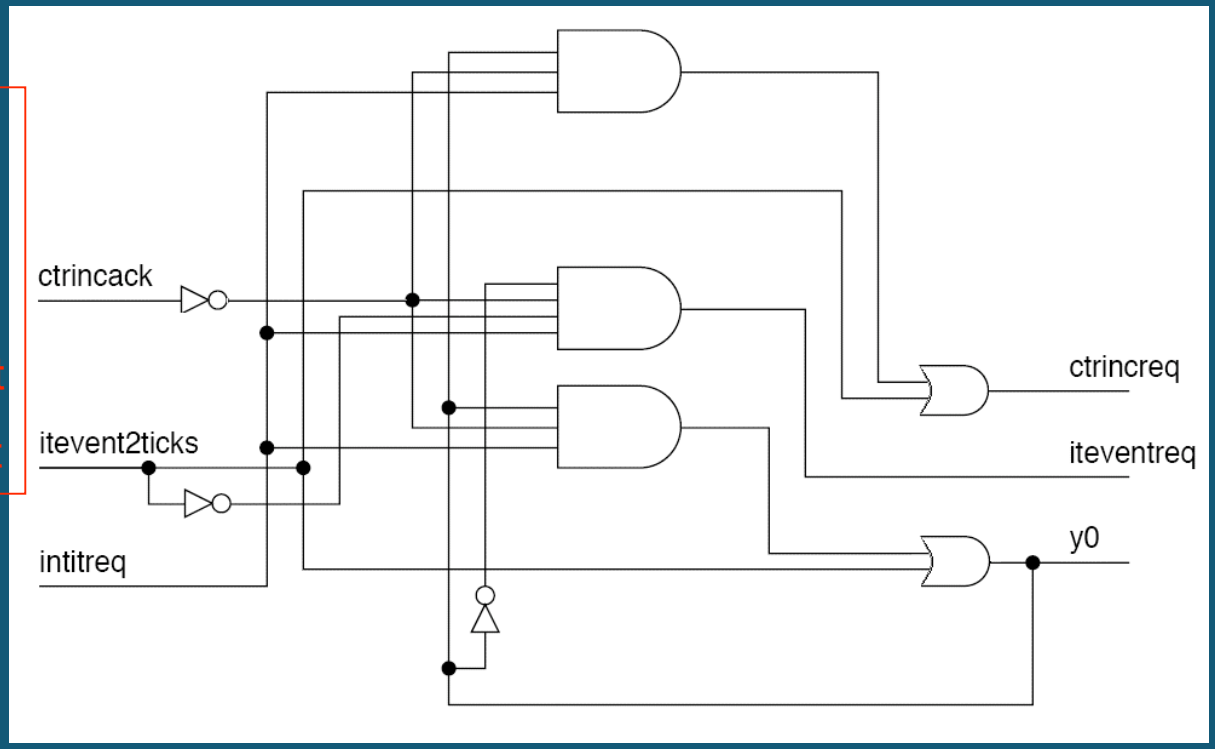


Case 2: Some outputs colorized, some outputs not. Both user-specified data and automated approaches are used to determine criticality. **ITEventReq** will use user-specified data to determine criticality. **CtrincReq** will default to automated mode to determine criticality.

Case 3: Every output is colorized. Automated approach is never used. **IntITReq-** is critical with respect to **CtrIncReq-**, while **ITEvent2Ticks-** is NOT critical to **CtrIncReq-**.

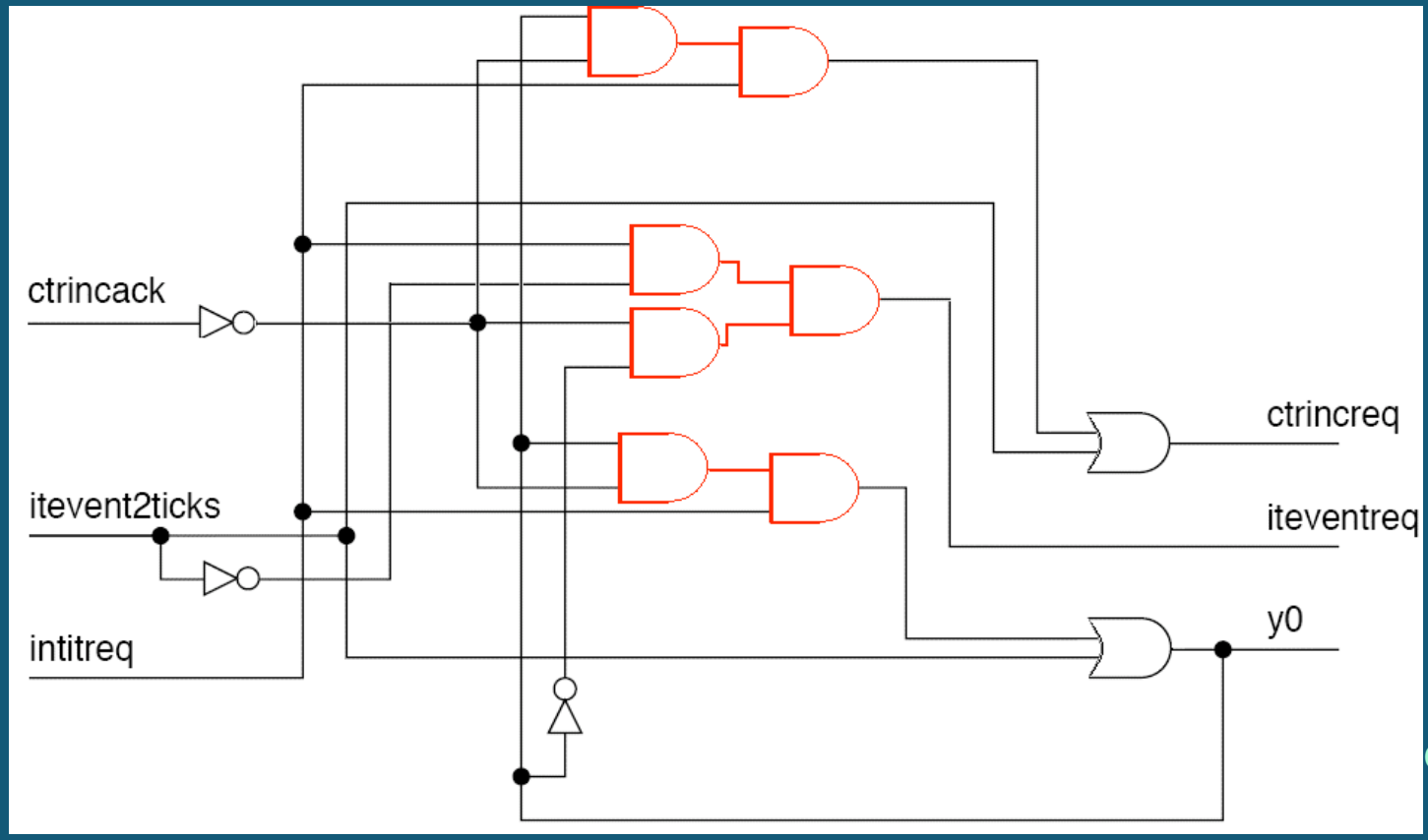
Feature Set - Initial Two-Level Implementation (before applying MLO)

The next four slides present different MLO output examples. For each example, the starting circuit (input to MLO) is this circuit



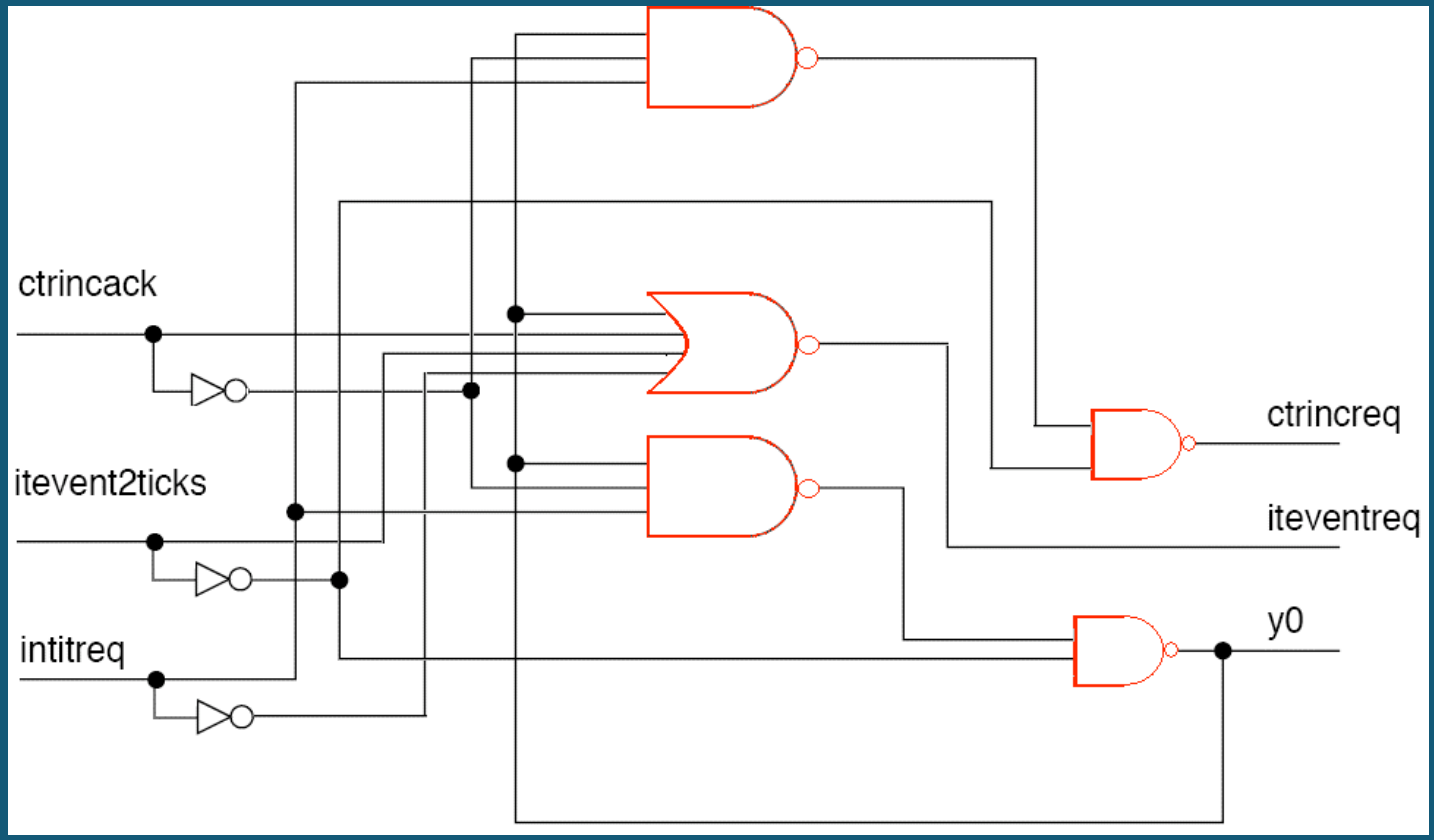
Two-level Structure from Minimalist Output

Feature Set Example 1 - Gate Fan-in Limitation



of 2

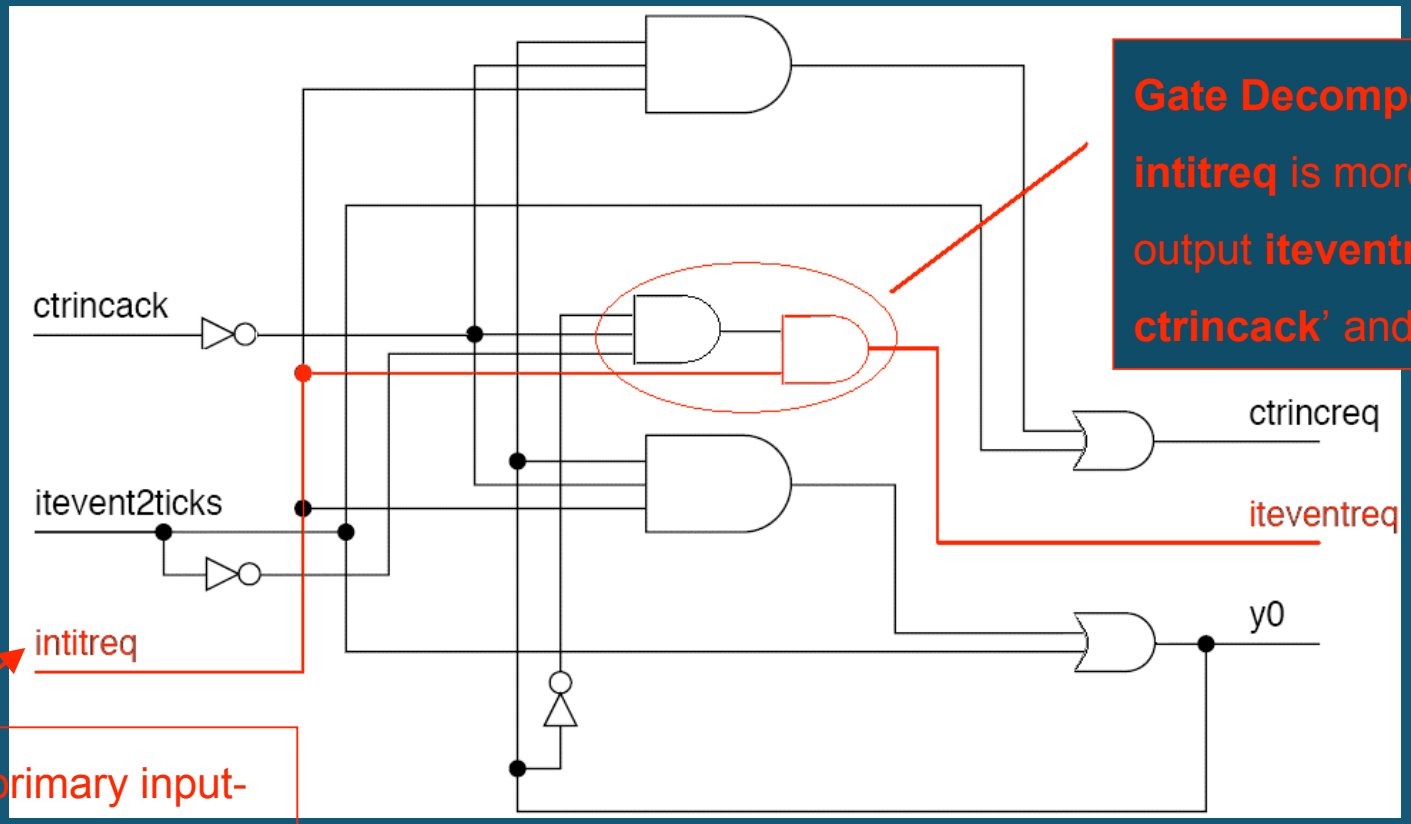
Feature Set Example 2 - Negative Logic



Result of MLO: Multi-Level Circuit using MLO Negative Logic

This mode carefully optimizes only hazard non-increasing safe transformations (DeMorgan's Law). Optimizations are also included to carefully eliminate extra inverters.

Feature Set Example 3 - CEO "critical event optimizer"

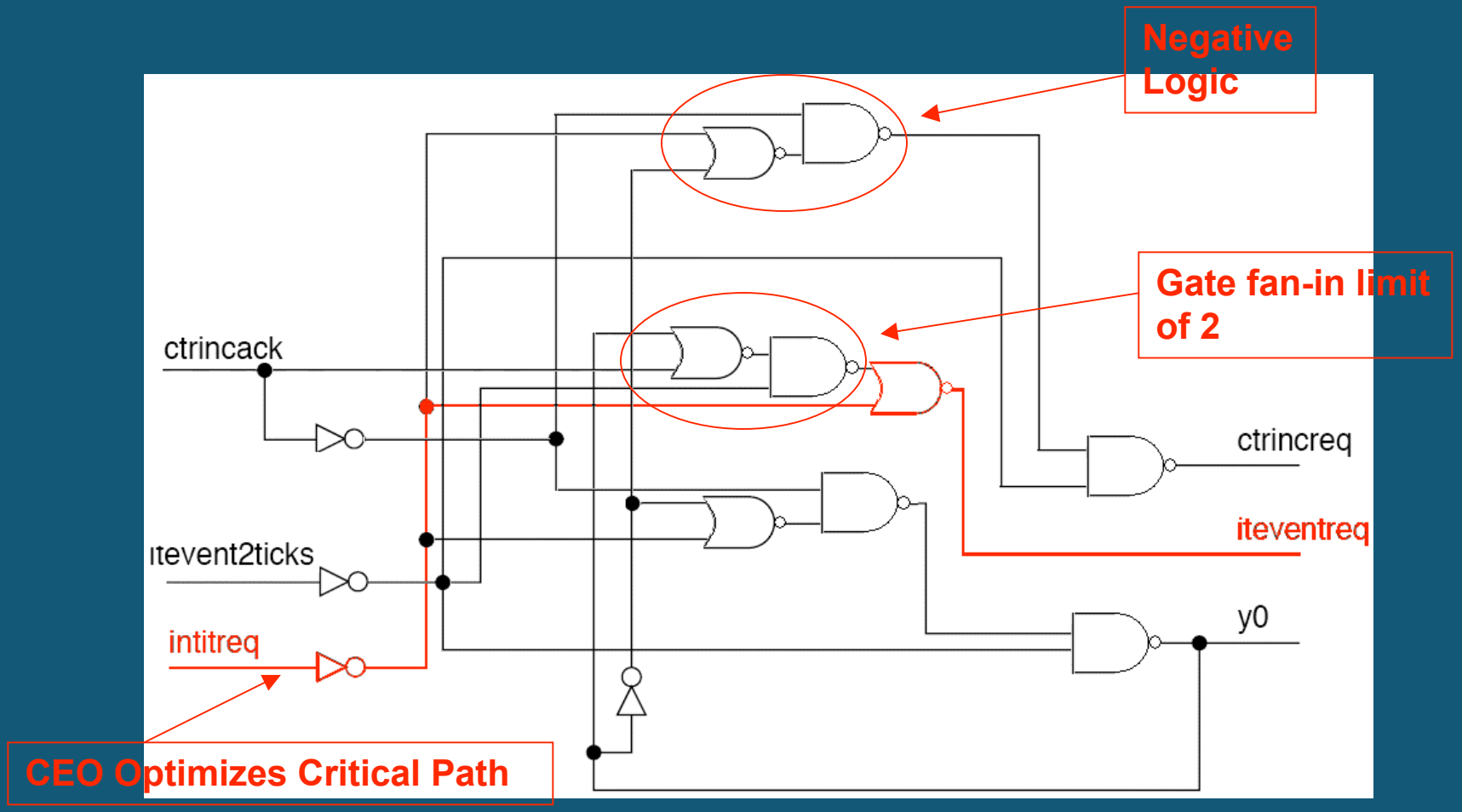


Gate Decomposed. Input **intitreq** is more critical to output **iteventreq** than **ctrincack**' and **y0**'

critical primary input-to-output path

Result of MLO: Multi-Level Circuit after MLO CEO is used

Feature Set Example 4 - Combined



Result of MLO: Multi-Level Circuit with negative logic, AND gate fan-in limit of 2, and CEO.