Digital Systems Area

Stephen E. Edwards
Steven M. Nowick
Stephen Unger

Outline

- Overview of Computer Engineering: Prof. Nowick
- The EE Side of Computer Engineering: Prof. Zukowski
- Research Overview: Prof. Edwards
- Research Overview: Profs. Nowick/Unger

Overview of Computer Engineering

- Interdisciplinary Program: between CS & EE Depts.
- Current Focus: Digital Systems area
- Pending:
  - 2 separate "tracks": (a) Digital Systems; (b) Networking
- History:
  - BS Program: started 1994
  - MS Program: being approved -- to start in Fall 2003
    - University Senate's Education Committee: *approved last week*
    - full University Senate: *being approved today*

People

- CS:
  - Stephen Edwards: embedded systems/languages
  - Steven Nowick: asynchronous digital design, CAD tools
  - Stephen Unger: asynchronous digital design
- EE:
  - Andrew Campbell: telecom/mobile systems
  - Ken Shepard: high-speed VLSI circuits, bio-chips
  - Charles Zukowski: digital circuits, low power

Undergraduate Majors

Recent # of majors (juniors + seniors):

	Fall/Spring
1998-99:	41/36
1999-00:	54/53
2000-01:	68/67
2001-02:	72/71

… *nearly 100% increase in majors in last 4 years*

Now one of largest majors in Engineering School

CS Course Requirements: Comp Eng Major

Highly structured: includes most of CS + EE core major reqs.

- Programming Sequence:
  - Intro. To Programming (1007/1009)
  - Data Structures/Algorithms (3137/3139)
  - Advanced Programming (3157)
- Hardware/Architecture:
  - Digital Logic (3823)
  - Computer Organization (3824)
- Software Systems:
  - Programming Languages & Translators (4115)
  - Operating Systems (4118)
- Theory:
  - Discrete Math (3203)
  - Computability (3261)

CS Technical Electives: Comp Eng Major

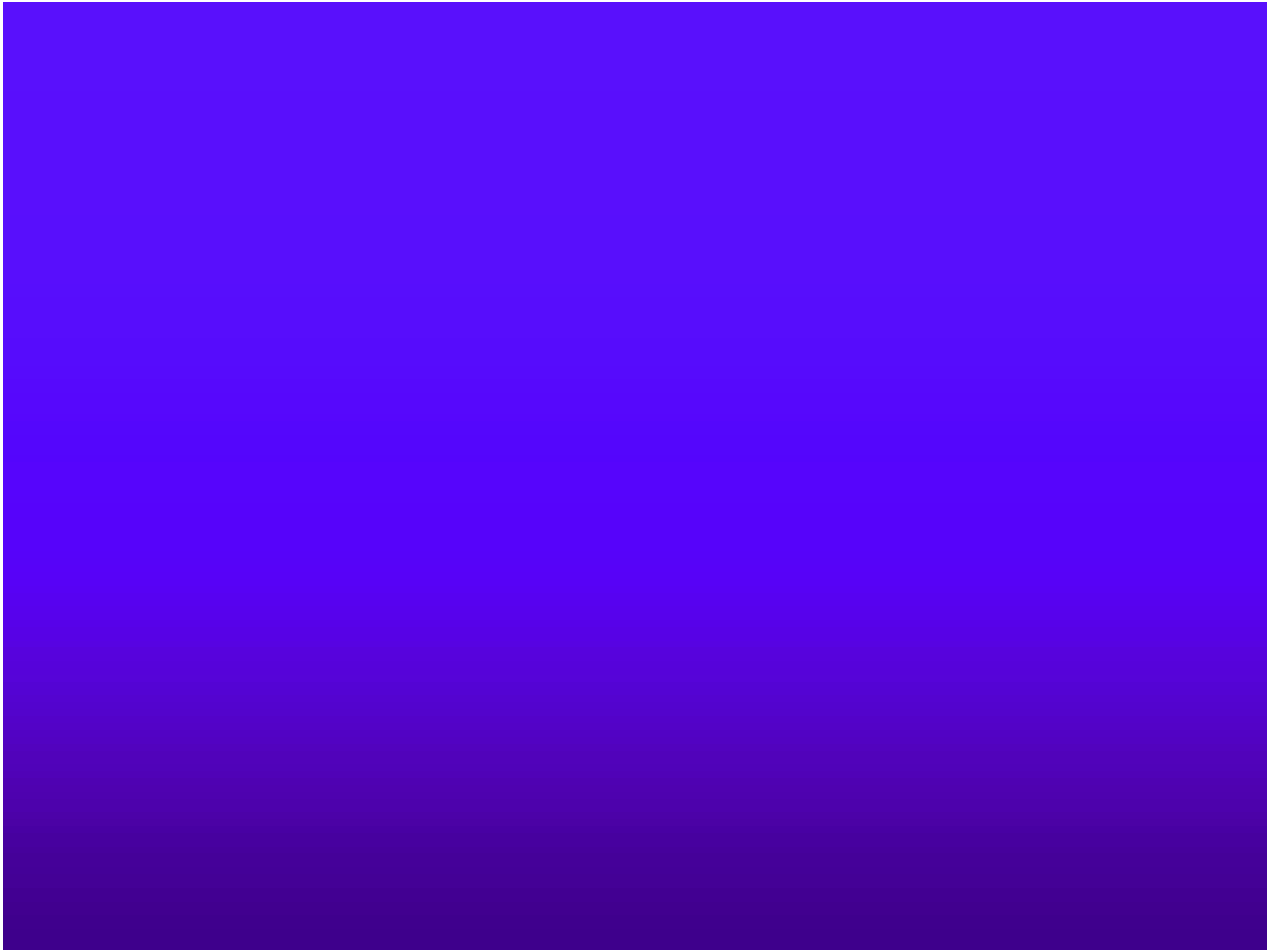
Specially relevant elective courses:

- Embedded systems
- Digital CAD
- Computer Architecture (advanced)
- Compilers & Translators

Future Needs: Computer Engineering

Currently, not enough Comp Eng faculty to cover all courses

- Some target hiring areas (CS side):
  - computer architecture (no architects currently in CS/EE)
  - digital system design
  - embedded systems
  - CAD: high-level synthesis, logic synthesis
  - reconfigurable computing
  - formal hardware verification
- Desired growth in "digital systems" area: next 3-4 years
  - 4-6 more faculty (CS + EE)



EE Side of Comp. Eng.

Charles Zukowski, EE Chair

1. Administration of Comp. Eng.
2. Academic Programs – EE Courses
3. Research Programs – NY MDC
4. Research in Digital Systems – EE Projects

1. EE Faculty on Computer Engineering Committee

a) Charles Zukowski (Committee)

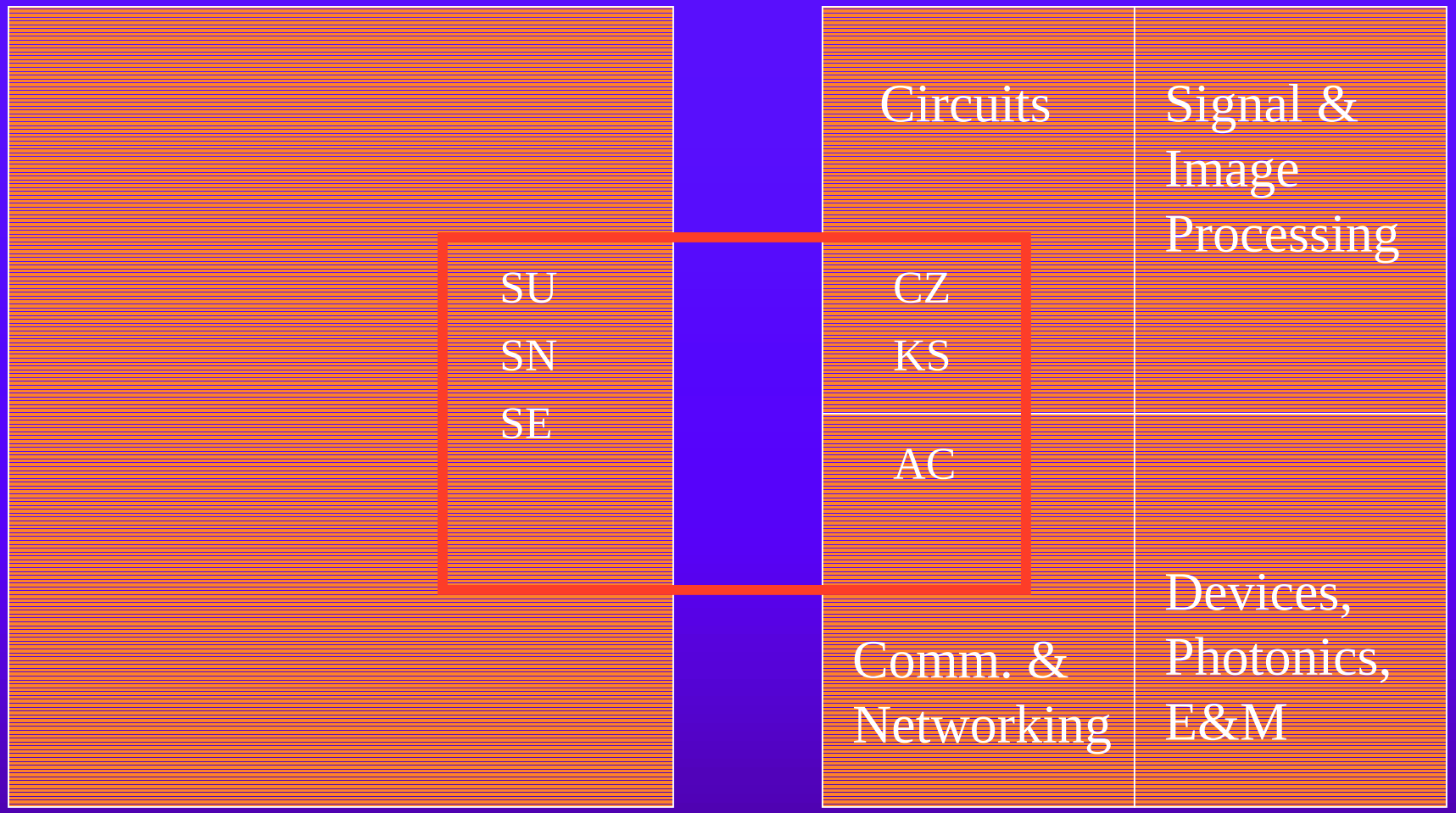
b) Ken Shepard (Committee, Acting Chair)

c) Andrew Campbell (Committee)

d) Potential Future Committee Member

- Position within EE Focus Areas
  - Communications/Networking (Lazar, Maxemchuk, Coffman, Jelenkovic, Campbell, Rubenstein)
  - Signal/Image Processing (Anastassiou, Chang, Eleftheriadis, Ellis, Wang)
  - Circuits (Tsividis, Kinget, Toth, Zukowski, Shepard)
  - Devices, Photonics, E&M (Osgood, Wang, Heinz, Bergman, Sen, Diament)

Computer Engineering Committee



2. EE Courses in Comp. Eng.

- Intro. to EE + Lab
- Circuits + Lab
- Electronics + Lab
- Signals & Systems + Lab
- Lab for Digital Systems
- Microprocessor / Computer Hardware Labs
- Technical Electives

3. Research Programs – New York Microelectronic Design Center (MDC)

- History
  - Virtual Center across NY Universities
  - NYSTAR ECAT Program
  - Critical Mass, Collaboration, Industry Links
- Structure
  - Director: Eby Friedman, U. Rochester
  - Assoc. Director: Charles Zukowski, Columbia
  - Rochester CEIS Administrative Staff
  - Columbia Comp. Eng. = 1/6 of 30 PIs at 12 Univ.

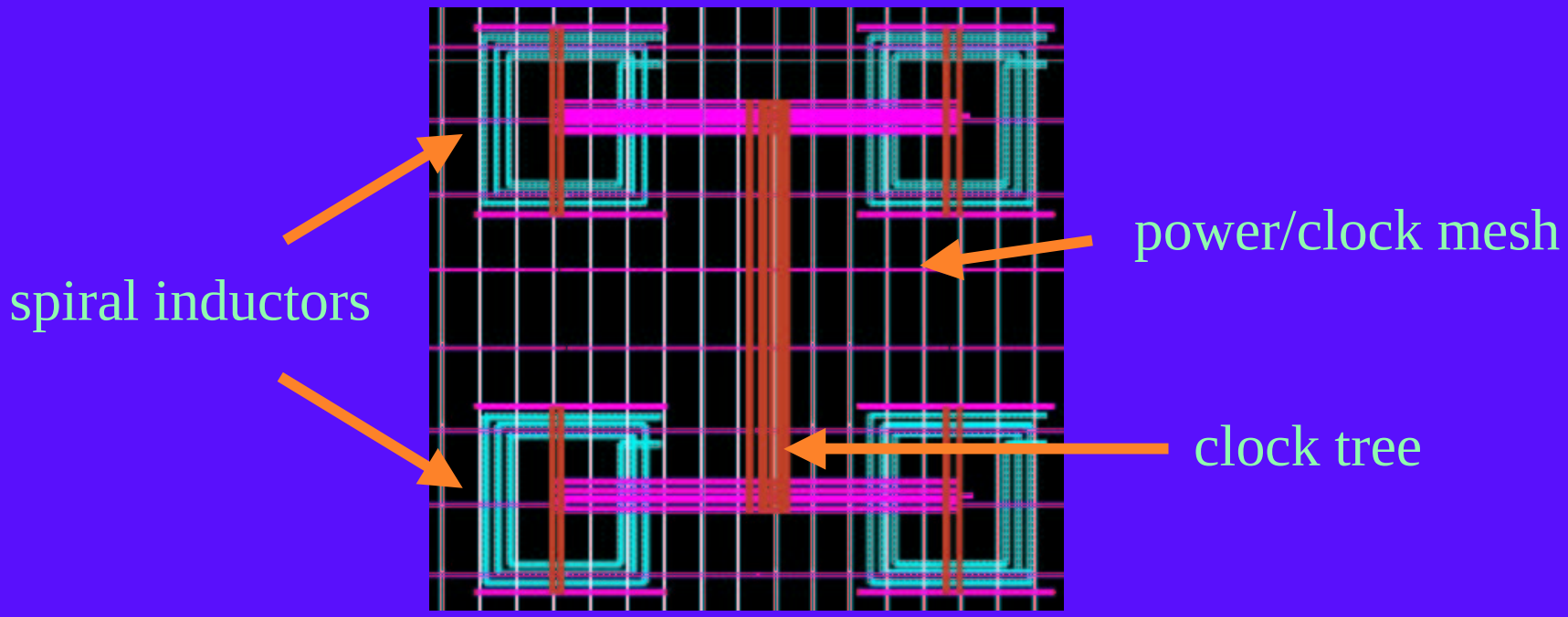
MDC (cont.)

- Funding
  - $2.6M / 2 yrs.
  - Individual & Collaborative Projects
  - Matching for Company Support
  - Technology Transfer is Focus
  - Semiconductor Research Corp. (SRC) Program
- Results
  - Ramping up well, NY Gov't Pleased
  - Annual Conference, Up to 120 (50 Industry)
  - Good for Columbia Comp. Eng. (+ Circuits)

4. EE Research Projects in Digital Systems

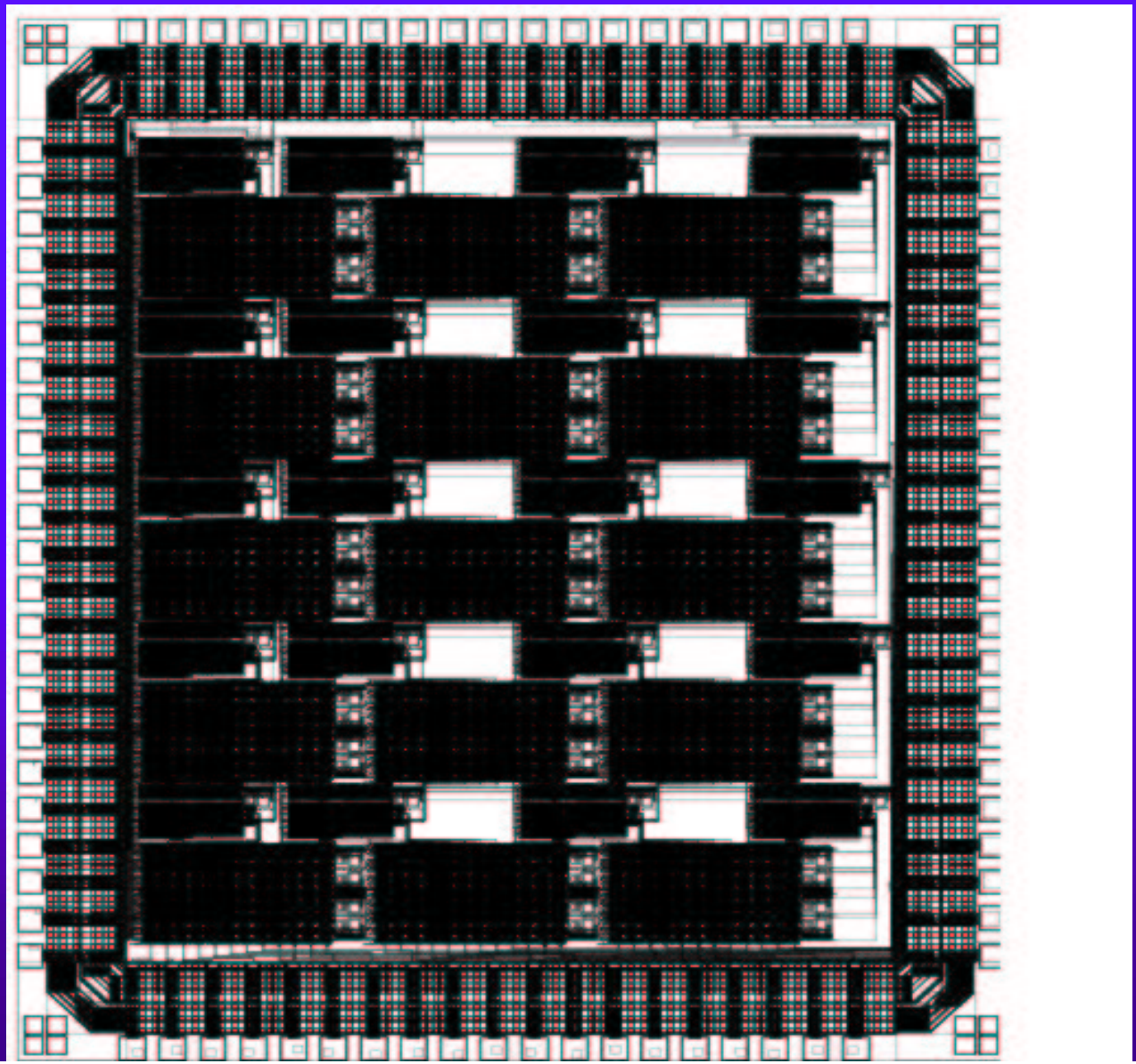
- Computer-Aided Design (CAD)
  - Stochastic Circuit Analysis (Shepard)
  - Interconnect Extraction & On-chip Sampling (Shepard)
- Circuit Design Techniques
  - Multi-GHz Resonant Clocking (Shepard)
  - Charge Recycling Voltage Domains (Shepard)
  - On-chip Serial Links (Shepard)
  - Low-power Logic Circuits (Zukowski)
- Applications
  - Asynchronous DSPs (Nowick, Shepard)
  - Active CMOS Biochips (Shepard)
  - Packet Processing with Dynamic VDD (Zukowski)
  - Hardware Simulation of Gene Networks (Zukowski)

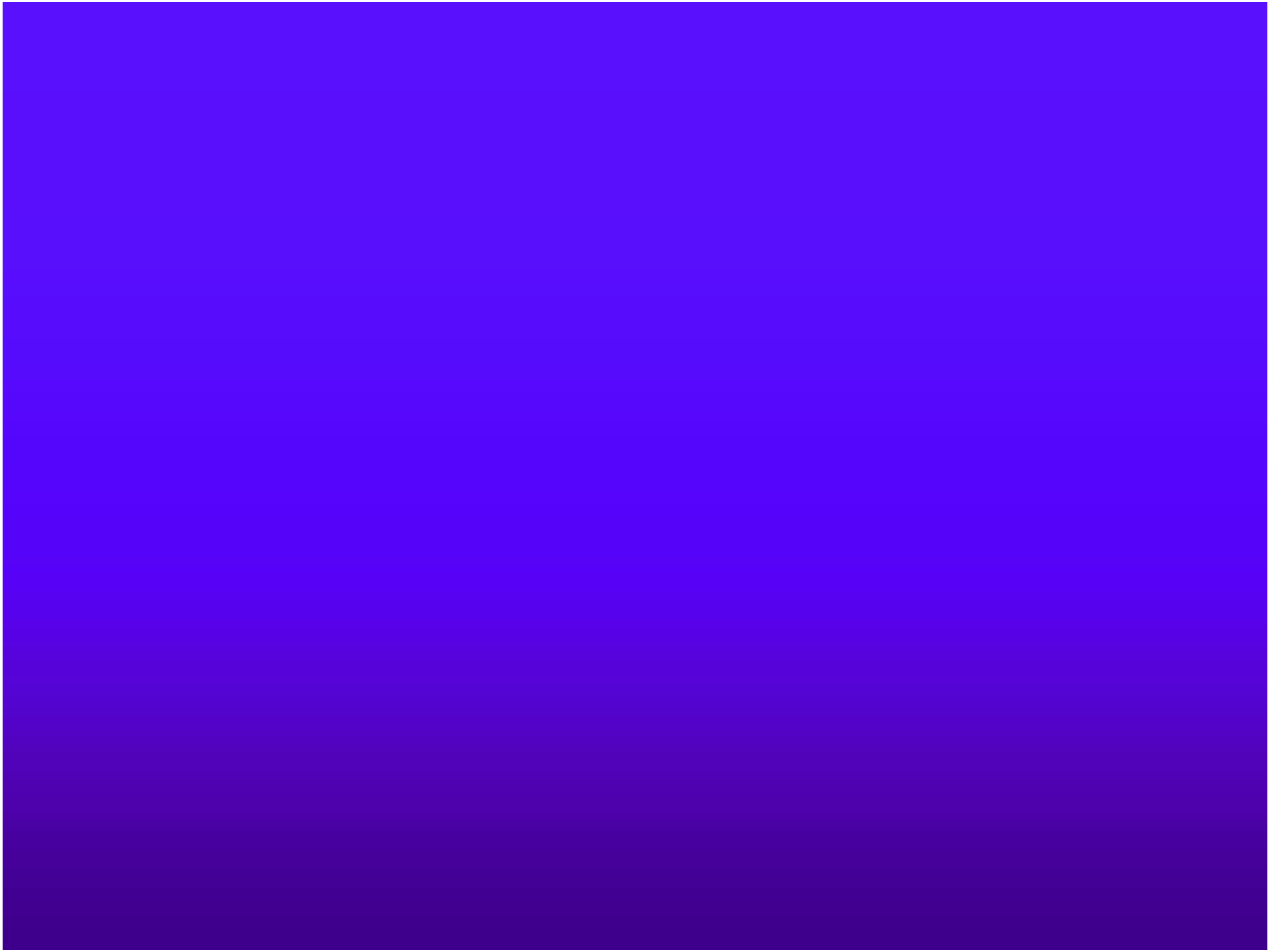
Resonant clock load



Traditional tree-driven grid augmented with spiral inductors to resonate the clock capacitance at the fundamental of the clock frequency.

Gene Network Simulator Chip





Research Areas

Stephen A. Edwards

Department of Computer Science,
Columbia University

www.cs.columbia.edu/~sedwards

sedwards@cs.columbia.edu

Embedded Systems

Computers masquerading as something else.

Casio Camera Watch

Nokia 7110 Browser Phone

Sony Playstation 2

Philips DVD Player

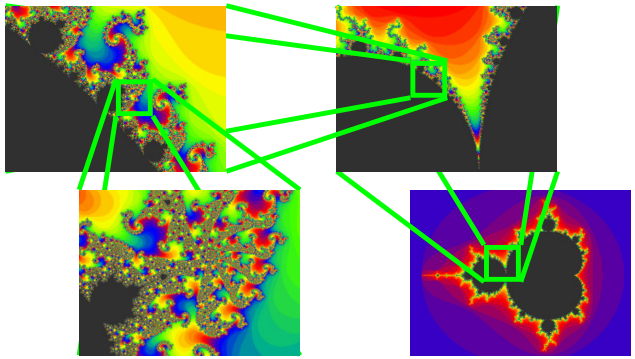
Philips TiVo Recorder

Long-Term Goal

Supplying tools that speed the development of embedded systems.

Embedded Systems Challenges

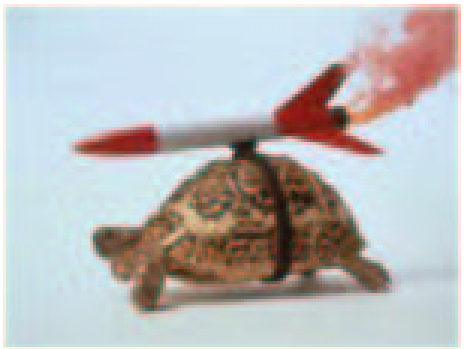
Real-time



Complexity

Photo by Thomas Donoghue

Concurrency



Legacy Languages

Software complexity growing

Size of Typical Embedded System

1985	13 kLOC	
1989	21 kLOC	↓ 44 % per year
1998	1 MLOC	
2000	2 MLOC	
2008	16 MLOC	≈ Windows NT 4.0
2010	32 MLOC	≈ Windows 2000

Source: “ESP: A 10-Year Retrospective,” Embedded Systems Programming, November 1998

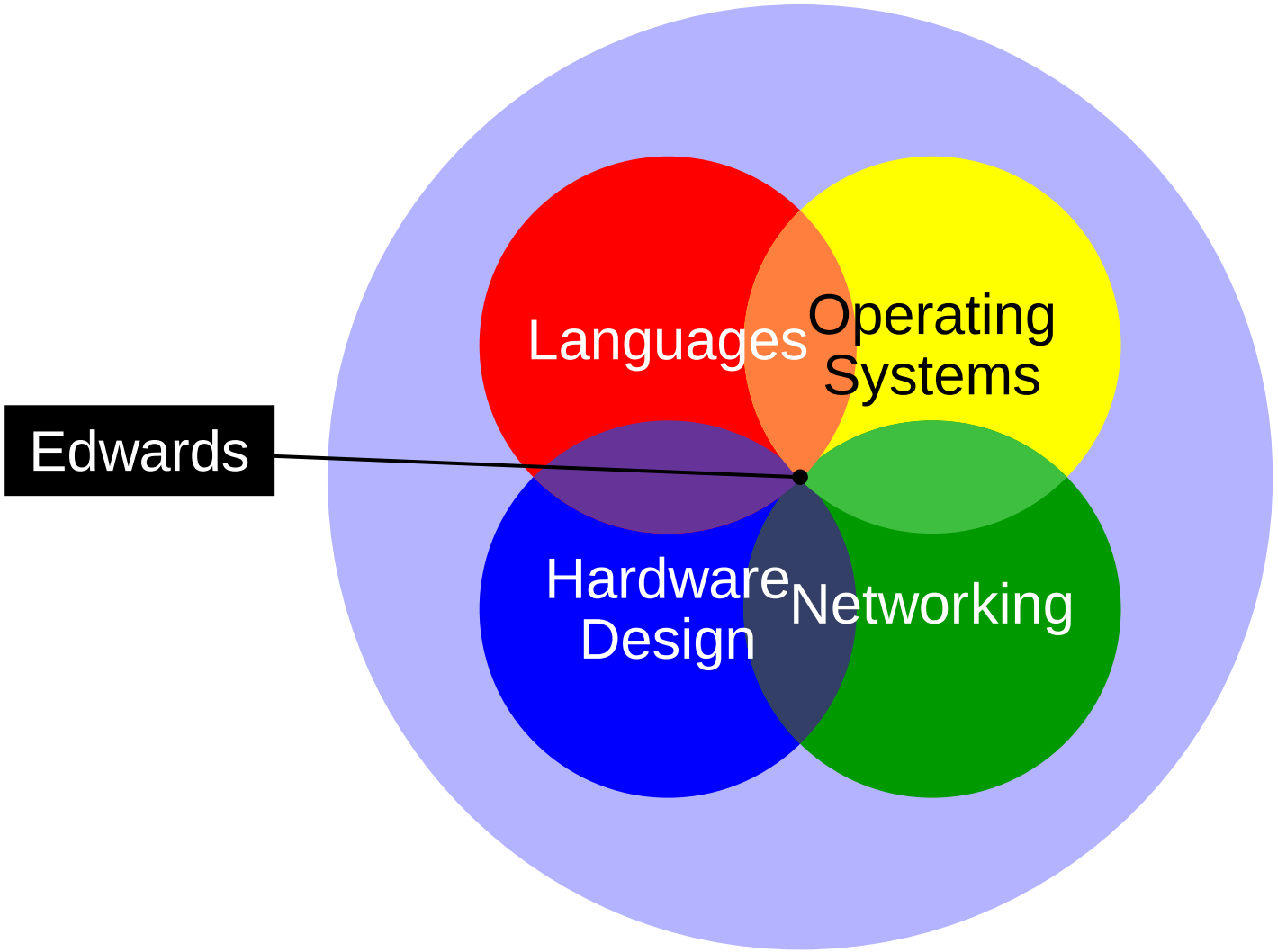
Written in stone-age languages

“Which of the following programming languages have you used for embedded systems in the last 12 months?”

C	81%
Assembly	70%
C++	39%
Visual Basic	16%
Java	7%

Source: “ESP: A 10-Year Retrospective,” Embedded Systems Programming, November 1998

Where I fit



Worlds:

DAC

ICCAD

IWLS

EMSOFT

Synchron

Domain-Specific Languages

Little languages that fit the problem

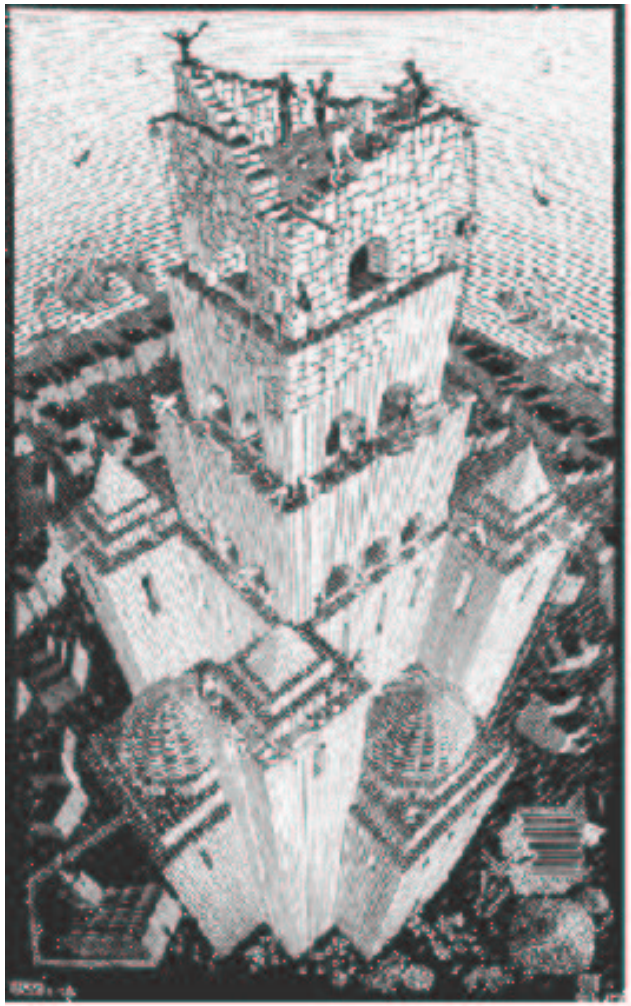
More succinct description that are

1. Quicker to create
2. Easier to get right

More opportunities for optimization and analysis

General-purpose languages hindered by undecidability

Domain-specific languages much simpler



Real-Time Languages

Esterel

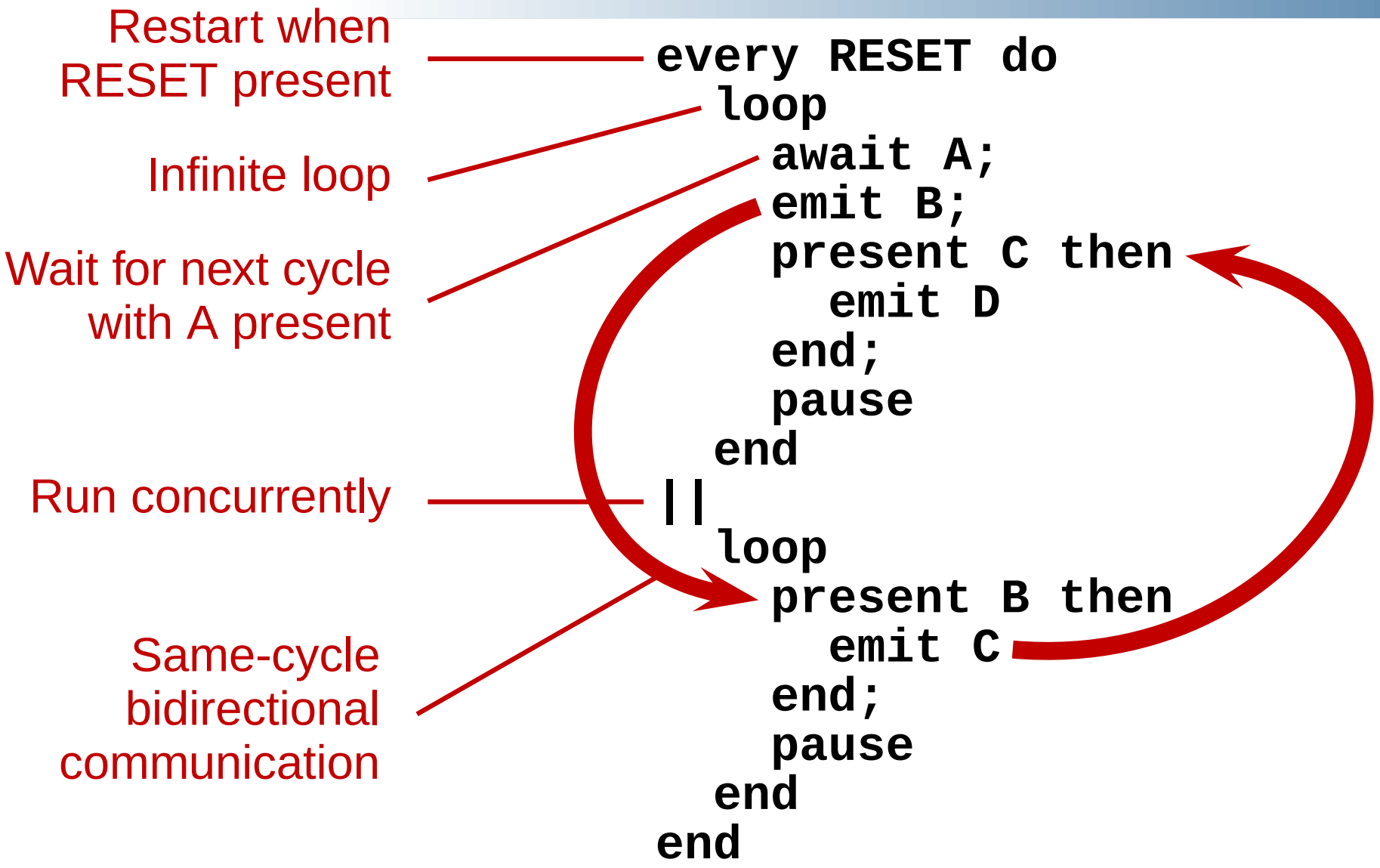
The Esterel Real-Time Langauge

Synchronous language developed by Gérard Berry in France

Basic idea: use global clock for synchronization in software like that in synchronous digital hardware.

Challenge: How to combine concurrency, synchronization, and instantaneous communication

Esterel


Restart when RESET present
Infinite loop
Wait for next cycle with A present
Run concurrently
Same-cycle bidirectional communication
every RESET do
loop
await A;
emit B;
present C then
emit D
end;
pause
end
||
loop
present B then
emit C
end;
pause
end
end

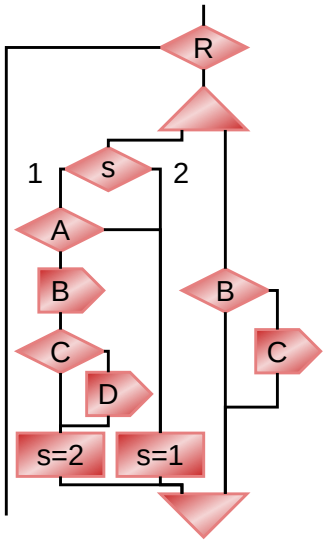
Previous Esterel Work

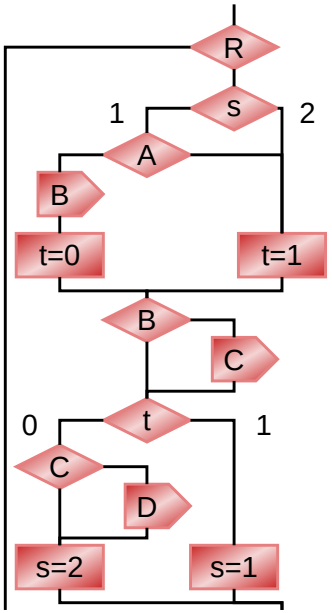
- Thesis on Esterel semantics in a heterogeneous environment (Ptolemy/Edward Lee/Berkeley)
- To appear in Science of Computer Programming

- Compiler that speeds up certain large programs 100×
- Used inside Synopsys' CoCentric System Studio
- Has limitations (e.g., owned by former employer)
- Published in IEEE Transactions on Computer-Aided Design 21(2), 2002.

Previous Esterel Compiler

```
every R do
   loop
      await A;
      emit B;
      present C then
         emit D end;
      pause
   end
||
   loop
      present B then
         emit C end;
      pause
   end
end
```






```
if ((s0 & 3) == 1) {
  if (S) {
    s3 = 1; s2 = 1; s1 = 1;
  } else
    if (s1 >> 1)
      s1 = 3;
    else {
      if ((s3 & 3) == 1) {
         s3 = 2; t3 = L1;
      } else {
         t3 = L2;
      }
```

Esterel Source

Concurrent CFG

Sequential CFG

C code

Ongoing Esterel Work

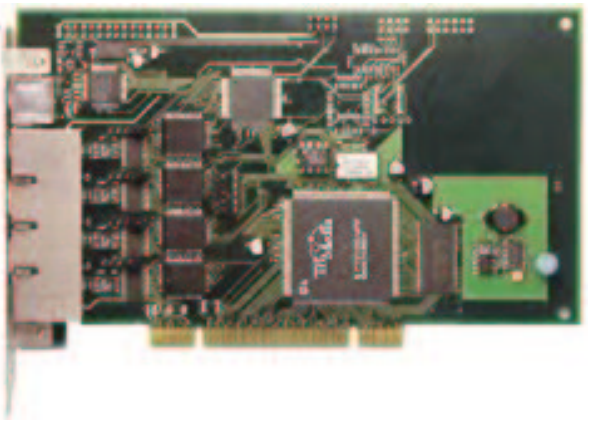
- New compiler infrastructure designed for research
- Better circuits from Esterel programs (Jia Zeng)
- Faster code from PDGs (Cristian Soviani)
- Event-driven code (Vimal Kapadia, Michael Halas)
- An interpreter for small-footprint applications (Aruchunan Vaseekaran)

The Hardware/Software Boundary

Device Drivers

Languages for Device Drivers

Device drivers are those pieces of software that you absolutely need that never seem to work



Big security/reliability hole: run in Kernel mode

Responsible for 80% of all Windows crashes

Tedious, difficult-to-write

Ever more important as customized hardware proliferates

Work by Others

Thibault, Marlet, and Consel

IEEE Transactions Software Engineering, 1999

Developed the Graphics Adaptor Language for writing XFree86 video card drivers

Report GAL drivers are 1/9th the size of their C counterparts

No performance penalty

Ongoing Work

Develop language for network card drivers under Linux (Chris Conway)

Sharing drivers between Linux and FreeBSD (Noel Vega)

Ultimate vision: compiler takes two programs: device spec. and OS spec. and synthesizes appropriate driver.

OS vendor makes sure OS spec. is correct; Hardware designer makes sure hardware spec. is correct.

NE2000 Ethernet driver (fragment)

```
ioports ne2000 {
  bits cr {
    bit stop, sta, transmit;
    enum:3 { 001=remRead, 010=remWrite,
             011=sendPacket, 1**=DMAdone }
    enum:2 { 00=page0, 01=page1, 10=page2 }
  }
  paged p {
    page0 { cr.page0; } {
      twobyte clda;
      byte bnry;
      bits tsr {
        bit ptx, 1, col, abt, crs, 0, cdh, owc;
     }
    page1 { cr.page1; } {
      byte:6 par;
      byte curr;
      byte:8 mar;
    }
```

Program Correctness

Verification Library Language

Verification Library Language

Joint work with Al Aho

Language extensions to support verification libraries for Java

Traditional Libraries	Provide functionality
Verification Libraries	Provide improved confidence in program correctness

Vision is a new methodology: verification as part of the development process, part of the same toolbox as adding functionality.

Example Verification Libraries

- Lint-like function call chekers
- Library that assumes the program is an FSM and can be checked using standard FSM tools
- Library that statically checks if a Java program uses a particular set of methods (e.g., deprecated ones)
- Library that removes array-bounds-checking code that can be proven unecessary

Think of a language mechanism that can supply **`-Wall`**, lint, purify, Spin, SLAM, Prefix, etc. as libraries as easy to use as those for I/O, GUIs, etc.

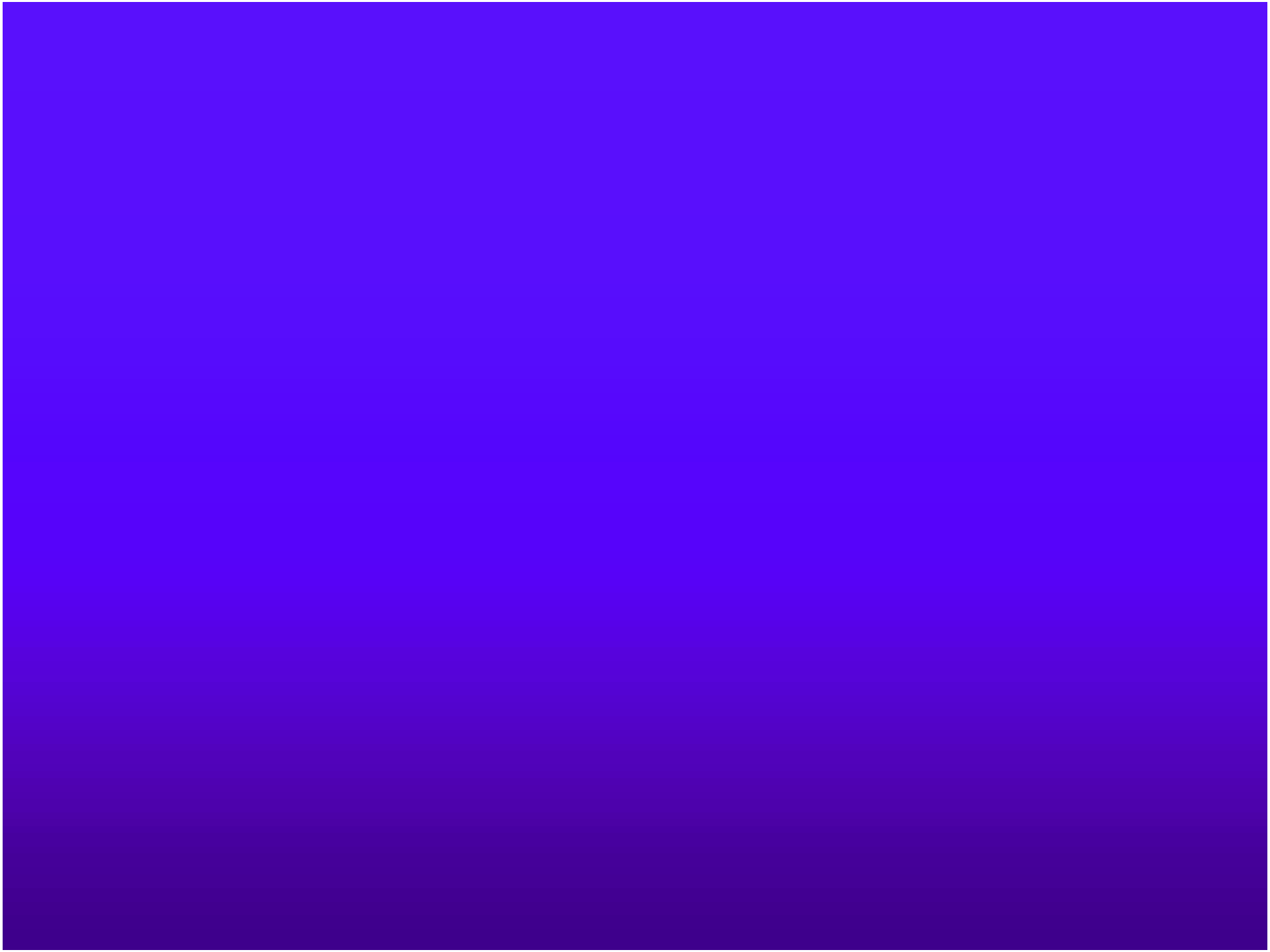
Funding and Collaborations

Interested in hardware synthesis technology; gift and hardware donation.

2002 CAREER award on Esterel and Device Driver language projects.

Multi-year grant to work on Esterel hardware synthesis problem.

Ongoing interaction/collaboration with company marketing Esterel-based development tools.



Research in Asynchronous and Mixed-Timing Digital Systems

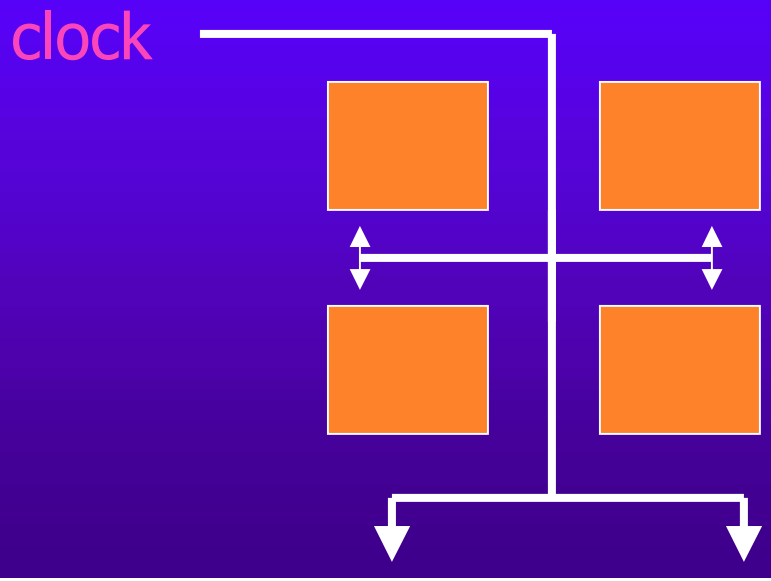
Prof. Steven M. Nowick

Prof. Stephen Unger *(on leave)*

Email: {nowick, unger} @cs.columbia.edu

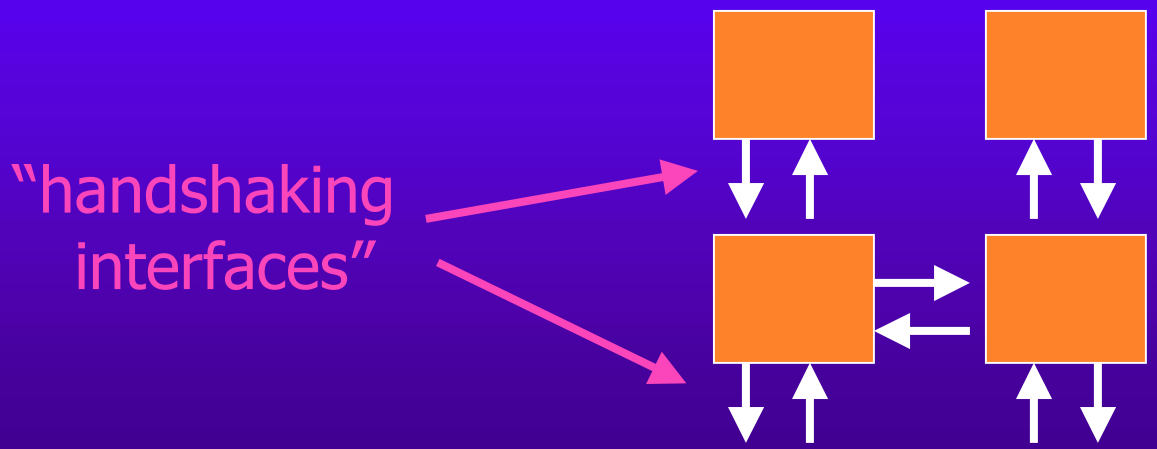
Introduction

- 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"*



Introduction (cont.)

Asynchronous Circuits:

- long history (since early 1950's), but...
- early approaches often impractical: slow, complex

Synchronous Circuits:

- used *almost everywhere!:* highly successful
- benefits: simplicity, support by existing design tools

But recently: renewed interest in asynchronous circuits

Trends and Challenges

Key Trends in Chip Design: next decade

- *"Semiconductor Industry Association (SIA) Roadmap"* (97-8)

Unprecedented Challenges:

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

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

Trends and Challenges (cont.)

1. Clock Rate:

- *1980: several MegaHertz*
- *2001: ~750 MegaHertz - 1+ GigaHertz*
- *2004: several GigaHertz*

Design Challenge:

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

Trends and Challenges (cont.)

2. Power Consumption

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

Design Challenge:

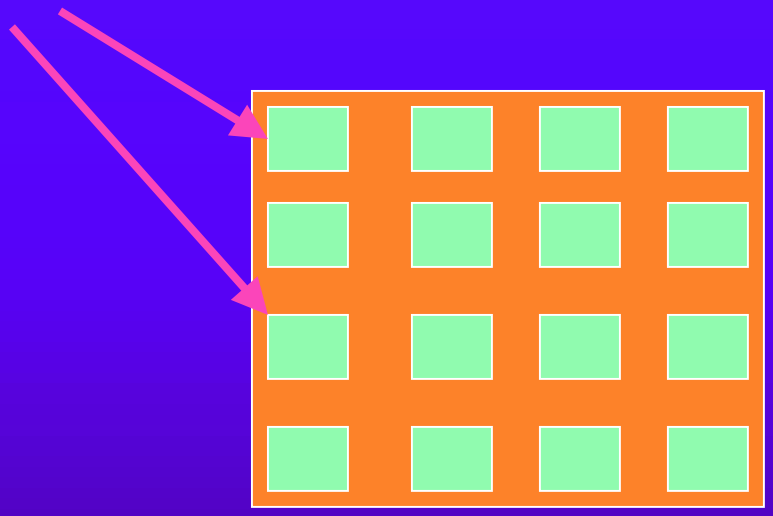
- *clock inherently consumes power continuously!*
- "power-down" techniques: only partly effective

Trends and Challenges (cont.)

3. Future Trends: "Mixed Timing" Domains

Chips themselves now becoming *distributed systems….*

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



Design Challenge: breakdown of single centralized clock control

Asynchronous Design

Several Potential Advantages:

- Lower Power
  - no clock *==>* components use power only "on demand"
- Robustness, Scalability
  - no global timing==>"mix-and-match" varied components
- Higher Performance
  - systems not limited to "worst-case" clock rate

Asynchronous Design: Challenges

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

Asynchronous Design: Recent Developments

1. Philips Semiconductors:
   - async chips in *commercial pagers, cell phones, smart cards*
   - *3-4x lower power,* less electromagnetic interference ("EMI")
2. Intel:
   - experimental async Pentium instruction-length decoder
   - *3-4x faster* than synchronous
3. Sun Labs, IBM Research:
   - experimental high-speed pipelines, routing fabric, systems

Recent Startups: Fulcrum, Theseus Logic

Government Agency: DARPA "Clockless Logic" Initiative (2003)

Research Highlights:

Prof. Unger *(on leave)*

Research Areas

- Designing Fast Async Components:
  - adders, comparators, etc. [Cheng/Unger, 1996-97]
  - develop novel "iterative circuits" in tree structures
- Novel Async Communication Protocols:
  - communicate between components using pulses
  - provide greater speed, simplicity, … [Plana/Unger, 1997]
- Hazard Elimination Techniques:
  - … by adding extra delays to a circuit
- Applications: Async ATM Switches [Cheng/Unger, 1996-pres.]

Research Highlights:

Prof. Nowick

Research Group

- Current Students:
  - Tiberiu Chelcea (PhD)
  - Cheoljoo Jeong (PhD)
  - Cheng-Hong Li (PhD)
  - Melinda Agyekum (MS)
  - Amitava Mitra (MS)
  - Charles O'Donnell (ugrad)
- Former PhD's:
  - Montek Singh (asst. prof., Univ. of North Carolina, CS Dept.)
  - Michael Theobald (visiting scientist, CMU CS Dept.)
  - Robert Fuhrer (IBM T.J. Watson)

My Research: Funding

NSF: 2 Medium-Scale *"ITR"* Awards ($2.5 Million) [2000]

1. "CAD Tools" to Design & Optimize Asynchronous Systems
   (joint with USC)
2. 3rd-Generation Wireless Systems (async, very low power)
   (joint with Columbia EE - Prof. Ken Shepard)

Other Funding: NSF, Sun, NYS MDC, Sloan Foundation

Research Highlights

3 Main Asynchronous Areas:

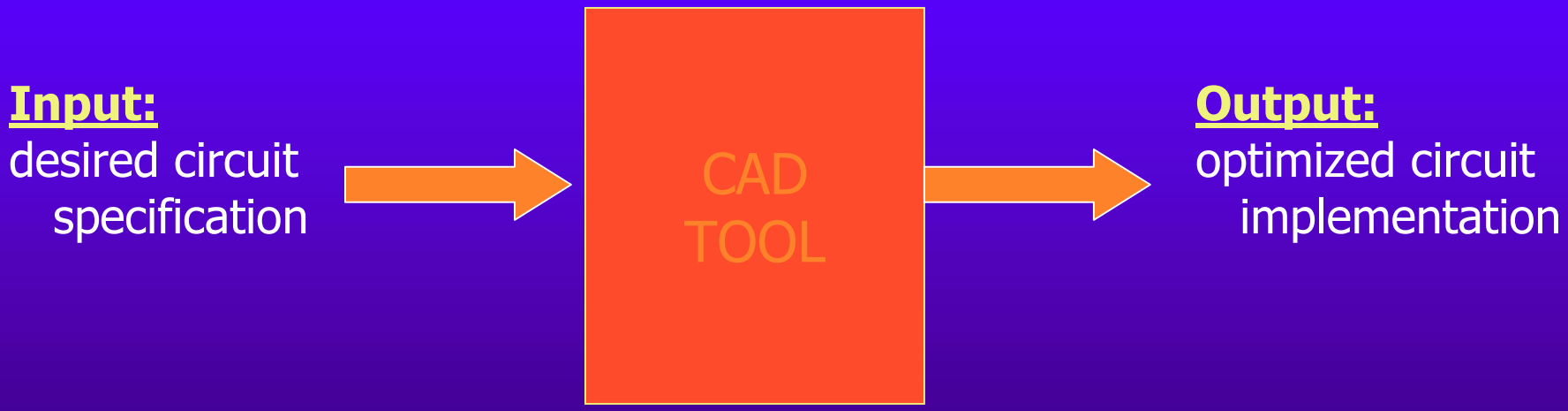
1. CAD Tools: optimization algorithms + software packages
   - (a) for large-scale systems
   - (b) for individual controllers
2. High-Speed Design: pipelined adders, multipliers, etc.
3. Interface Circuits: for mixed-timing systems

1. Developing Asynchronous CAD Tools

Software programs to aid designers =

"computer-aided design" tools

– automatically *synthesize* and *optimize* digital circuits



1. Developing Asynchronous CAD Tools

Focus: 2 types of CAD tools

(a) for *entire* digital systems ☐

(b) for *individual* controllers (*i.e.,* finite-state machines)

(a) High-Level Synthesis Package

– M. THEOBALD, T. CHELCEA

(b) The "MINIMALIST" Package

– R. FUHRER, M. THEOBALD

Include: sophisticated optimization algorithms

Goal: provide many flexible designer options

1(a). Synthesis and Optimization of Large-Scale Asynchronous Systems

Several existing CAD tools for *large async systems*

- Tangram: Philips Semiconductors
  -- used in research labs + product divisions
- Balsa: Univ. of Manchester (UK) -- public-domain

Start 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

Many commercial + academic chips/applications:

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

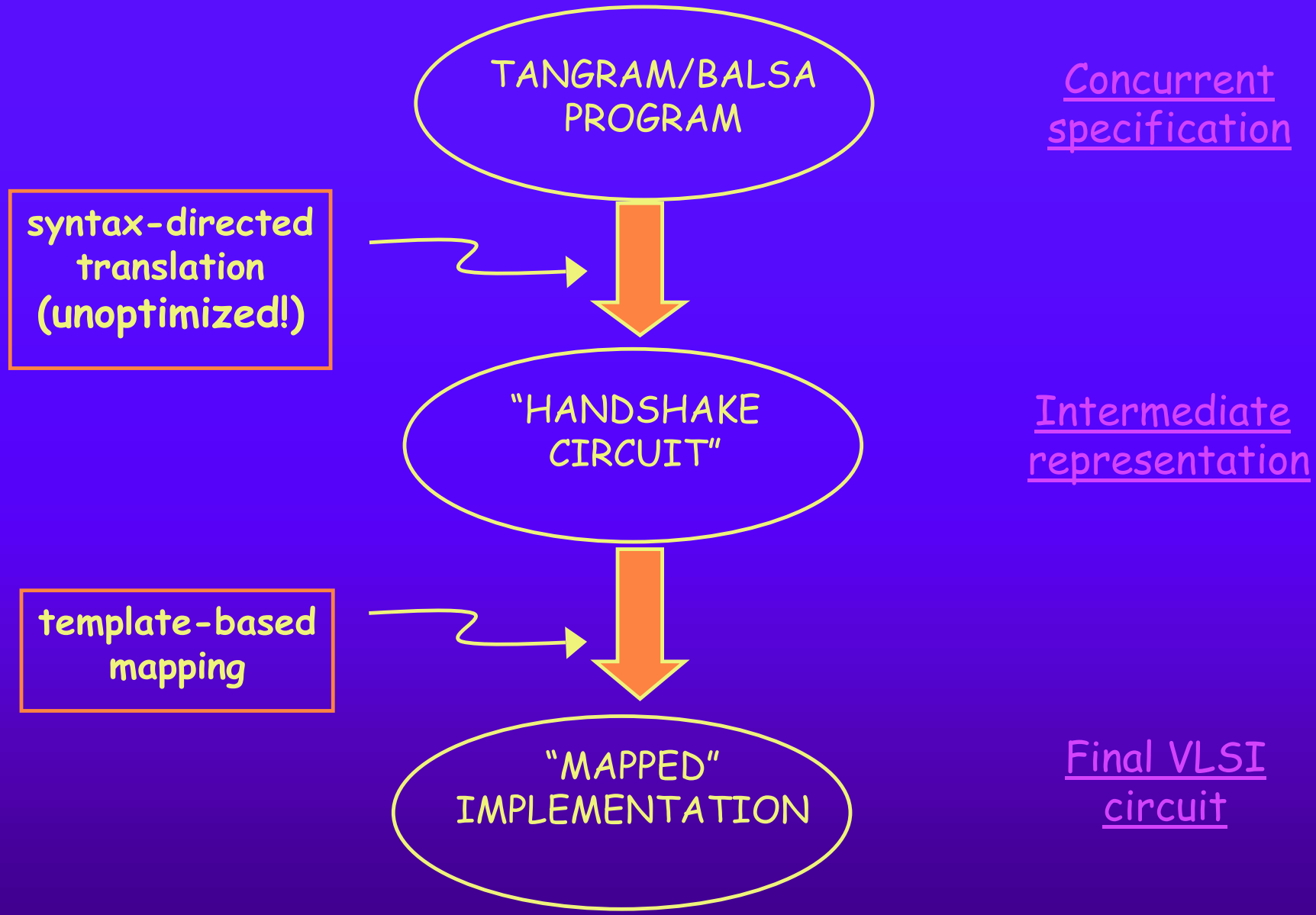
    ==> smart cards, pagers, cell phones
- Balsa: ARM processors, smart cards, …

Many sophisticated tool features:

- profilers, early estimation tools (power, delay), testing

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

Basic "Silicon Compiler" Flow



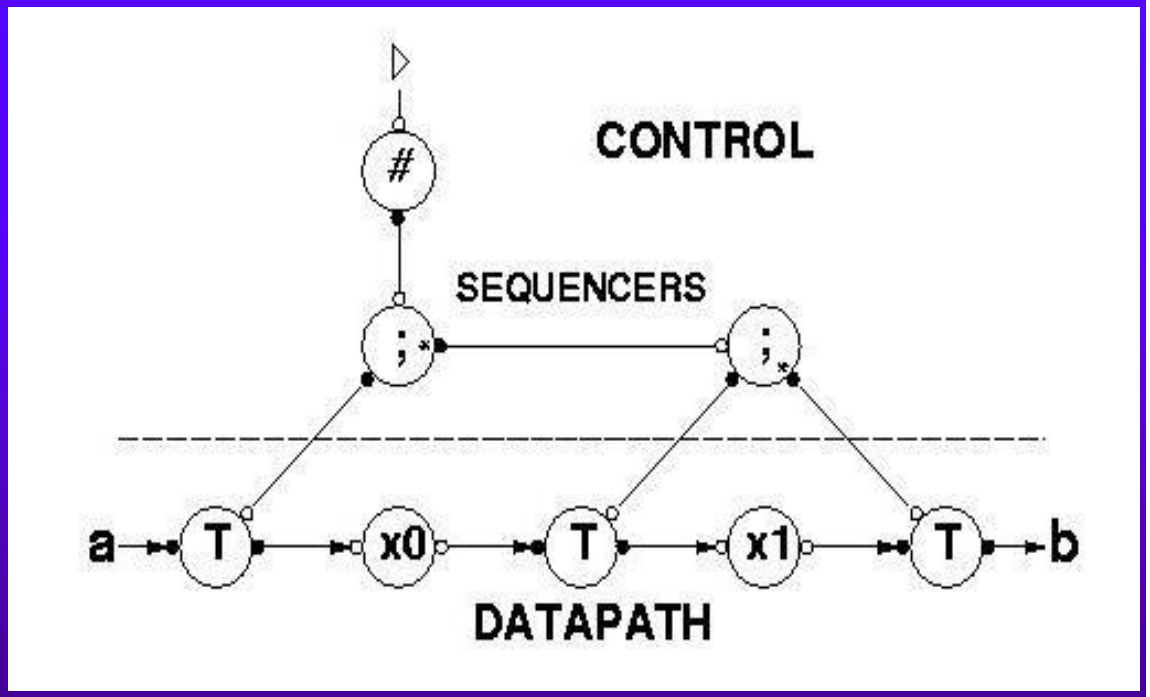
Synthesizing a System: a Small Example

2-Place "Ripple Register" (FIFO)

Tangram Program

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

Intermediate "Handshake Circuit"



syntax-directed translation (with no optzns.)

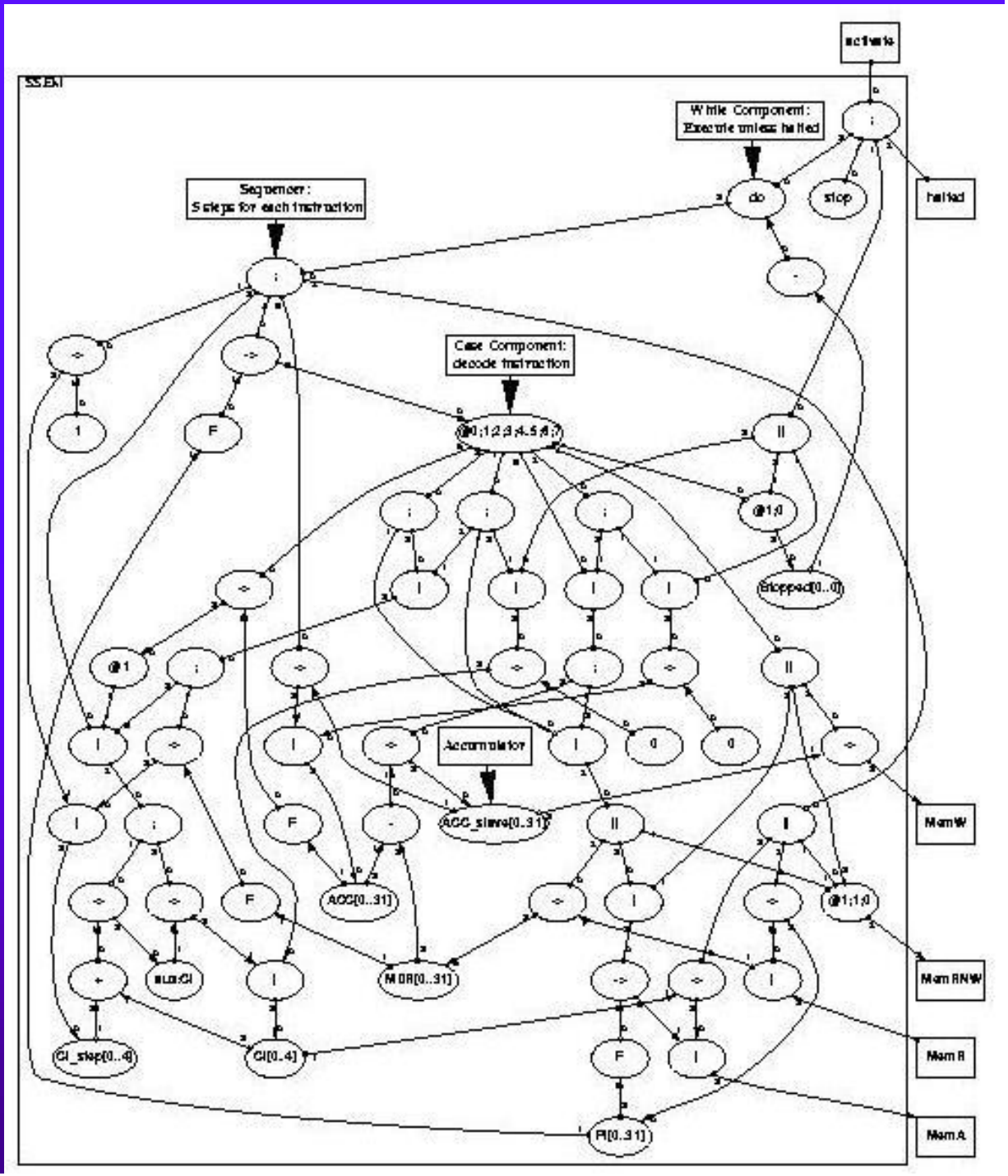
A Larger Example

Manchester "Baby" Processor (1948):

SSEM

Intermediate "Handshake Circuit"

Our Research Goal:
to develop a powerful optimizer for these "intermediate circuits"

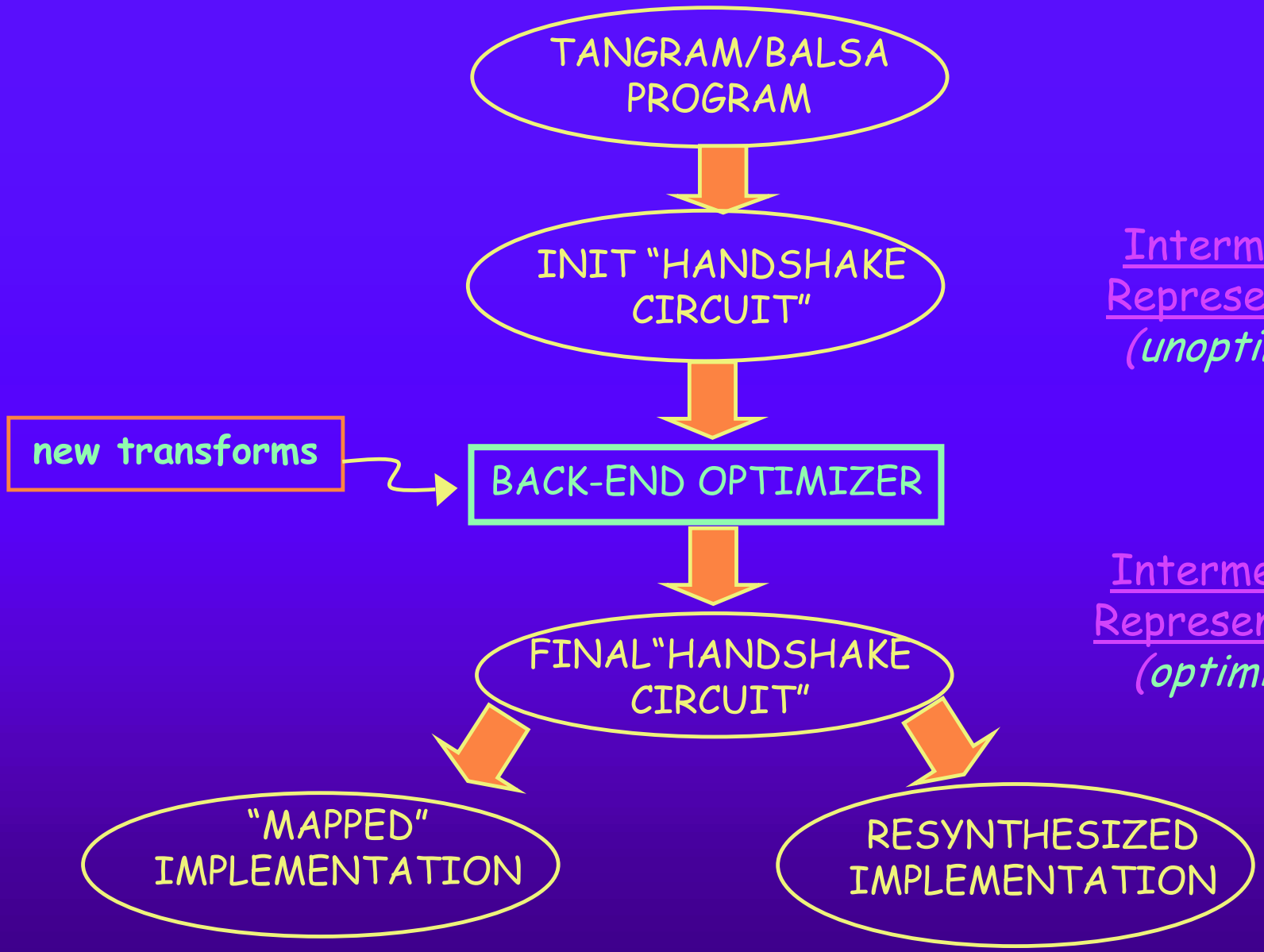


Contribution: a "Back-End" Optimizer

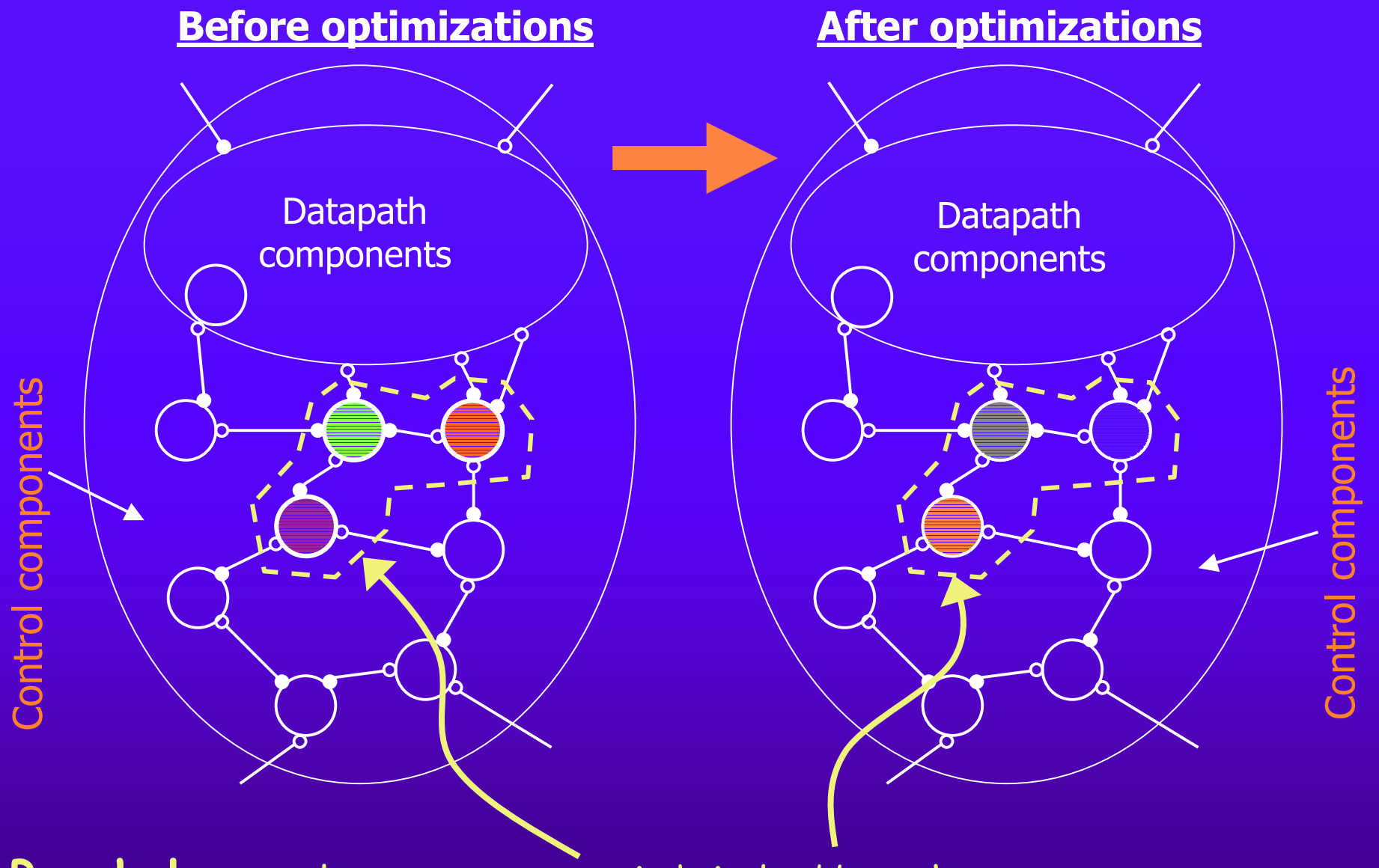
- New Optimizing Transformations:
  - Peephole: replace components in sliding 'window' by other components
  - Resynthesis: re-synthesize a cluster of components

  goal = *overcome limitations of strict syntax-directed compilation*
- CAD Package Integration:
  - integrating into Univ. of Manchester's "Balsa" tool
  - work in progress...:
    - incorporating commercial CAD tools (Synopsys, Cadence)
    - improving and formalizing transforms
    - developing 'design scripts'
- Collaborators: University of Manchester, UK (S. Furber et al.)
- Applications: async ARM processors for smart cards, etc.

Revised Synthesis Flow: with Back-End Optimizer

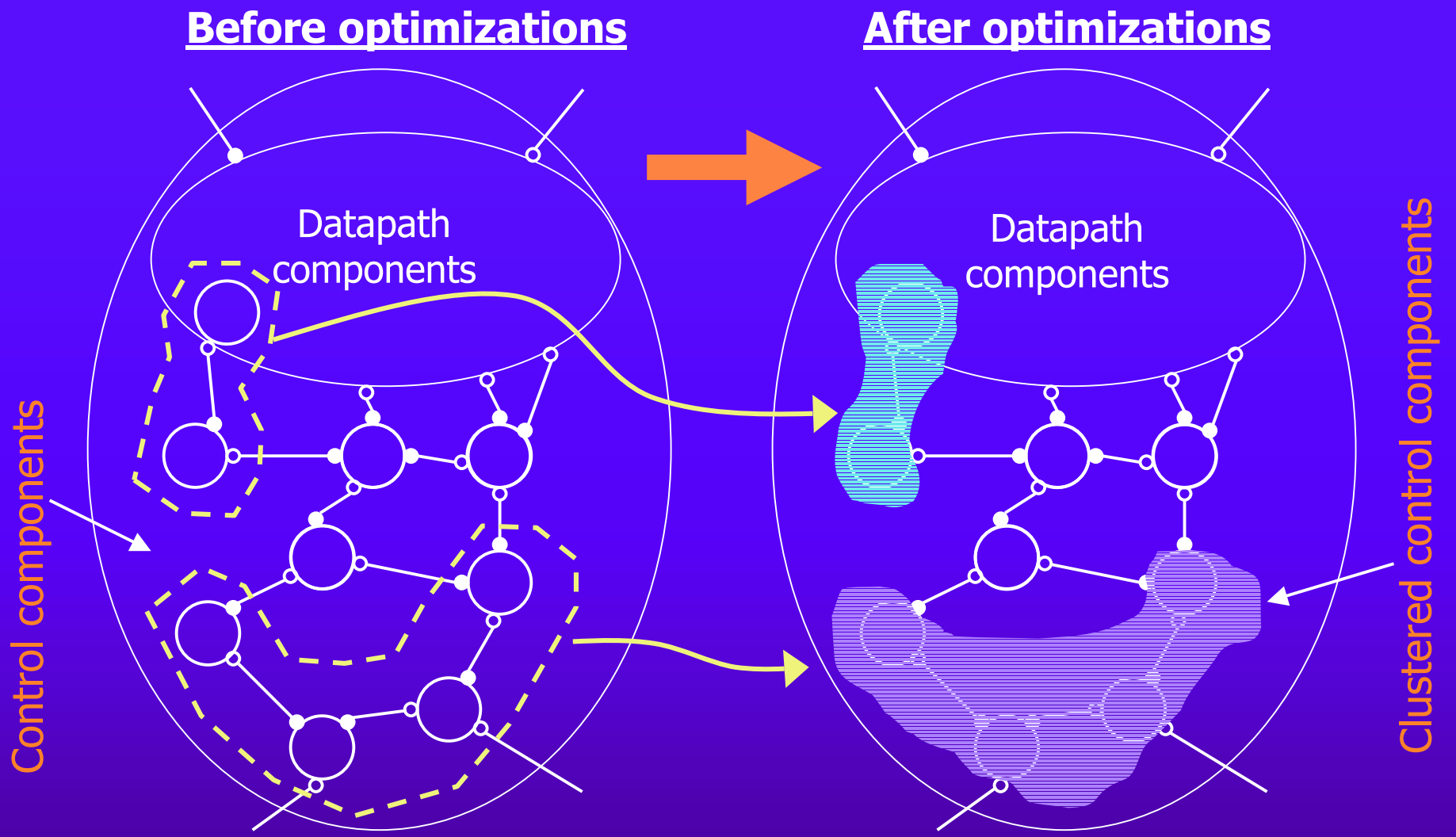


New Optimizations (1)



Peephole: replace components in 'window' by other components

New Optimizations (2)



Resynthesis: cluster together components (both control and datapath)

Summary: Synthesis & Optzn. of Large-Scale Async Systems

Developing powerful "back-end optimizer" for Balsa CAD package

- Contributions: [Chelcea/Nowick, DAC-01]
  - propose new set of resynthesis + peephole optimizations
  - integrate into public-domain Balsa CAD tool
- Overcome drawback of unoptimized syntax-directed translation
  - up to 54% performance improvement
- Work in progress …
  - formal verification
  - develop "design scripts" to apply transforms
  - introduce new transforms
  - large case-studies: entire microprocessors

Research Highlights

3 Main Asynchronous Areas:

1. CAD Tools: optimization algorithms + software packages
   - (a) for large-scale systems
   - (b) for individual controllers
2. High-Speed Design: pipelined adders, multipliers, etc.
3. Interface Circuits: for mixed-timing systems

1(b) The MINIMALIST CAD Package

MINIMALIST: developed at Columbia University [1994-]

- extensible CAD package for async controllers
- integrates synthesis + verification tools
- used in 80+ sites/17+ countries (... now being taught in IIT Bombay)
- URL: http://www.cs.columbia.edu/async

Includes several optimization tools:

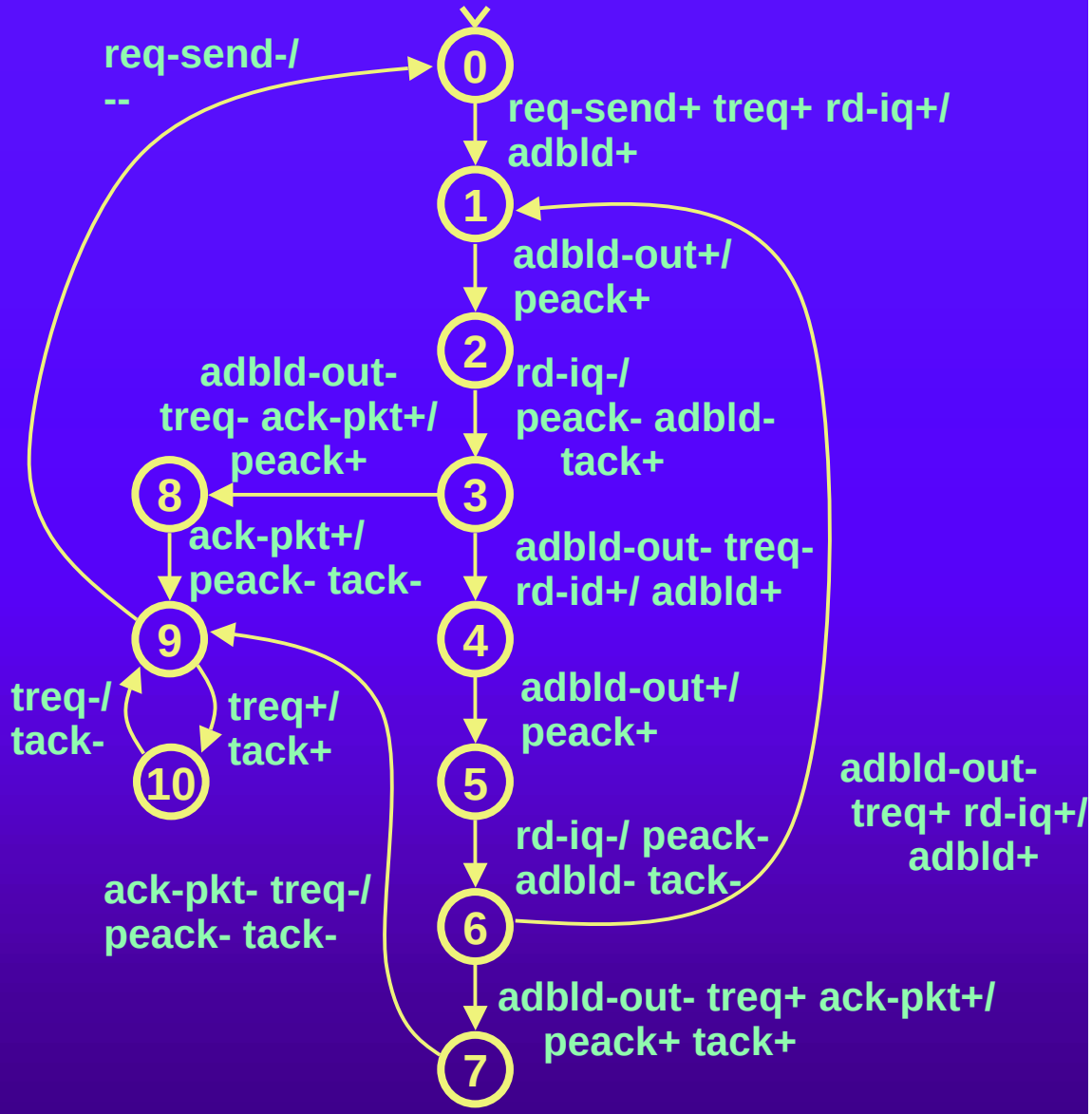
- State Minimization
- CHASM: optimal state encoding
- 2-Level Logic Minimization
- Verilog Back-End
- Verifier

Key goal: *facilitate design-space exploration*

1(b). Synthesizing A Controller Using the "MINIMALIST" CAD Tool

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

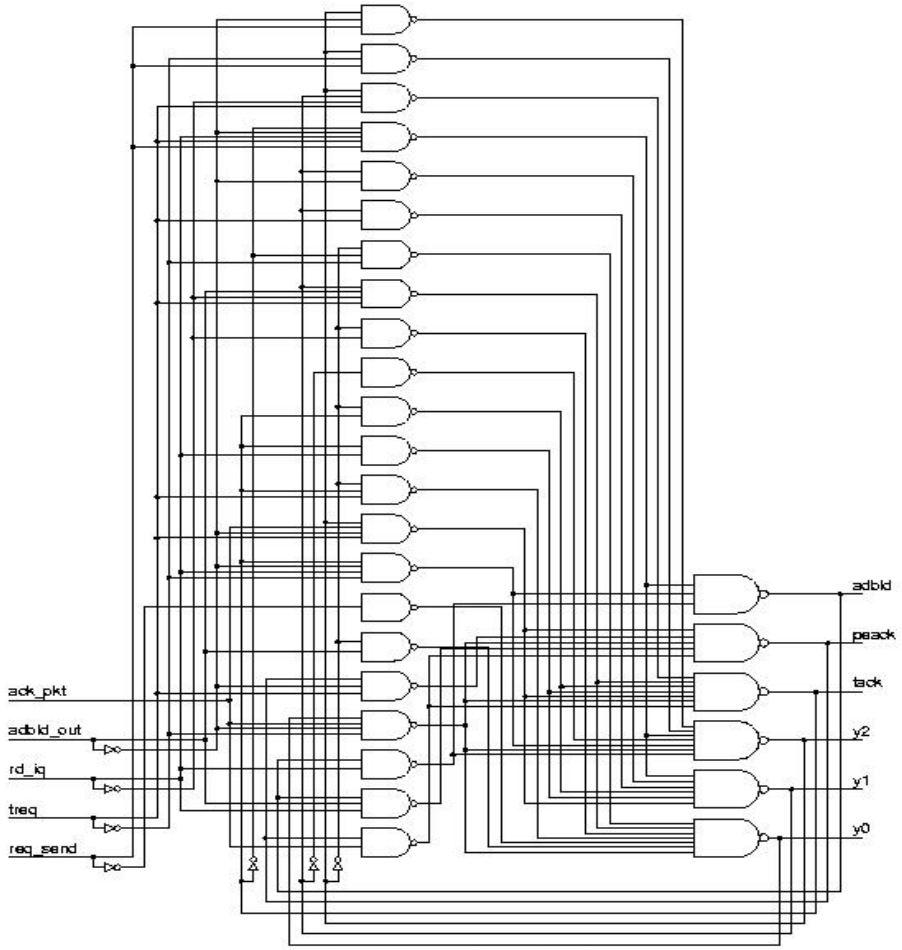
Outputs:
tack
peack
adbld



From HP Labs
"Mayfly" Project

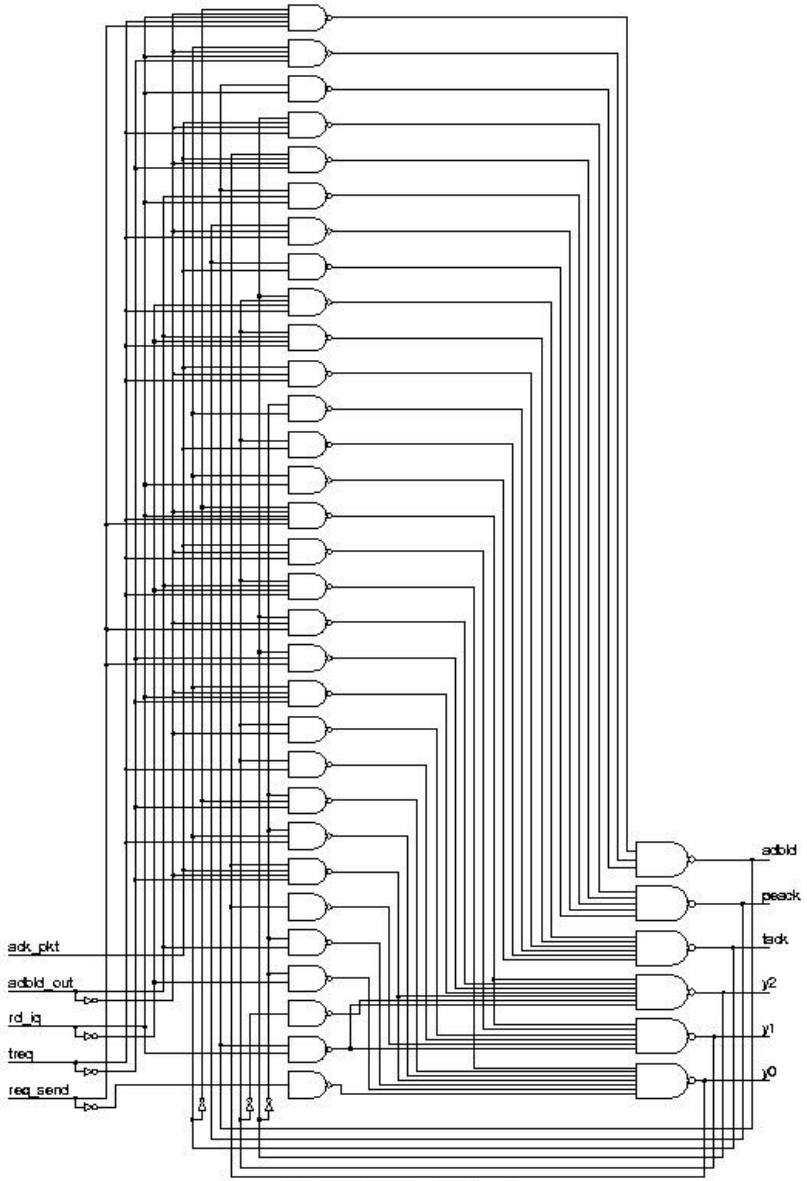
Design-Space Exploration using MINIMALIST:

different 'modes' provide user with flexibility



PE_SEND_IFC-FL

using an "area script"



PE_SEND_IFC-Fs

using a "speed script"

Research Highlights

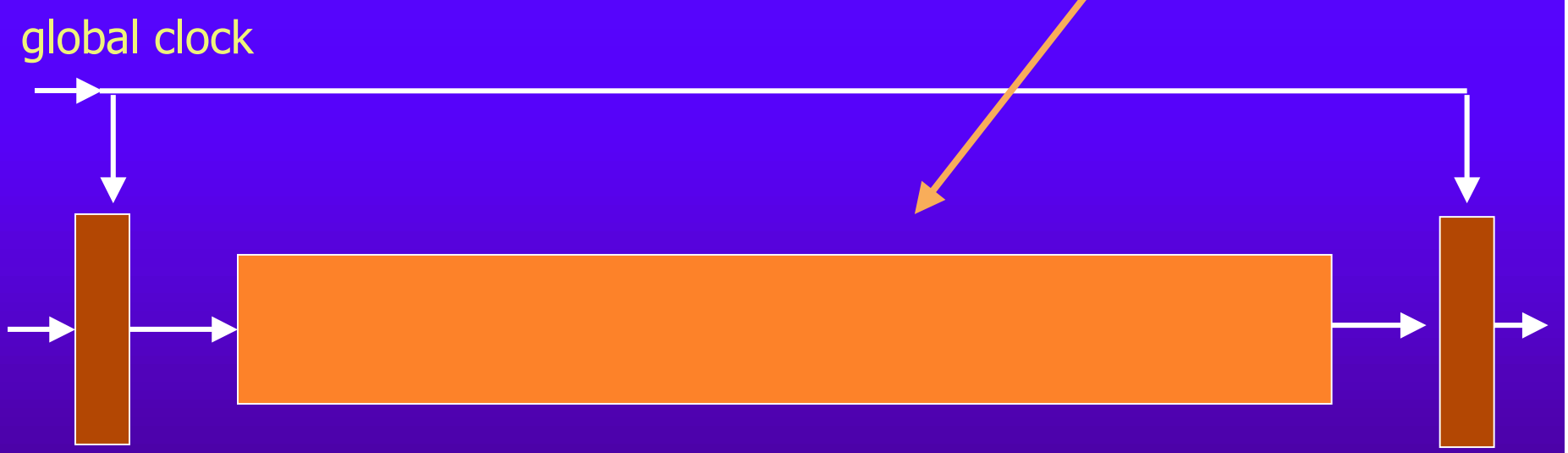
3 Main Asynchronous Areas:

1. CAD Tools: optimization algorithms + software packages
   - (a) for large-scale systems
   - (b) for individual controllers
2. High-Speed Design: pipelined adders, multipliers, etc.
3. Interface Circuits: for mixed-timing systems

2. Ultra High-Speed Digital Design

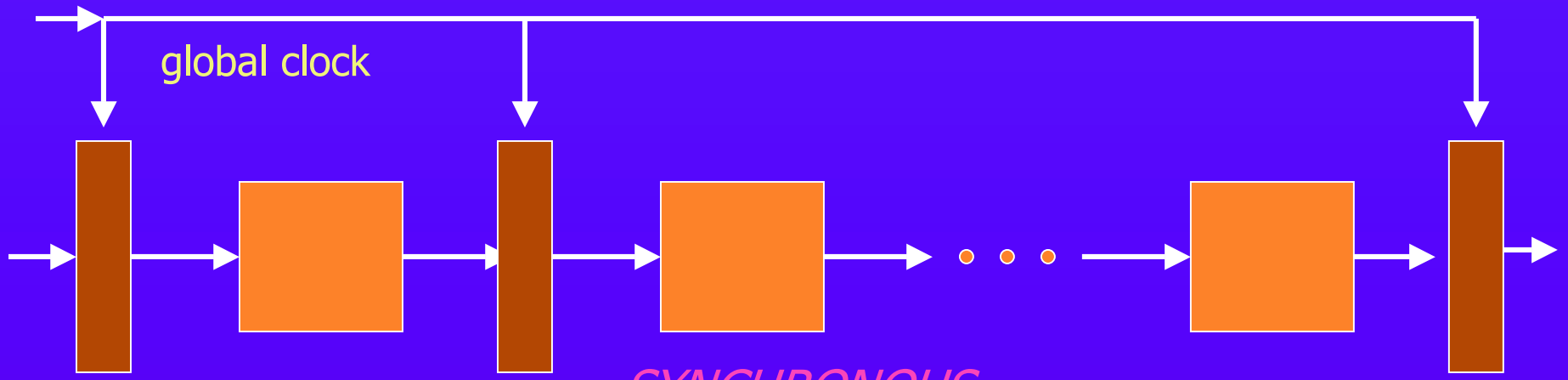
NON-PIPELINED COMPUTATION:

"datapath component" = adder, multiplier, etc.



2. Ultra High-Speed Digital Design

"PIPELINED COMPUTATION": like an assembly line



SYNCHRONOUS

no global clock

ASYNCHRONOUS

2. Ultra High-Speed Digital Design

Goal: extremely fast async datapath components

- speed: comparable to fastest existing synchronous designs
- additional benefits:
  - gracefully adapt to varying interface speeds
  - "elastic" processing of data in pipeline
  - no global clock distribution

Contribution: 3 new async pipeline styles [SINGH/NOWICK]

- use novel highly-concurrent protocols
- basic operating speed: 3.5+ GigaHertz

Technology Transfer to IBM Research

Recently invited by IBM to transfer async pipeline technology:

… to fabricate an experimental FIR filter chip (for disk drives)

- PhD Student (Montek Singh): 5-month internship (5-12/00)
- IBM Design Application: filter design
- Fabricated Chip: evaluated in Feb.-March 2001 [ISSCC-02]
- Mixed-Timing Features:
  - "sandwich" async between sync interfaces
- Benefits: power, speed, flexibility

Potential for future use in disk drives….

Research Highlights

3 Main Asynchronous Areas:

1. CAD Tools: optimization algorithms + software packages
   - (a) for large-scale systems
   - (b) for individual controllers
2. High-Speed Design: pipelined adders, multipliers, etc.
3. Interface Circuits: for mixed-timing systems

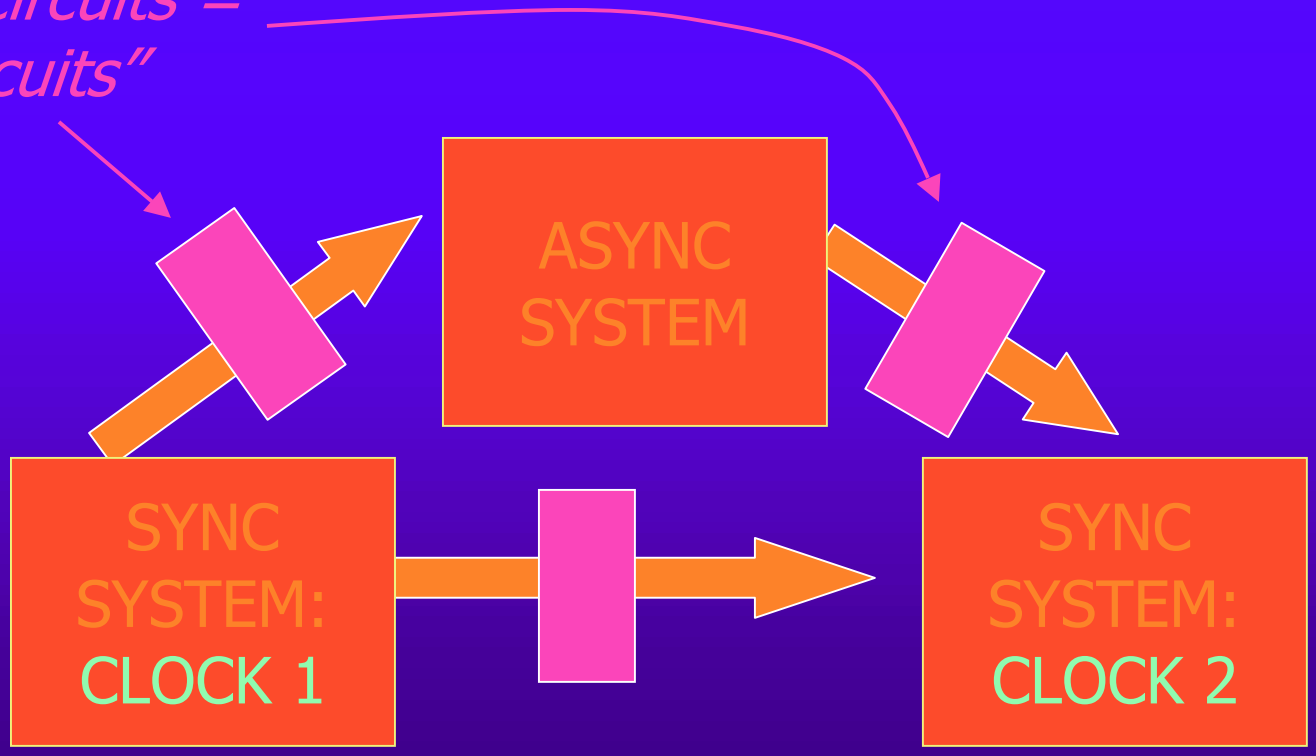
3. Robust Interface Circuits for "Mixed-Timing" Systems

Critical challenge: to *interface* sync/async systems,

sync/sync systems with different clock rates

--*robustly, at high-speed* [CHELCEA/NOWICK, DAC-01]

interface circuits = "glue circuits"



Summary: Collaborations/Tech Transfer

1. 3rd Generation Wireless Systems: Columbia EE [Prof. K. Shepard]
   … using asynchronous components

2. CAD Tools for Large-Scale Async Systems:
   - USC [Prof. P. Beerel]
   - Univ. of Manchester (UK) [Profs. D. Edwards, S. Furber]

3. High-Speed Pipelines:
   - IBM T.J. Watson [Drs. J. Tierno, P. Kudva]

4. Interface Circuits for Mixed-Timing Domains:
   - […a Silicon Valley company…]

Conclusions

Main Research Goal: making async circuits practical for designers

Projects:

1. CAD Tools:
   - for large-scale systems: optimizing back-end silicon compiler
     … can be used to synthesize entire ARM processors
     – integration into public-domain Balsa CAD Package (Manchester, UK)
   - for individual controllers: the "MINIMALIST" tool
2. High-Throughput Pipelines:
   – 3 new styles: for both static + dynamic CMOS circuits
3. Interface Circuits for Mixed-Timing Domains:
   – for sync/sync, mixed async-sync