Asynchronous and GALS Design: Overview and Recent Advances

Prof. Steven M. Nowick
nowick@cs.columbia.edu

Department of Computer Science
Columbia University
New York, NY, USA

Computer Systems Laboratory Presentation
October 24, 2016
(NY, NY)
Synchronous vs. Asynchronous Systems?

- **Synchronous Systems:** use a *global clock*
  - entire system operates *at fixed-rate*
  - uses “*centralized control*”
Synchronous vs. Asynchronous Systems? (cont.)

- Asynchronous Systems: *no global clock*
  - components can operate at *varying rates*
  - communicate locally via “handshaking”
  - uses “distributed control”

“handshaking interfaces” (channels)
Trends and Challenges

Trends in Chip Design: next decade

- International Technology Roadmap for Semiconductors (ITRS)

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....
1. **Clock Rate:**

- **1980:** several MegaHertz
- **2016:** 1-6 GigaHertz (and falling)

**Design Challenge:**

- **clock skew:** clock must be near-simultaneous across entire chip
  - Various optimization techniques: optimal clocking, skew-tolerant, resonant clocking, etc.
2. Chip Size and Density:

Total #Transistors per Chip: exponential increase (*Moore’s Law*)

- 1971: 2300 (Intel 4004 microprocessor)
- 2016 and beyond: 1-5 billion+

Design Challenges:

- system complexity, design time, clock distribution
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
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

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

**Lower Power**

- no clock
  - components inherently use dynamic power only "on demand"
  - no global clock distribution
  - effectively provides automatic clock gating at arbitrary granularity

**Robustness, Scalability, Modularity: “Lego-like” construction**

- no global timing: plug-and-play design
  - “mix-and-match” variable-speed components, different block sizes
  - supports dynamic voltage scaling
  - modular design style ➔ “object-oriented”

**Higher Performance (… sometimes)**

- not limited to “worst-case” clock rate

**“Demand- (Data-) Driven” Operation**

- instantaneous wake-up from standby mode
Example: Current Comparison – 80c51 Microcontroller


(Philips Semiconductors, 2000)
Potential Targets

Large variety of asynchronous design styles

- Address different points in “design-space” spectrum...
  - **extreme timing-robustness:**
    - supports unknown transmission times, arbitrary inter-bit skews
    - **PVT variation tolerant:** providing near “delay-insensitive (DI)” operation
  - **ultra-low power, energy:**
    - “on-demand” operation, instant wakeup
    - sub-/near-threshold benefits: J. Rabaey, K. Roy, S. Nowick/M. Seok
  - **ease-of-design/moderate performance/low EMI** (electro-magnetic interference)
    - e.g. goal at Philips Semiconductors
  - **very high-speed:** asynchronous pipelined systems
    - ... comparable throughput to high-end synchronous design
    - **with added benefits:** lower system latency, support variable I/O rates
  - **modular heterogeneous systems:** integrate clock domains via async
    - “GALS-style” (globally-async/locally-sync)
  - **use in emerging technologies:** QCA, CNT, nano-magnetics, etc.
Components usually communicate & synchronize on **channels**

**Overview: Asynchronous Communication**

Sender  

Receiver  

channel

**without data**
Overview: Signalling Protocols

Communication channel: usually instantiated as 2 wires

Sender

Receiver

without data
Overview: Signalling Protocols

4-Phase Handshaking

Active (evaluate) phase

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

Return-to-zero (RZ) phase
Overview: Signalling Protocols

Sender  →  Receiver

First communication

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

2-Phase Handshaking = “Transition-Signalling”

Second communication

req

ack
Overview: How to Communicate Data?

Data channel: replace “req” by (encoded) data bits
- ... still use 2-phase or 4-phase protocol

Sender  
\[\text{data}\]  
Receiver

ack
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, input arrival time, etc.

- **4-phase (RZ) protocols:**
  - dual-rail (1-of-2): *widely used!*
  - 1-of-4
  - m-of-n

- **2-phase (NRZ) protocols:**
  - transition-signaling (1-of-2)
  - LEDR (1-of-2) [“level-encoded dual-rail”] [Dean/Williams/Dill, Adv. Research in VLSI ’91]
  - LETS (1-of-4) [“level-encoded transition-signalling”] [McGee/Agyekum/Mohamed/Nowick IEEE Async Symp. ’08]
Overview: How to Encode Data?

DI Codes: “dual-rail”: 4-Phase (RZ)

<table>
<thead>
<tr>
<th>Bit X</th>
<th>Dual-rail encoding X1 X0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 1</td>
</tr>
<tr>
<td>1</td>
<td>1 0</td>
</tr>
<tr>
<td>no data</td>
<td>0 0 = NULL (spacer)</td>
</tr>
</tbody>
</table>

Sender

Receiver

X1
X0
Y1
Y0

ack
Overview: How to Encode Data?

DI Codes:
“1-of-4”: 4-Phase (RZ)

<table>
<thead>
<tr>
<th>Bits A B</th>
<th>Dual-rail encoding X3 X2 X1 X0</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0 0 0 1</td>
</tr>
<tr>
<td>01</td>
<td>0 0 1 0</td>
</tr>
<tr>
<td>10</td>
<td>0 1 0 0</td>
</tr>
<tr>
<td>11</td>
<td>1 0 0 0</td>
</tr>
<tr>
<td>no data</td>
<td>0 0 0 0 = NULL (spacer)</td>
</tr>
</tbody>
</table>
More advanced DI codes:

- **M-of-N codes**: 3-of-6, 2-of-7, etc.
  - Provide better coding efficiency + dynamic power
  - Used in U. of Manchester “Spinnaker” Project – neuromorphic processors

- **“DI Bus-Invert” codes**: [Agyekum/Nowick DATE-11]
  - Provide better coding efficiency + dynamic power

- **“Zero-Sum” codes**: [Agyekum/Nowick DATE-10, IEEE TVLSI-12]
  - Provide fault tolerance (error detection/correction)

- **“LETS” codes**: 2-phase [McGee/Agyekum/Mohamed/Nowick Async-08]
  - Provide better dynamic power + higher throughput
  - Used in Stanford “Neurogrid” Project – neuromorphic processors
Overview: How to Encode Data?

Single-Rail “Bundled Data” -- with timing constraints

Uses synchronous single-rail data (potentially glitchy!)
+ local worst-case matched delay

“bundling” signal

req

A
B

data “bundle”

ack

Sender

Receiver
High-Speed Asynchronous Pipelines

“PIPELINED COMPUTATION”: like an assembly line

SYNCHRONOUS

go global clock

ASYNCHRONOUS

no global clock
MOUSETRAP: A Basic FIFO (no computation)

Stages communicate using transition-signaling (2-phase):

Features: standard cell design, single D-latch register per stage

[Singh/Nowick, IEEE Trans. on VLSI Systems (June 2007), ICCD (2001)]
**“MOUSETRAP” Pipeline: adding computation**

**Function Blocks:**
- use "synchronous" logic blocks (not hazard-free!)
- + a local "matched delay" (req)

**“Bundled Data” Requirement (1-sided):**
- Each req must arrive after data inputs valid and stable
**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

Asynchronous Design: a Brief History...

Phase #1: Early Years (1950’s-early 1970’s)
- Macromodules Project: plug-and-play design (Washington U., Wes Clark/C. Molnar)
- Commercial graphics/flight simulation systems: LDS-1 (Evans & Sutherland, C. Seitz)
- Basic theory, controllers: Huffman, Unger, McCluskey, Muller

Phase #2: The Quiescent Years (mid 1970’s-early 1980’s)
- Advent of VLSI era: leads to synchronous domination and major advances

Phase #3: Coming of Age (mid 1980’s-late 1990’s)
- Re-inventing the field:
  - correct new methodologies, controllers, high-speed pipelines, basic CAD tools
  - initial industrial uptake: Philips Semiconductors products, Intel/IBM projects
  - first microprocessors: Caltech, Manchester Amulet [ARM]

Phase #4: The Modern Era (early 2000’s-present)
- Leading applications, commercialization, tool development, demonstrators
Asynchronous Design: Recent Developments

1. Philips Semiconductors: low-/moderate-speed embedded systems

- **Wide commercial use:** >700 million async chips (mostly 80c51 microcontrollers)
  - consumer electronics: *pagers, cell phones, smart cards, digital passports, automotive*
  - commercial releases: 1990’s-2000’s

- **Benefits (vs. sync):**
  - 3-4x lower power (and lower energy consumption/op)
  - 5x lower peak currents
  - much lower “electromagnetic interference” (EMI) – no shielding of analog components
  - correct operation over wide supply voltage range
  - instant startup from stand-by mode (no PLL’s)

- **Complete commercial CAD tool flow:** synthesis/testing, design-space exploration
  - “Tangram”: Philips (late 1980’s-early 2000’s)
  - “Haste”: Handshake Solutions (incubated spinoff, early-late 2000’s)
1. Philips Semiconductors (cont.)

- **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)
      - Integrated flow with Synopsys/Cadence/Magma tools
  - +: fast, ‘transparent’, easy-to-use
  - -: few optimizations, low/moderate-performance only
Asynchronous Design: Recent Developments

2. Fulcrum Microsystems/Intel: high-speed Ethernet switch chips

* Async start-up out of Caltech ➔ now Intel’s Switch & Router Division (SRD) (2011)
* Target: low system latency, extreme functional flexibility
* Intel’s FM5000-6000 Series (~2013 release)
  * 72-port 10G Ethernet switch/router
  * Very low cut-through latency: 400-600ns
  * 90% asynchronous ➔ external synchronous interfaces
  * 1.2 billion transistors: largest async chip ever manufactured (at release time)
  * > 1 GHz asynchronous performance (65 nm TSMC process)
* CAD flow:
  * semi-automated, incl. spec language (CAST)


- Developed out of DARPA’s SyNAPSE Program
- Massively-parallel, fine-grained neuromorphic chip
  - Fully-asynchronous chip! → neuronal computation (bundled data) + interconnect (DI)
  - IBM’s largest chip ever: 5.4 billion transistors
  - Models 1 million neurons/256 million synapses → contains 4096 neurosynaptic cores
    - ... MANY-CORE SYSTEM!
  - Extreme low energy: 70 mW for real-time operation → 46 billion synaptic ops/sec/W
  - Asynchronous motivation: extreme scale, high connectivity, power requirements, tolerance to variability

Example network topology:
showing only 64 cores (out of 4096)
[IBM, 2014*]

3. Neuromorphic Chips: Other Recent Async/GALS Processors

a. U. of Manchester (UK): SpiNNaker Project, ~2005-present (S. Furber et al.)
   * GALS systems: many-core ARM-based systems + async NoC’s: single-chip/multi-chip

b. Stanford University: Neurogrid Project (Brains in Silicon) (K. Boahen et al.)
   * Uses analog neurons + async digital synapses (interconnect)
   * Scientific American (May 2005) – cover story
   * Proceedings of the IEEE (May 2014)
   * uses our delay-insensitive “LETS” codes for robust inter-neuron communication

c. Intel Labs (Hillsboro, OR): new research project (Kshitij Bhardwaj [2006])

➤ Each uses robust async NoC’s to integrate massively-parallel many-core system

- Highly-reconfigurable accelerator-based many-core GALS architecture
- Entirely asynchronous NoC: enables fine-grain power- & variability-management
- First prototype: delivered 80 GOPS performance with only 2W power consumption
- Has evolved into the company’s “STHORM” Platform (2014)

Asynchronous Design: Recent Developments

5. Columbia/AMD Research: high-performance/low-energy NoC’s

- **Ongoing collaboration w/our group (2015-):** under DOE “Exascale Project”
  - Weiwei Jiang: project lead (6 month internship)

- **Target:** implement async NoC switch in advanced industrial 14nm FinFET library
  - **Application:** system configuration + power/performance monitoring (GPU/CPU chips)
  - Uses new async VC approach [*credit-based*]
  - **Initial tool flow:** harness sync design validation + physical design flow (some manual)

- **Experimental results (pre-layout):** direct comparison to AMD commercial NoC
  - **sync:** uses fine-grain clock gating
  - **async:** 55% less area, 28% lower latency, power savings = 88% (idle)/58% (active)

![Bar chart comparison of sync vs. async in terms of area, latency, idle power, and active power.](chart.png)

**Actual layout for proposed async router:** Weiwei Jiang (Columbia)/Greg Sadowski (AMD)
Asynchronous Design: Recent Developments

6. Computational Units/Embedded Subsystems

(a) Fast Huffman Decoder for Compressed-Code Embedded Processors
- For compressed memory storage: decompress on-the-fly during cache refill
- Async decoder: optimized for average-case Huffman codes (variable processing rate)
- higher throughput than state-of-art synchronous decoders at the time (+ low area)

(b) Floating-Point Adder
- Cornell Group: Sheikh/Manohar
- Exploits data-dependent optimization, micro-level concurrency
- Leading combination of performance and energy-efficiency

(c) Laser Space Measurement Chip (Columbia joint w/NASA Goddard [2006-2008])
- For “time-of-flight” measurement in science missions (laser altimeters, mass spectrometers)
- Async design: significantly lower power + area vs. NASA-deployed synchronous chip
- Meets all performance targets, eliminates high-speed sampling clock
7. Emerging Technologies/New Paradigms

- **Ultra-Low Energy:** sub-threshold/near-threshold computing
  - Async is highly-robust to timing variability (PVT)
  - Key results: Rabaey’s group (UCB), Kaushik Roy’s group (Purdue), Nowick/Seok (CU)

- **Energy Harvesting**
  - Async “event-driven” logic, adapts to highly-variable power availability
  - Christmann/Beigne (CEA-LETI): 40% power efficiency gain vs. synchronous

- **Continuous-Time DSP’s (CT-DSP’s)**
  - Nowick/Tsividis collaboration [2010-2015]
  - Variable sampling-rate DSP’s: avoids aliasing, highly reusable, low energy

- **Handling Extreme Environments:** space, terrestrial
  - E.g. support full operation over 400° C temperature range

- **Use with Emerging Technologies**
  - Flexible electronics: bending material induces unpredictable and large delay variations
    - (i) Seiko/Epson ACT11 microprocessor (ISSCC-05), (ii) Ogras group (ASU)
  - Nano-magnetics
  - Quantum cellular automata (QCA)
A Reading List

Overview/survey articles: introduction to asynchronous/GALS design


Our asynchronous/GALS network-on-chip (NoC) research:

1. **Basic 5-ported switch design + semi-automated tool flow:**

2. **Support for virtual channels (VC’s):**

3. **N-way asynchronous arbiters:**

4. **Performance acceleration:**

5. **Support for efficient multicast:**