Introduction

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

![Diagram of Synchronous System with clock and components](image-url)
**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”

“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
   - 2015: 2-6 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)
   - 2015 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
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**

- **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**

![Current Draw Graphs](image)

(Philips Semiconductors, 2000)


---

**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)
Overview: Asynchronous Communication

Components usually communicate & synchronize on channels.

Sender  
channel  
Receiver

without data

Overview: Signalling Protocols

Communication channel: usually instantiated as 2 wires.

Sender  
req  
ack
Receiver

without data
Overview: Signalling Protocols

4-Phase Handshaking

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

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

![Diagram of sender and receiver with data and ack signals](image)

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”] [McGeer/Agyekum/Mohamed/Nowick, IEEE Async Symp. ‘88]
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>

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>

Overview: How to Encode Data?

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
data “bundle”

Sender

Receiver
**High-Speed Asynchronous Pipelines**

"PIPEDINED COMPUTATION": like an assembly line

- **Synchronous**
  - Global clock
  - Stages connected with no global clock

- **Asynchronous**
  - No global clock

**MOUSETRAP: A Basic FIFO (no computation)**

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

- Latch Controller
- Data Latch
- 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

Mixed-Timing Interfaces: Challenge

Goal: provide low-latency communication between "timing domains"
Challenge: avoid synchronization errors
Mixed-Timing Interfaces: Solution

Asynchronous Domain 1 → Async-Sync FIFO → Synchronous Domain 1

Async-Sync FIFO

Async-Sync FIFO

Synchronous Domain 2

Synchronous Domain 2 → Mixed-Clock FIFO’s

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 Quiet 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 Industrial 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 Industrial 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)


Asynchronous Design: Recent Industrial Developments

- 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
  - 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*]

Asynchronous Design: Recent Industrial Developments

3. Neuromorphic Chips: Other Recent Async/GALS Processors
   a. U. of Manchester (UK): SpiNNaker Project, ~2005-present (S. Furber et al.)
   b. Stanford University: Neurogrid Project (Brains in Silicon) (K. Boahen et al.)
      - Scientific American (May 2005) – cover story
      - Proceedings of the IEEE (May 2014)
   ➔ Each uses robust async NoC’s to integrate massively-parallel many-core system

Asynchronous Design: Recent Industrial Developments

   - 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)

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]