Yale ENGINEERING

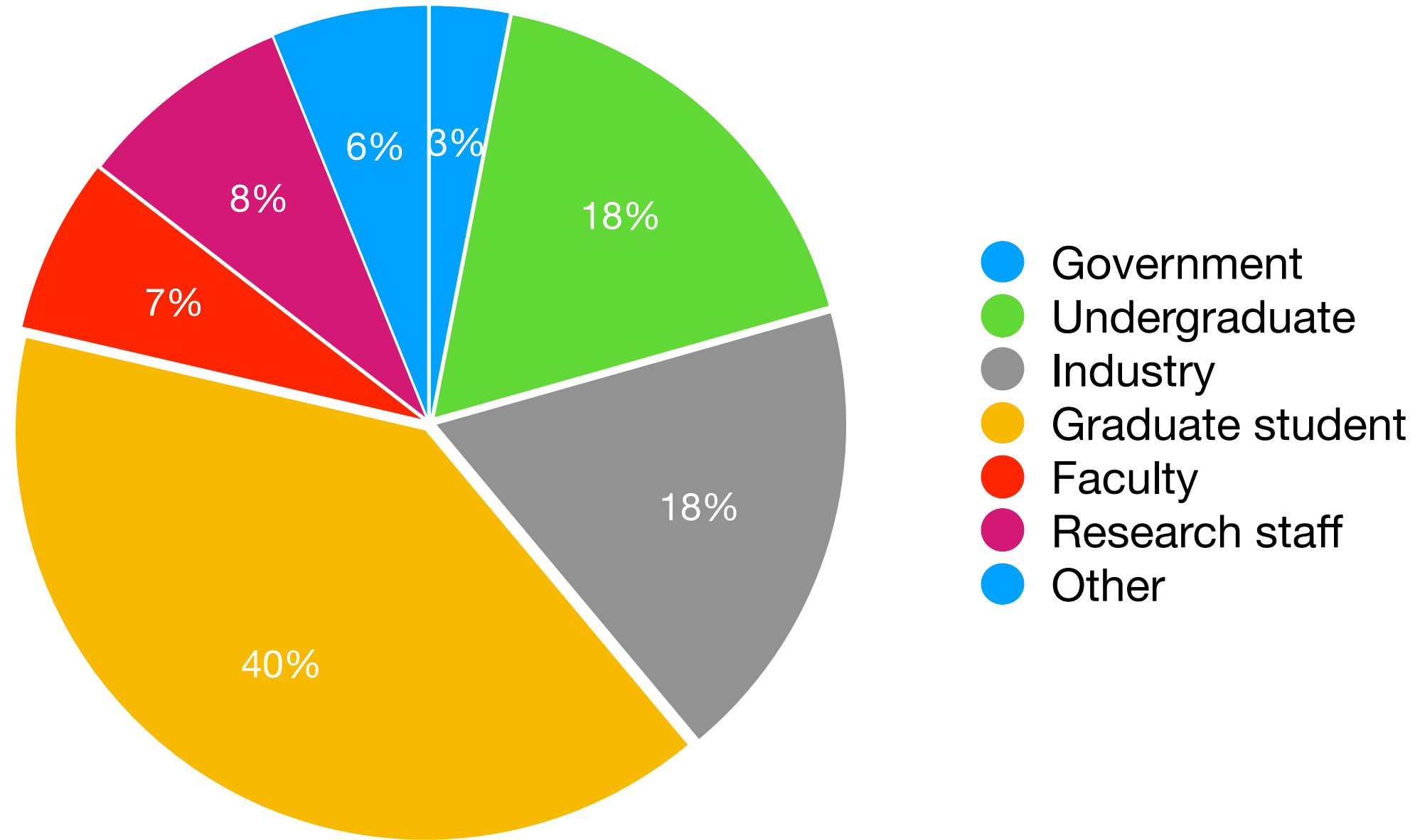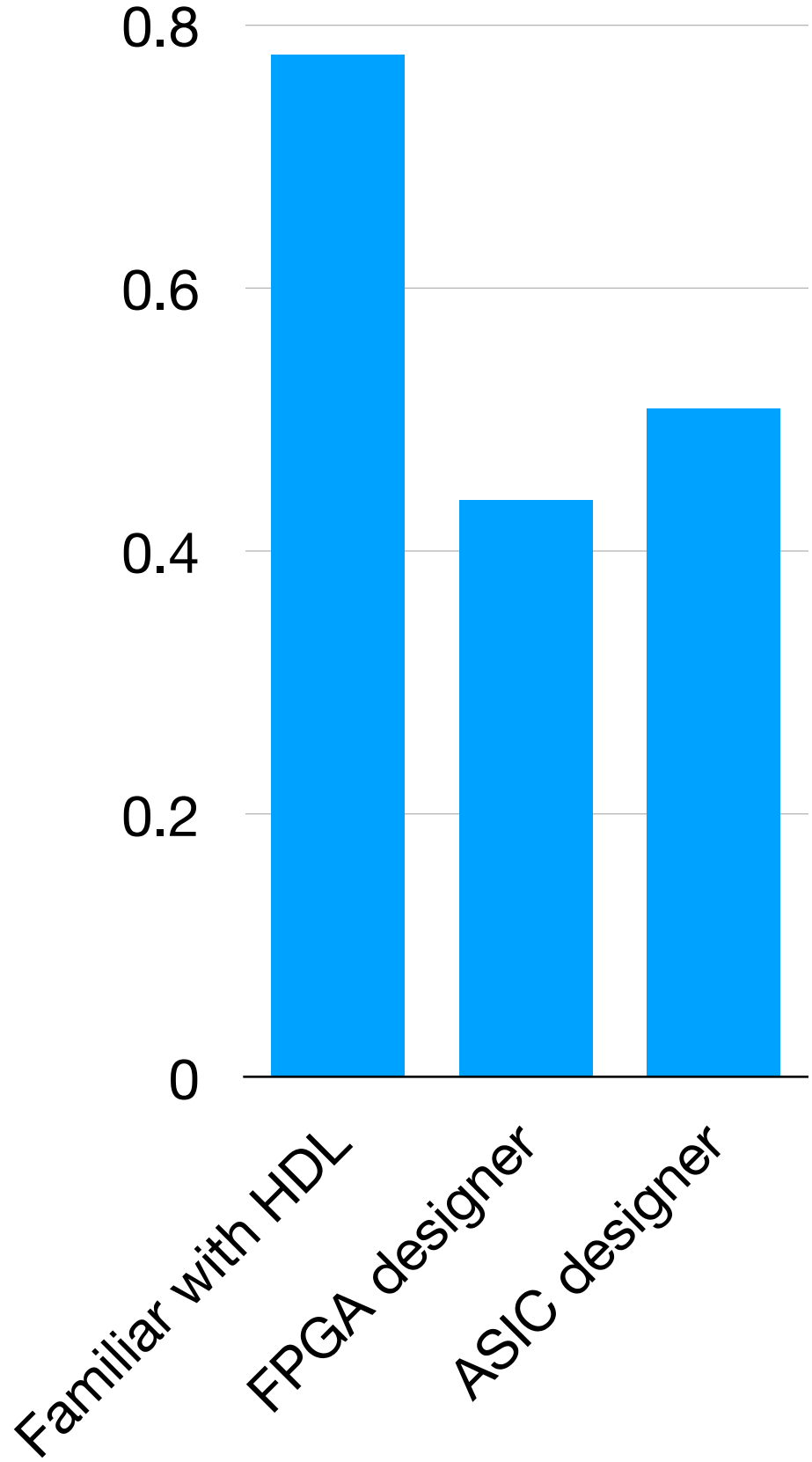# Introduction to the ASYNC summer school

AVLSI

Logistics

- Questions and Answers
  - ❖ Please use Mattermost!
    - ‣ Self-signup for account
      `https://bit.ly/3a6Xzto`

    - ‣ Mattermost link
      `https://avlsi.csl.yale.edu:8000/`
      Channel: summer2024

- Three weeks, 9:00AM to 1:00PM Eastern Time
  - ❖ Mon July 1: Behavioral design
  - ❖ Mon July 8: Gate-level design
  - ❖ Mon July 15: Circuit and physical design

- Slides, videos: `https://asyncsymposium.org/async2024/`

Mattermost self-signup link

Yale ENGINEERING

AVLSI

Registration summary

Attended 2022 summer school: 15%

Week 1 schedule

9:00 AM	Welcome	Ivan Sutherland
9:10 AM	Overview + introduction to asynchronous design	Rajit Manohar
9:40 AM	*break*	
9:45 AM	Message-passing behavioral description	Rajit Manohar
10:45 AM	Examples	
11:00 AM	Pipelined asynchronous circuits: basic performance estimation	Montek Singh
11:45 AM	Examples	
11:55 AM	*break*	
12:00 PM	From dataflow to gates	Montek Singh
1:00 PM	End of Day 1	

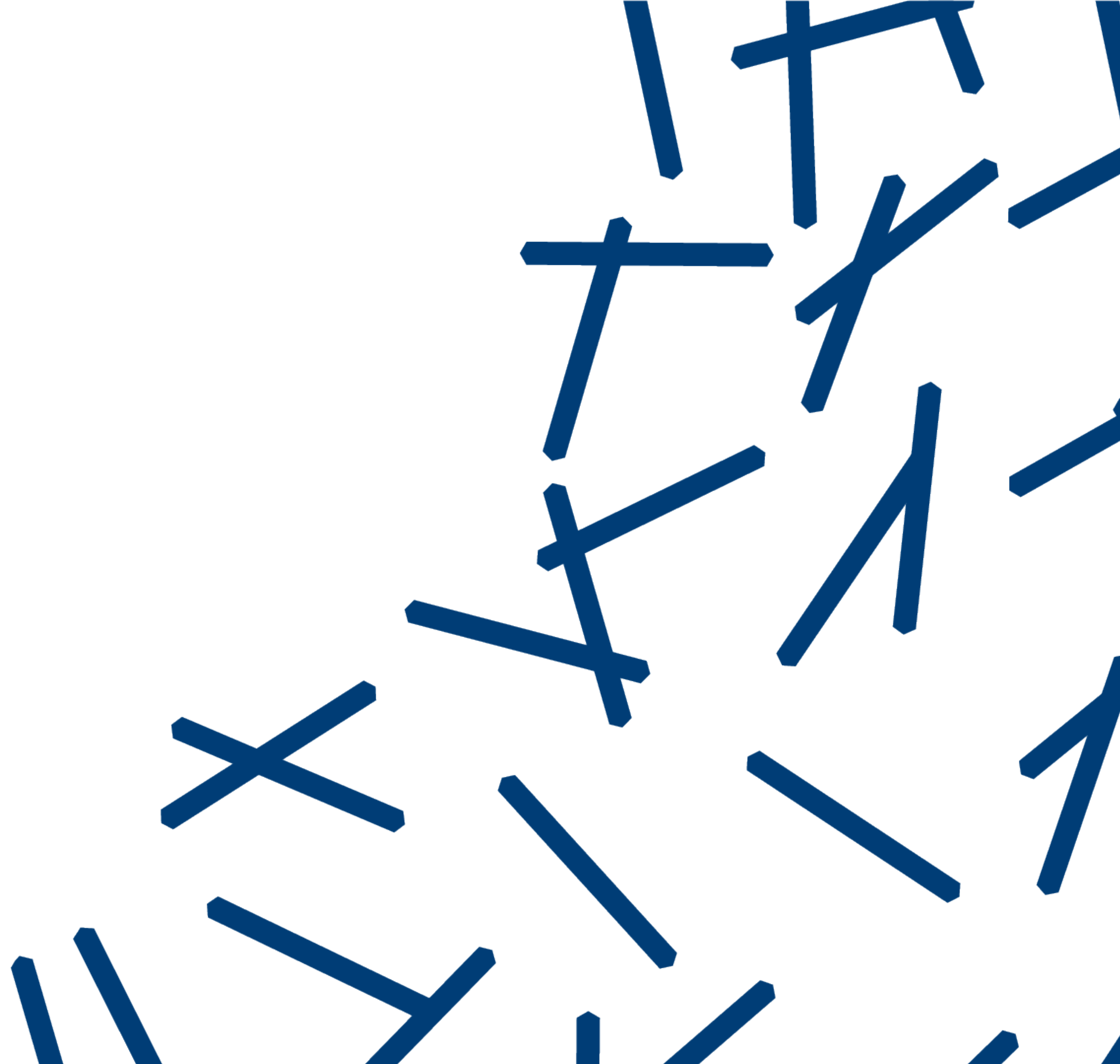Yale ENGINEERING

AVLSI

Relation to summer school from 2022

- Week 1 changes
  - ❖ 2022: link and joint abstraction for circuit-independent design
    - ▸ Slides, videos are available online: `https://avlsi.csl.yale.edu/act/doku.php?id=summer2022:start`
  - ❖ 2024: dataflow to gates

- Week 2 changes
  - ❖ 2022: focus on syntax-directed translation
  - ❖ 2024: Petri-net based synthesis

- Week 3 changes
  - ❖ 2022: custom cell ASIC flow
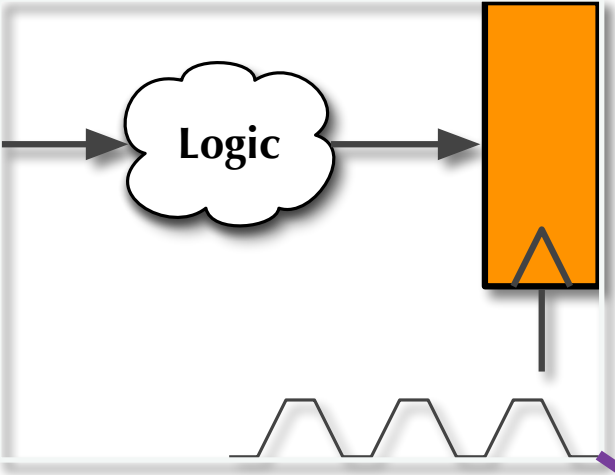 - ❖ 2024: integration with analog/mixed-signal flows and standard cell flows

Yale ENGINEERING

AVLSI

# Introduction to asynchronous design

Rajit Manohar

Approaches to computation

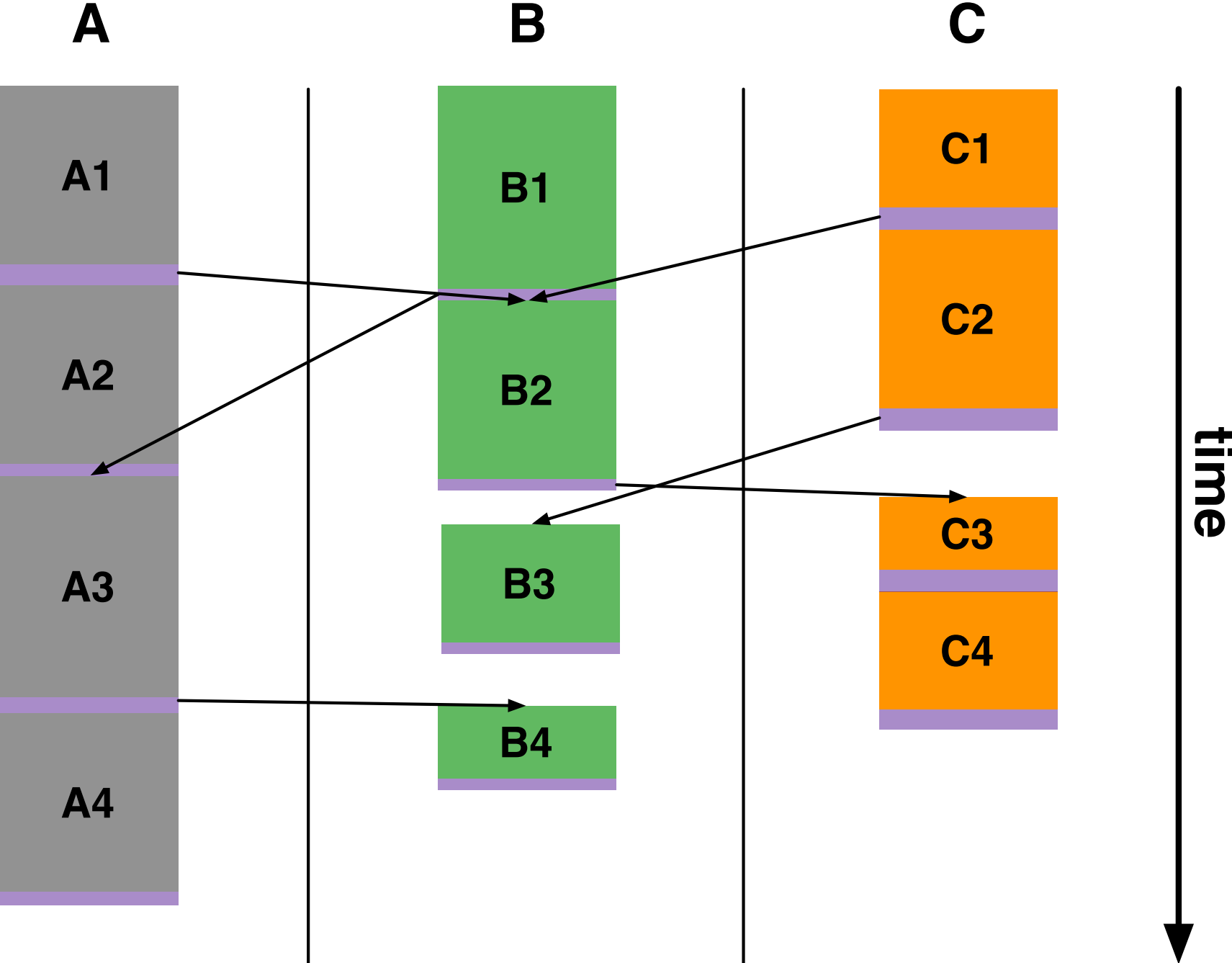|	Discrete Time	Continuous Time
Discrete Value	Digital, synchronous logic	Digital, asynchronous logic
Continuous Value	Switched-capacitor analog	General analog computation

- Event-driven at the (digital) circuit level of abstraction
  - ❖ "Data-driven computing"
  - ❖ "Self-timed computing"
  - ❖ "Asynchronous computing"
  - ❖ "Clockless computing"

Yale ENGINEERING
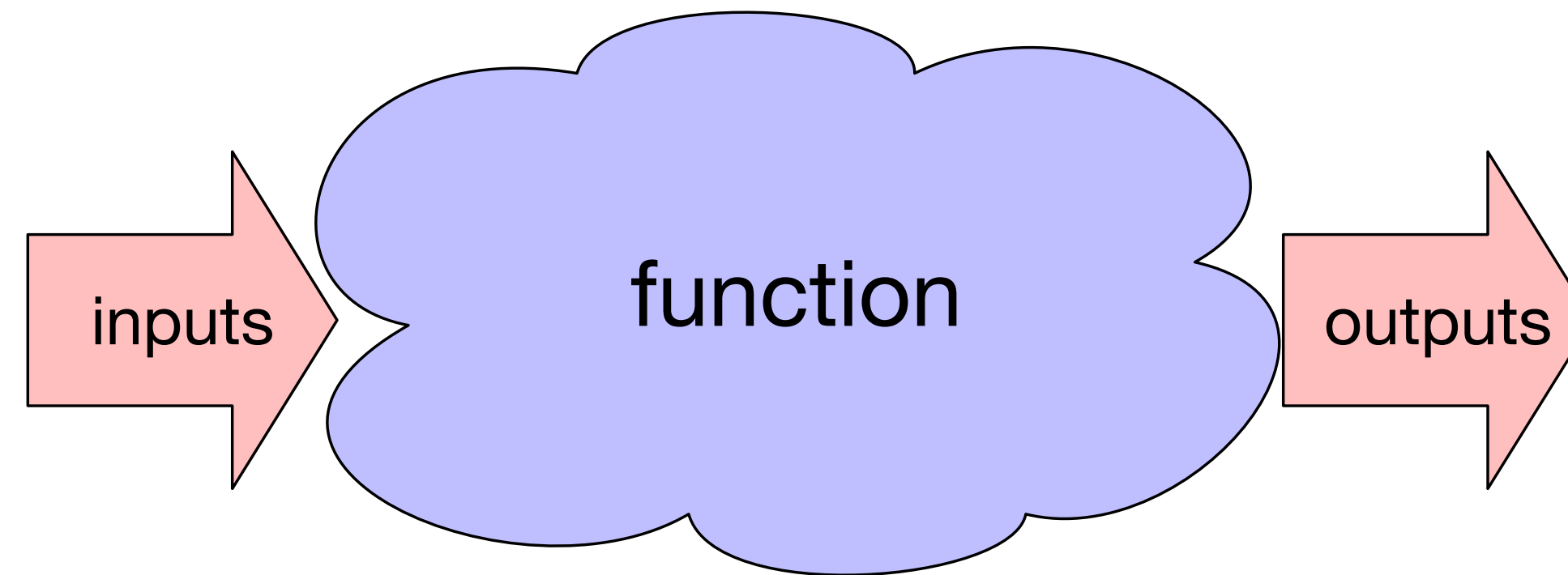
AVLSI
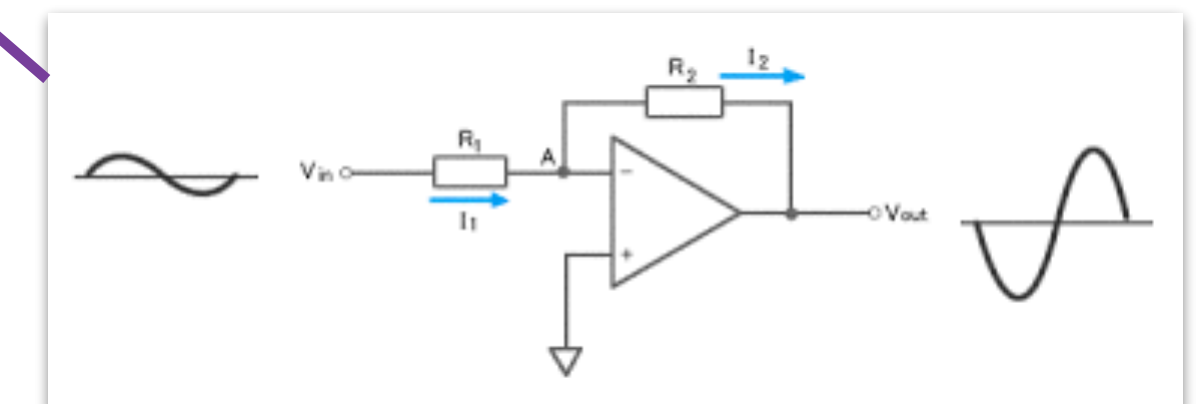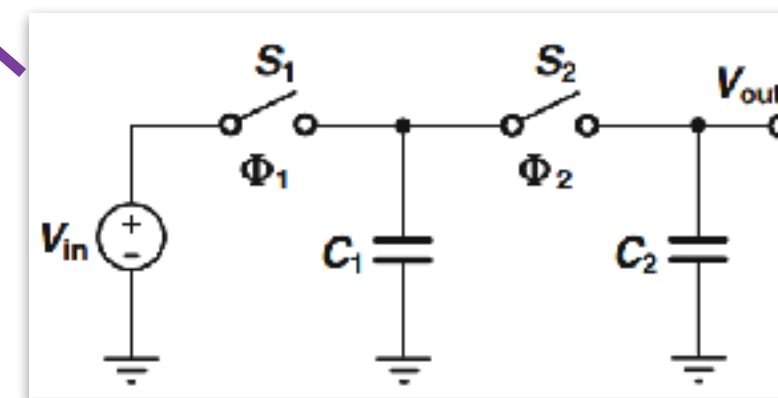
Discrete versus continuous time
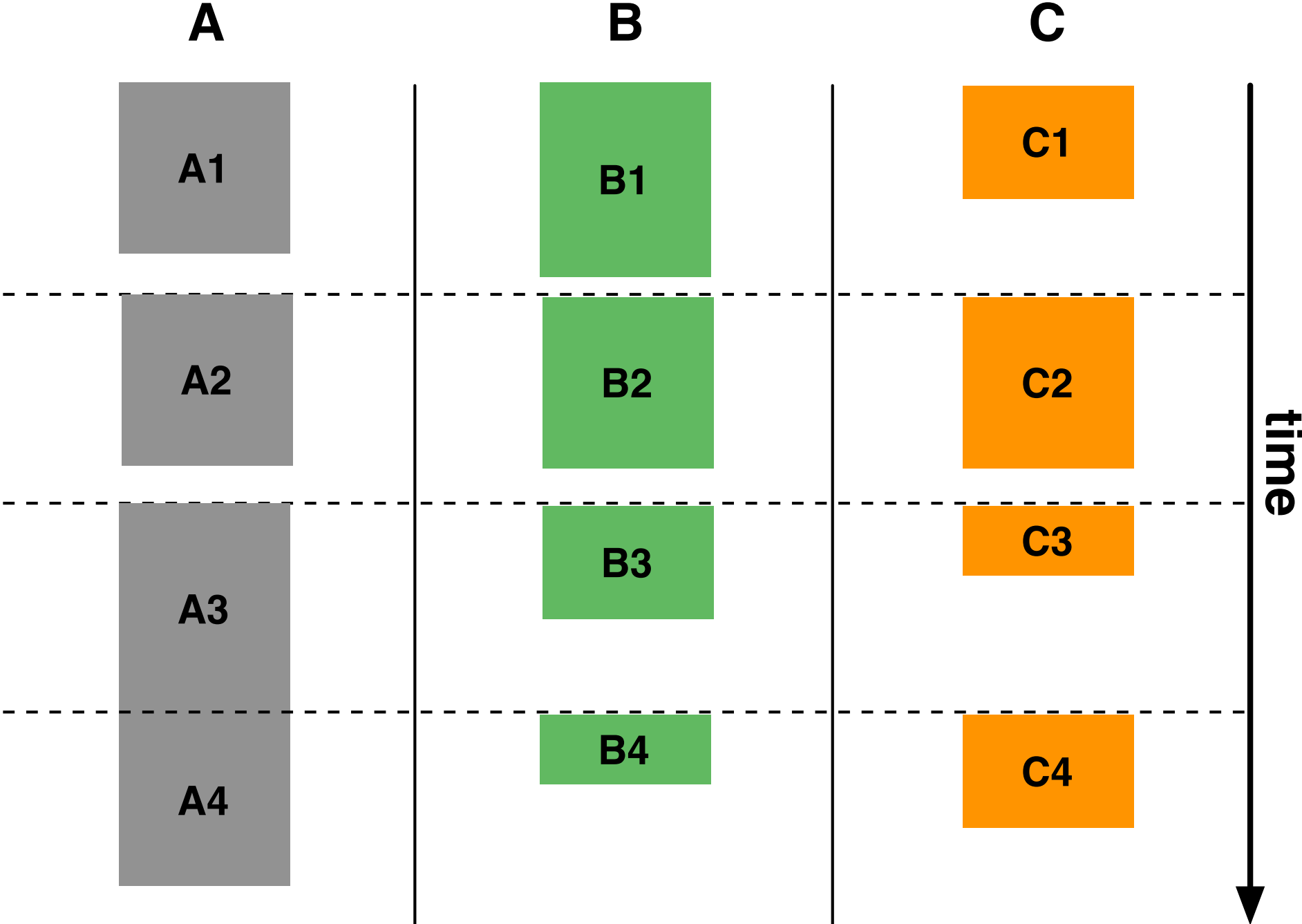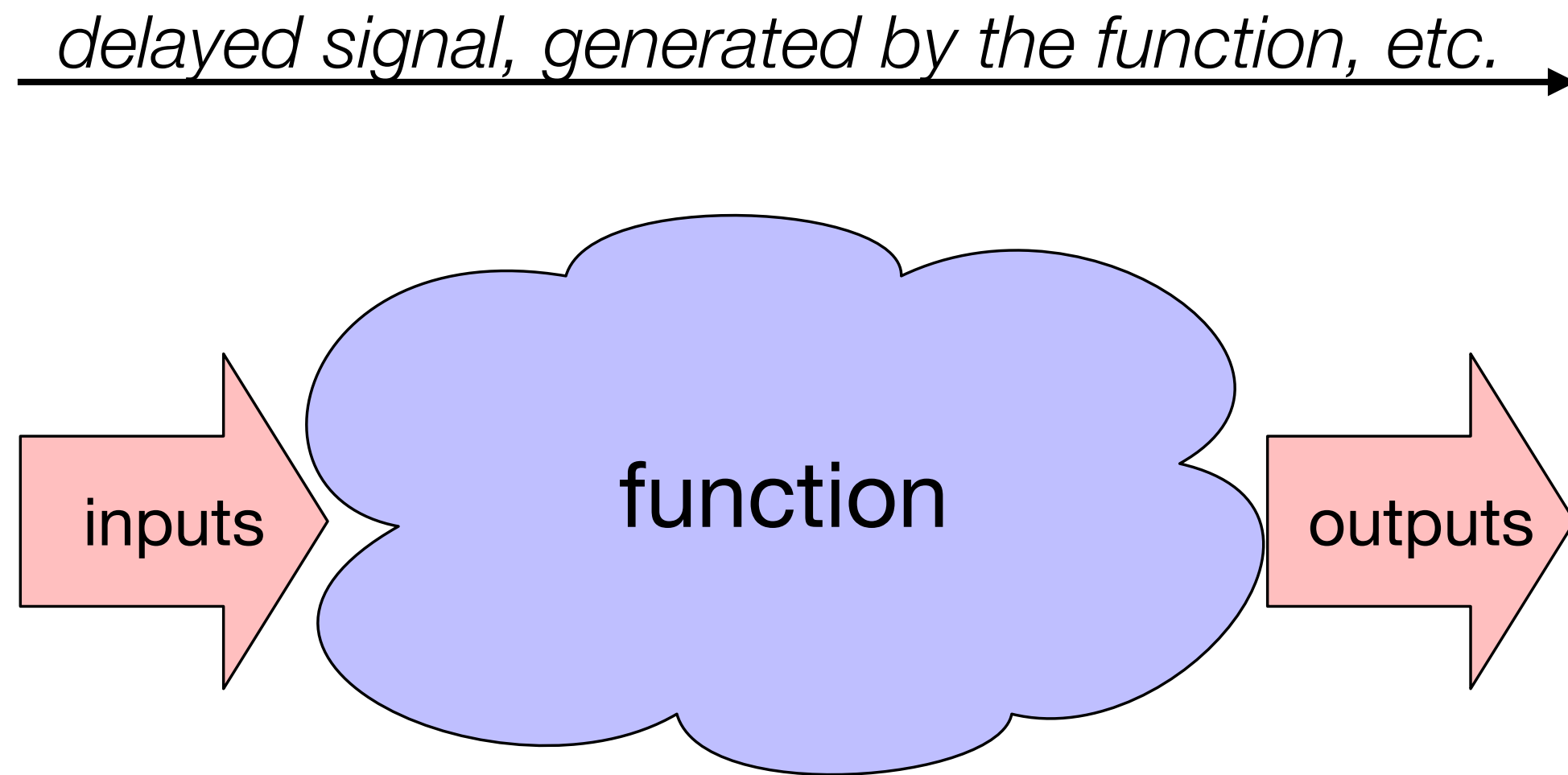

Discrete-time computation

Continuous-time computation

Computing asynchronously

- When the input arrives, the function is triggered
- Eventually the output is produced

Computing asynchronously

delayed signal, generated by the function, etc.

- When the input arrives, the computation is triggered
- Eventually the output is produced
  - ❖ We need to know **when** the output is ready

Computing asynchronously
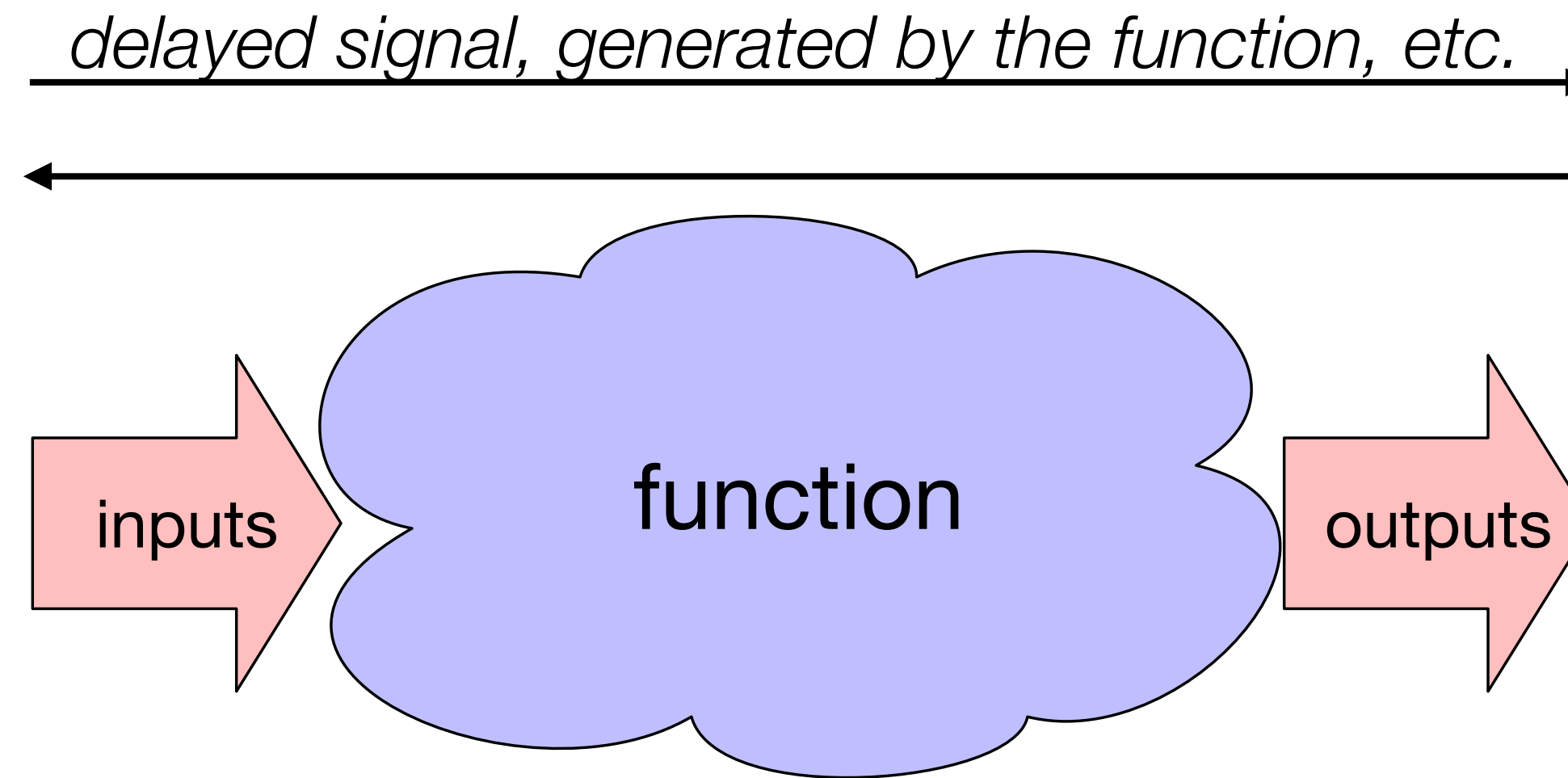
delayed signal, generated by the function, etc.

function

inputs

outputs

- When the input arrives, the computation is triggered

- Eventually the output is produced

  ❖ We need to know **when** the output is ready

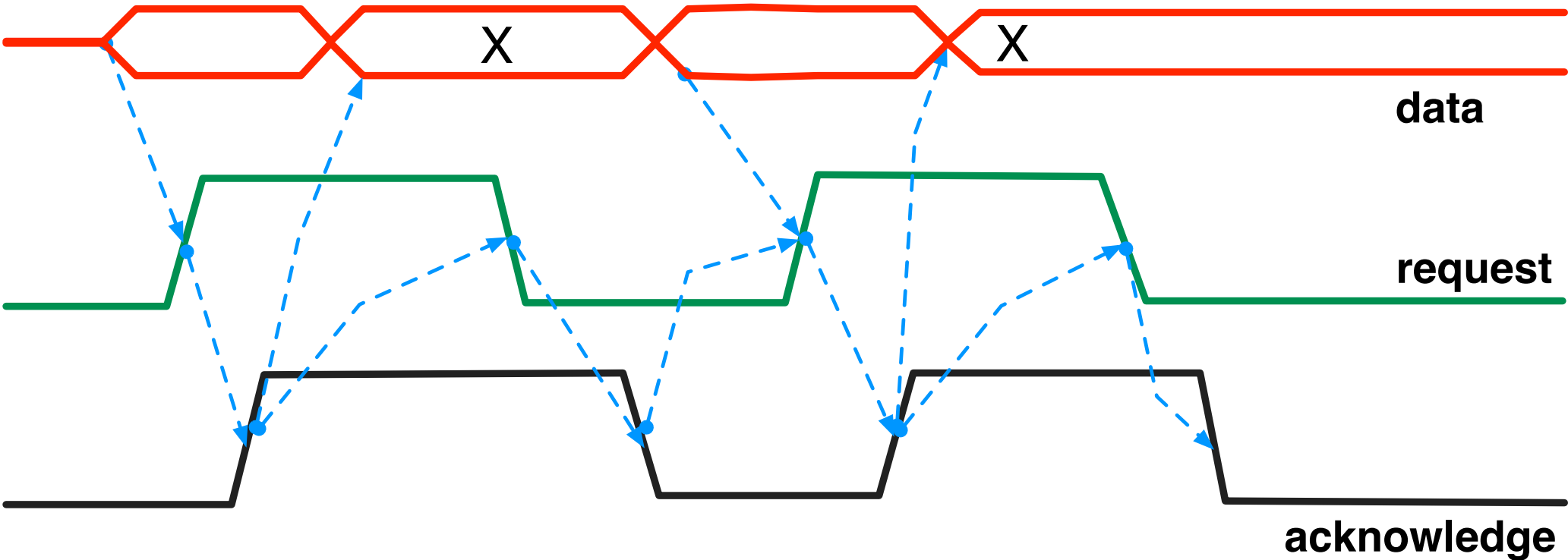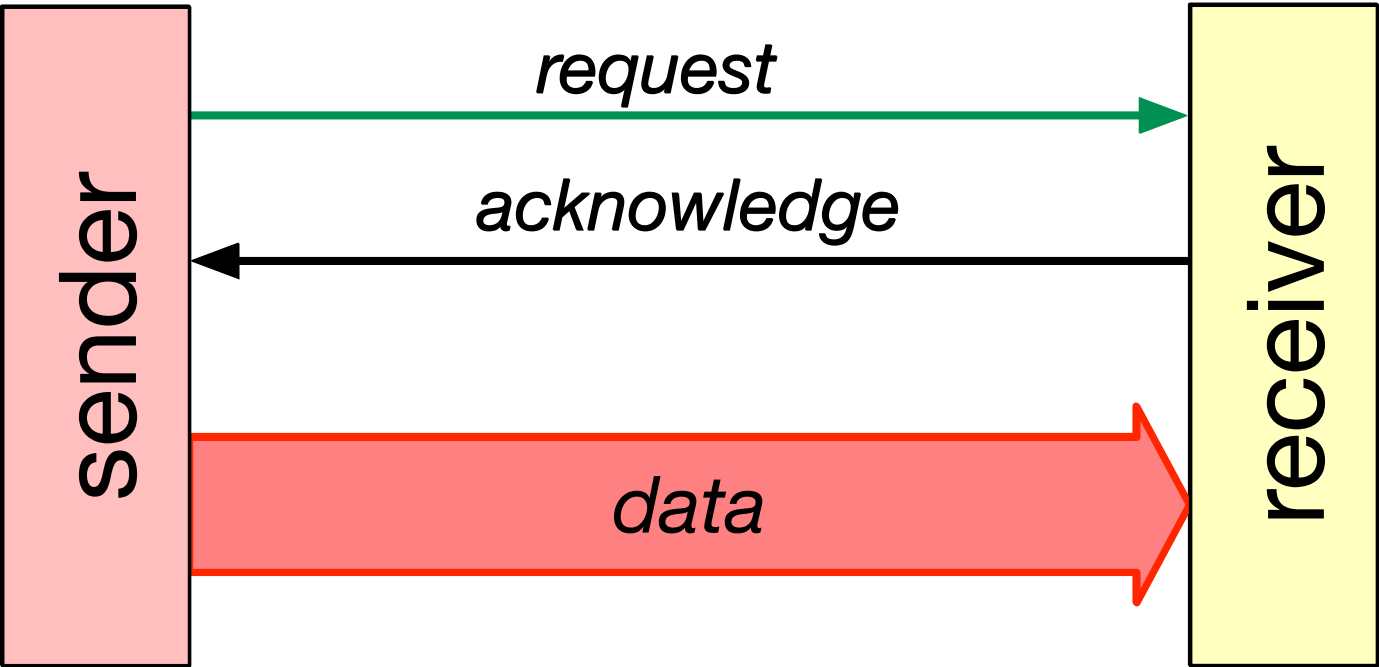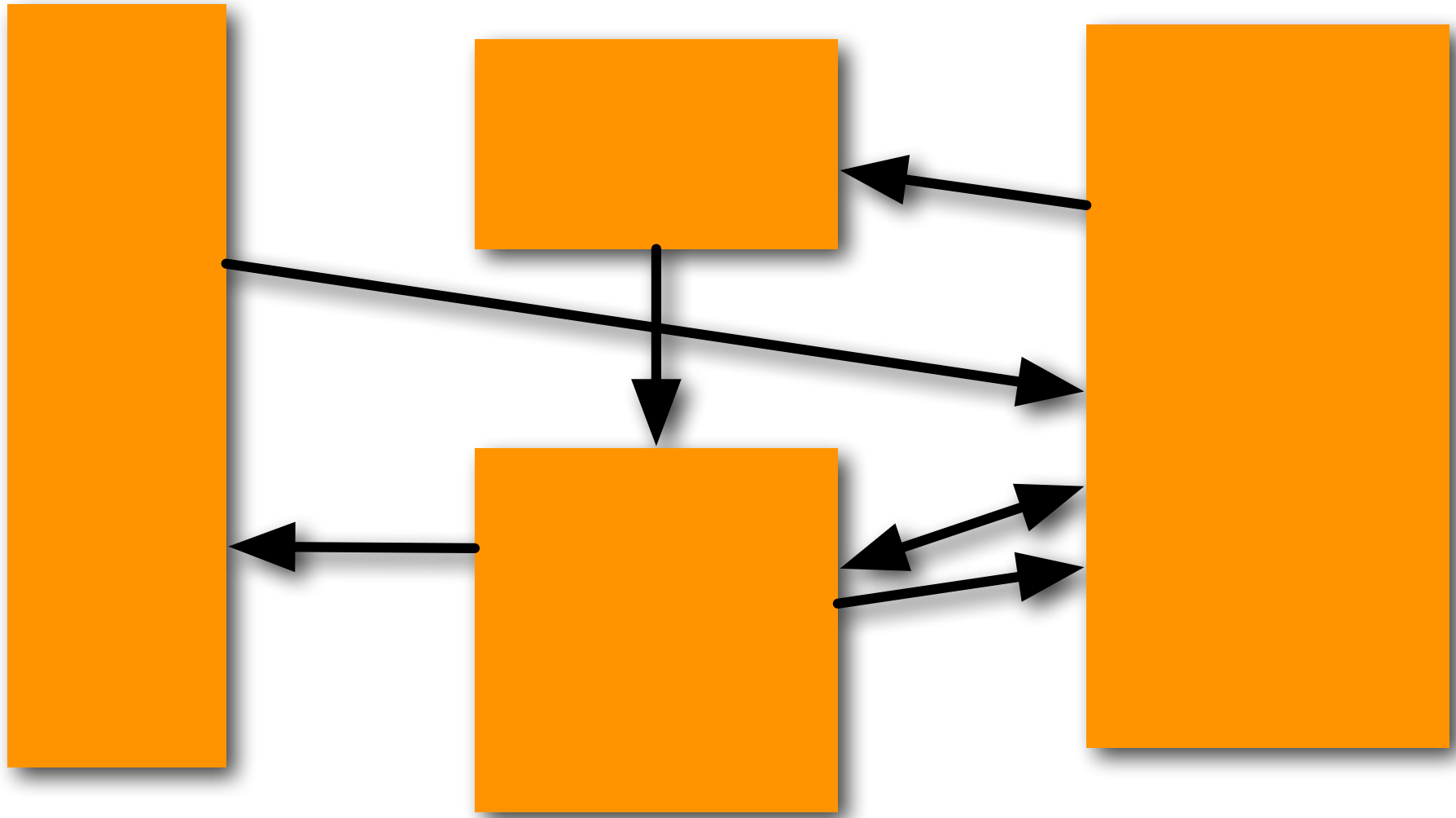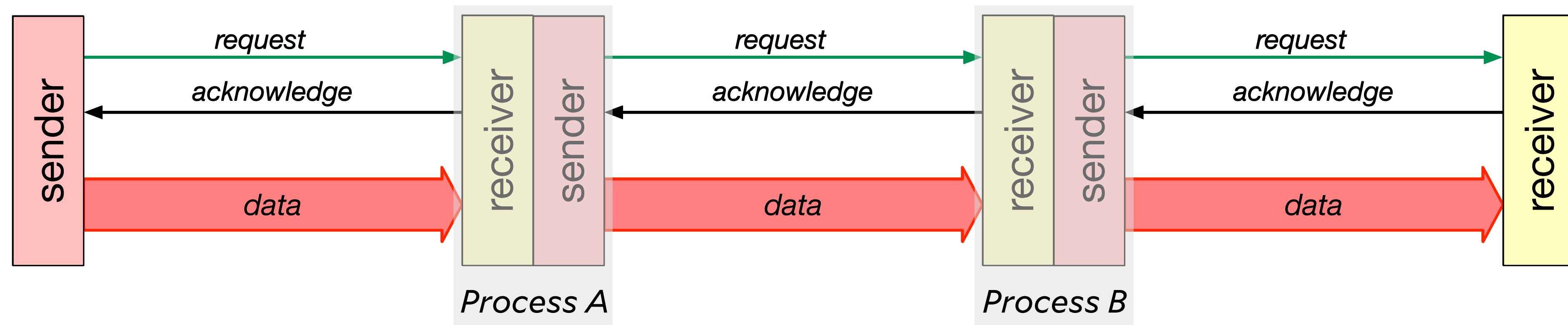 ❖ We need to know **when** we can produce the next output

Basic model

- Chip is a parallel program
  - ❖ Components: processes
  - ❖ Communication: via message-passing channels
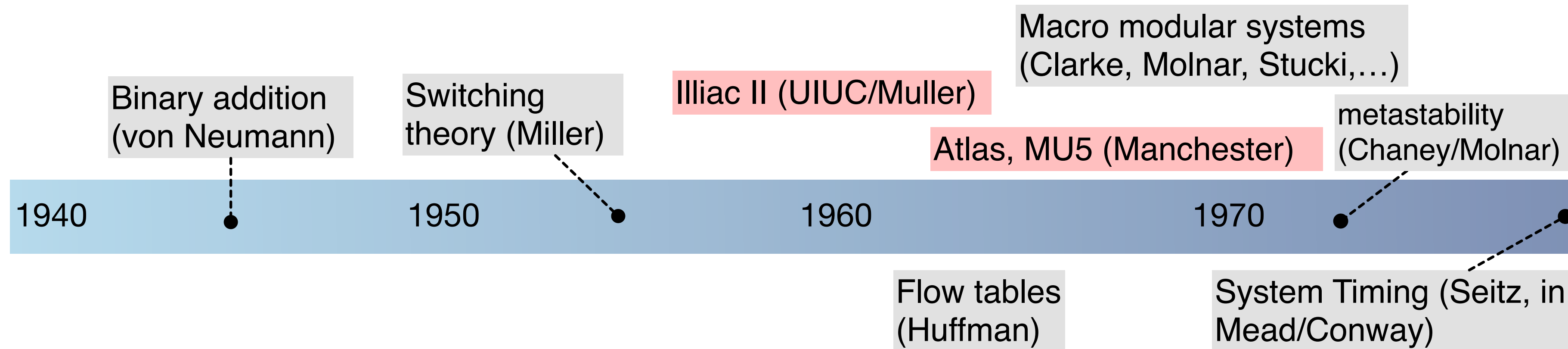 - ❖ Explicit synchronization through communication

A compute pipeline

- Data moves through the pipeline at its own pace

A little bit of (biased) history…

Macro modular systems
(Clarke, Molnar, Stucki,…)

Binary addition
(von Neumann)

Switching
theory (Miller)

Illiac II (UIUC/Muller)

metastability
(Chaney/Molnar)

Atlas, MU5 (Manchester)

| 1940 | 1950 | 1960 | 1970 |

Flow tables
(Huffman)

System Timing (Seitz, in
Mead/Conway)

"There was never any real question of using a clock. Atlas [1962] had been asynchronous, for good reasons, so the new machine would be. In any case, it was going to be physically so big that it would not have been any use having a clock."

— The History of MU5 (U. of Manchester)

Yale ENGINEERING

AVLSI

Managing noise in hand-wired computers...

- The physical issue...

  ❖ Signaling noise, and coupling between wires

- Wires were **manually** routed...

  ❖ ... so we don't know exactly where they are going to be!

- Instead

  ❖ Force **one** wire to be routed carefully (the clock)
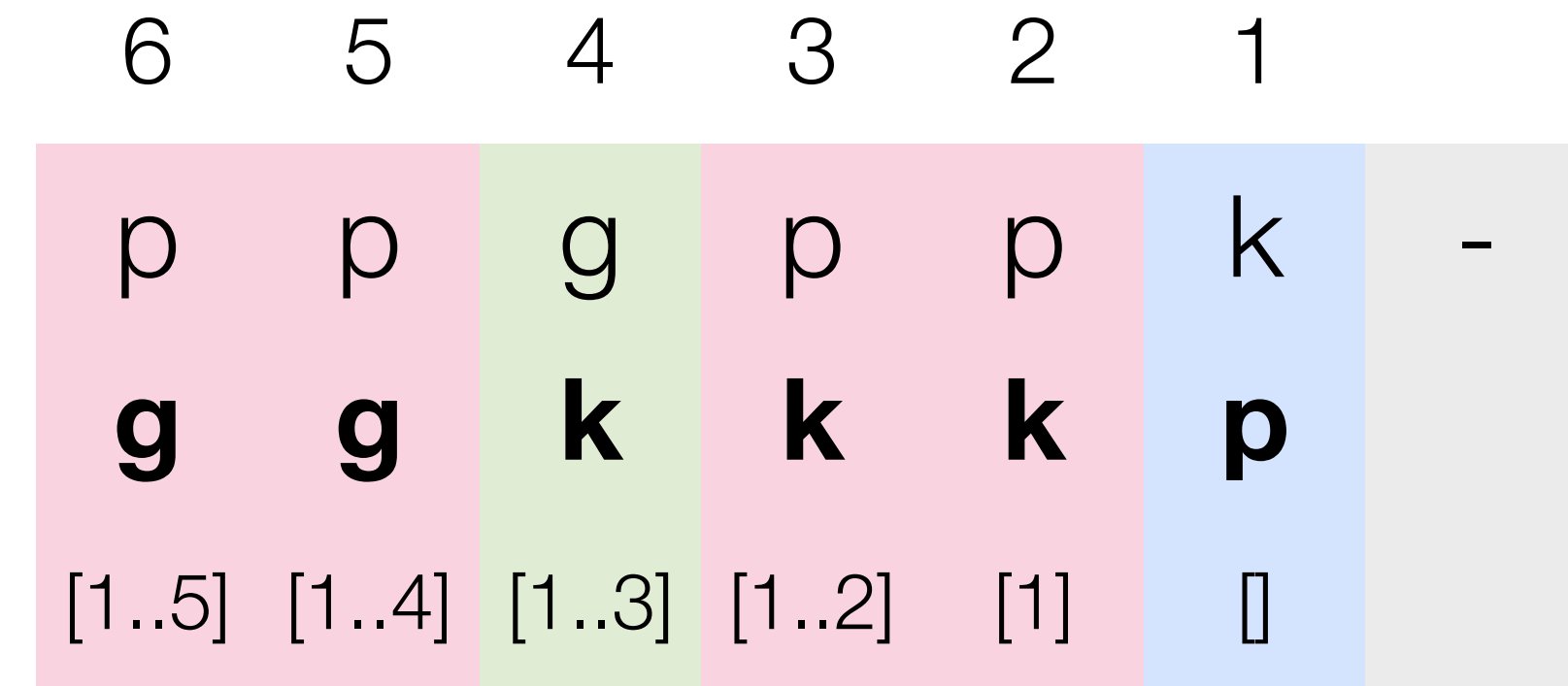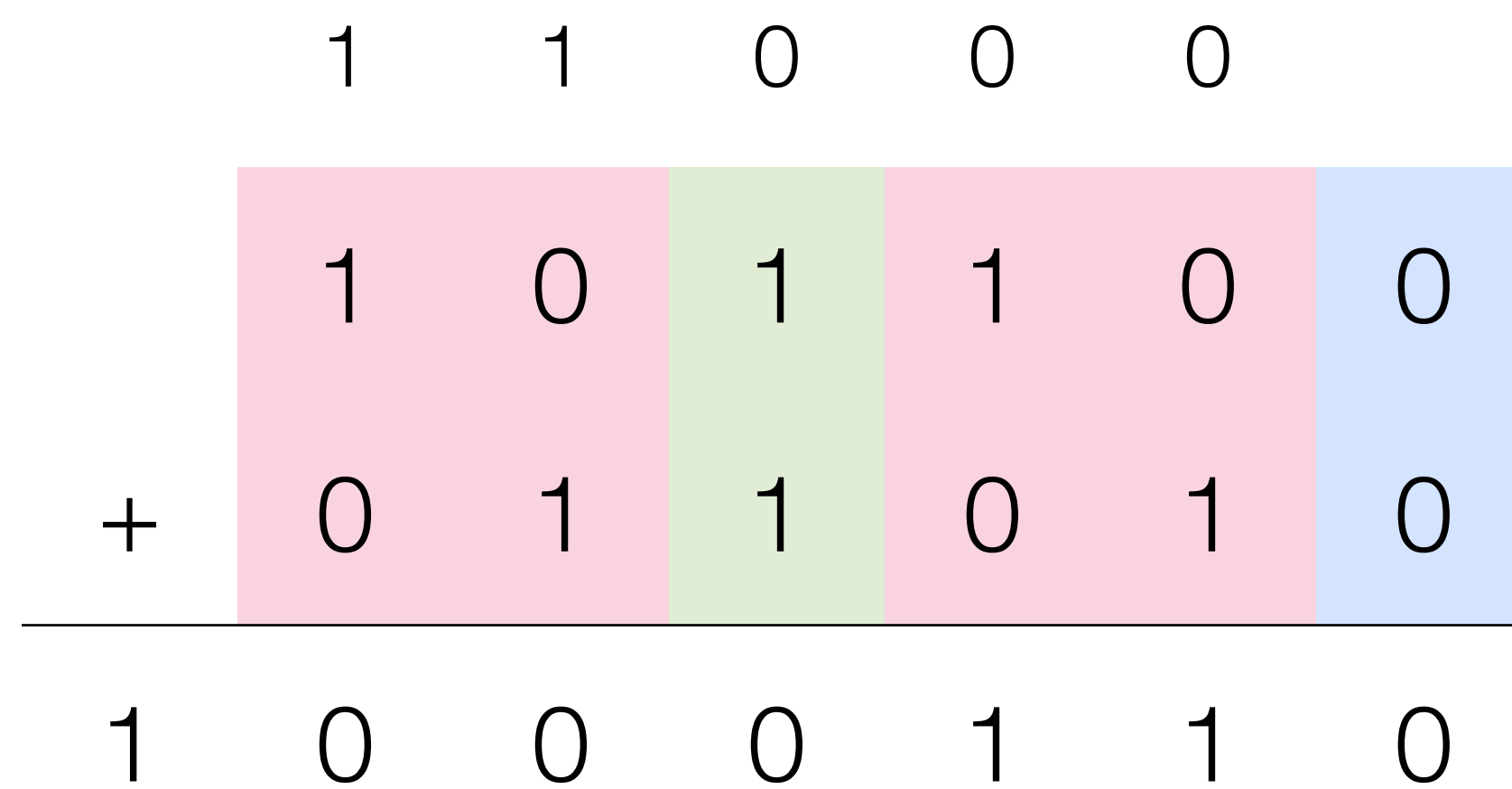
  ❖ Make it slow so that all the noise settles


The Atlas: backplane wiring

Yale ENGINEERING

AVLSI

An example: asynchronous binary addition

```
      1   1   0   0   0
      1   0   1   1   0   0
  +   0   1   1   0   1   0
  _____
  1   0   0   0   1   1   0
```

- Slow part: computing carry values
- Standard technique: classify input cases into three categories
  - ❖ kill
  - ❖ propagate
  - ❖ generate

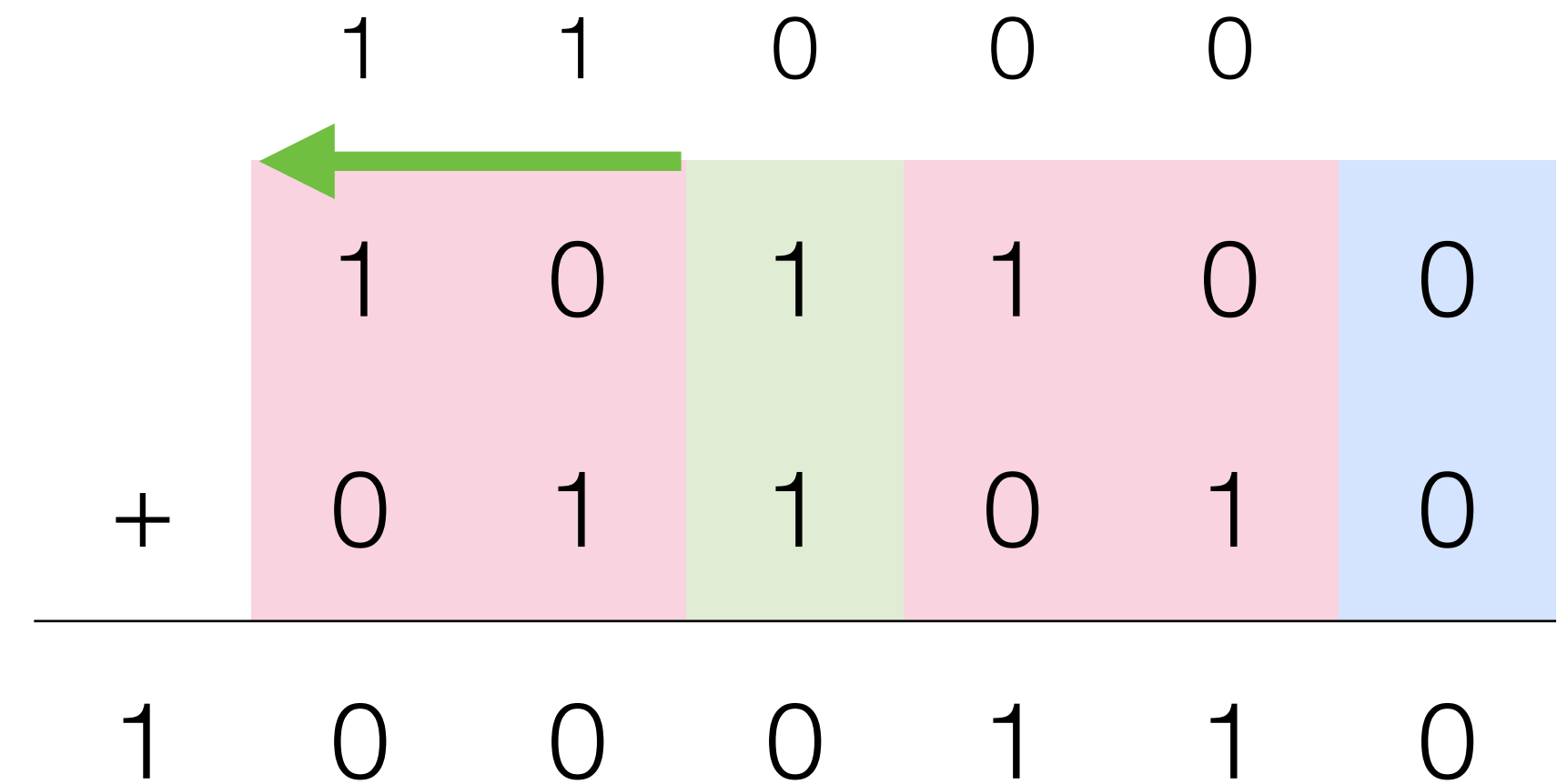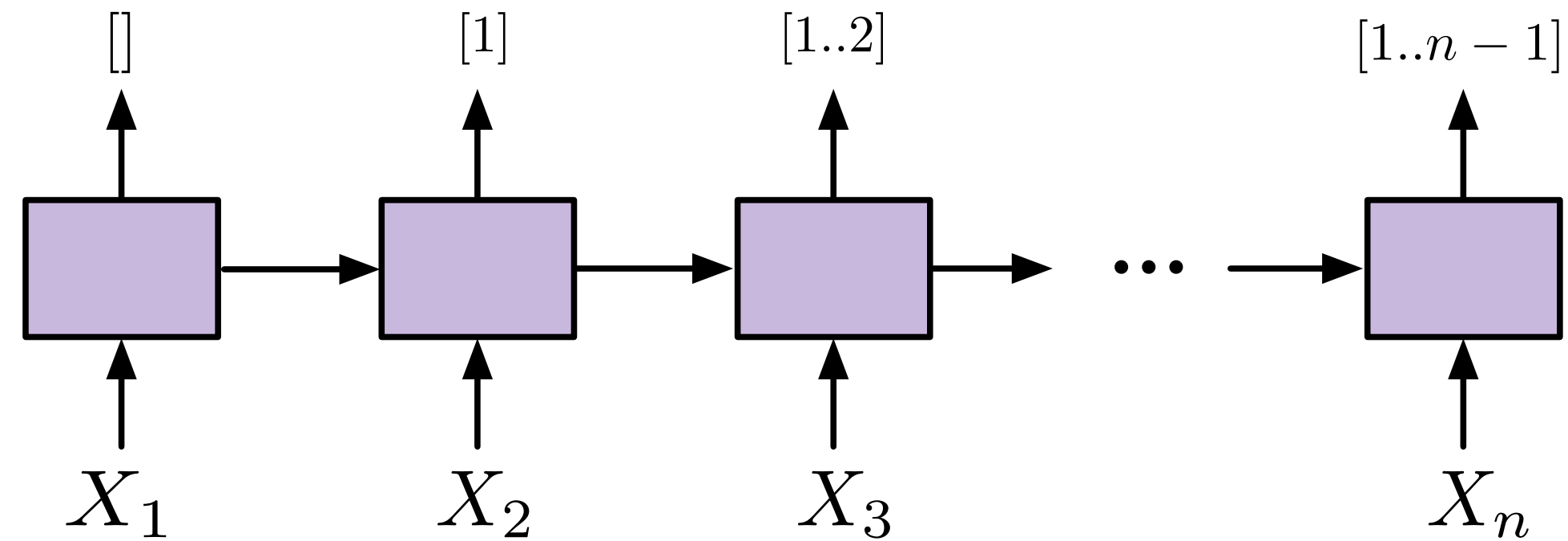# An example: asynchronous binary addition

```
    1   1   0   0   0

    1   0   1   1   0   0

+   0   1   1   0   1   0
_____

1   0   0   0   1   1   0
```

6	5	4	3	2	1	
p	p	g	p	p	k	-
g	**g**	**k**	**k**	**k**	**p**	
[1..5]	[1..4]	[1..3]	[1..2]	[1]	[]	

- We need to compute the *cumulative effect* of the k-p-g encoding
  - ❖ Combine codes to: [], [1], [1..2], [1..3], [1..4], etc.
  - ❖ "Prefix computation"

Y_1 Y_2 $\cdots$ Y_n

kpg block of size n

X_1 X_2 $\cdots$ X_n

Yale ENGINEERING

AVLSI
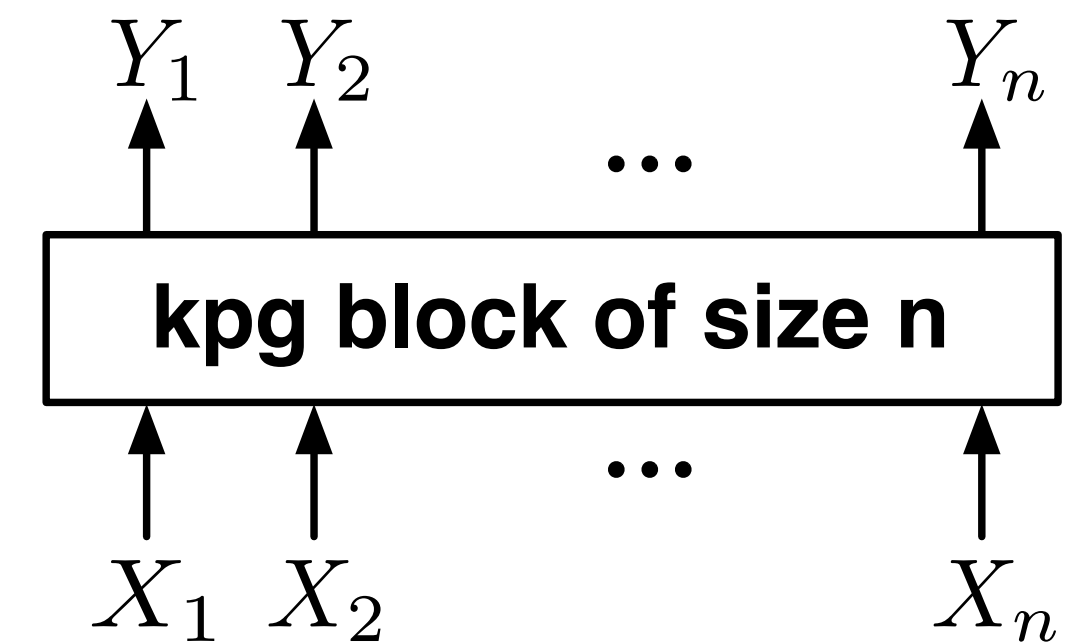
Simple topology for combining the codes

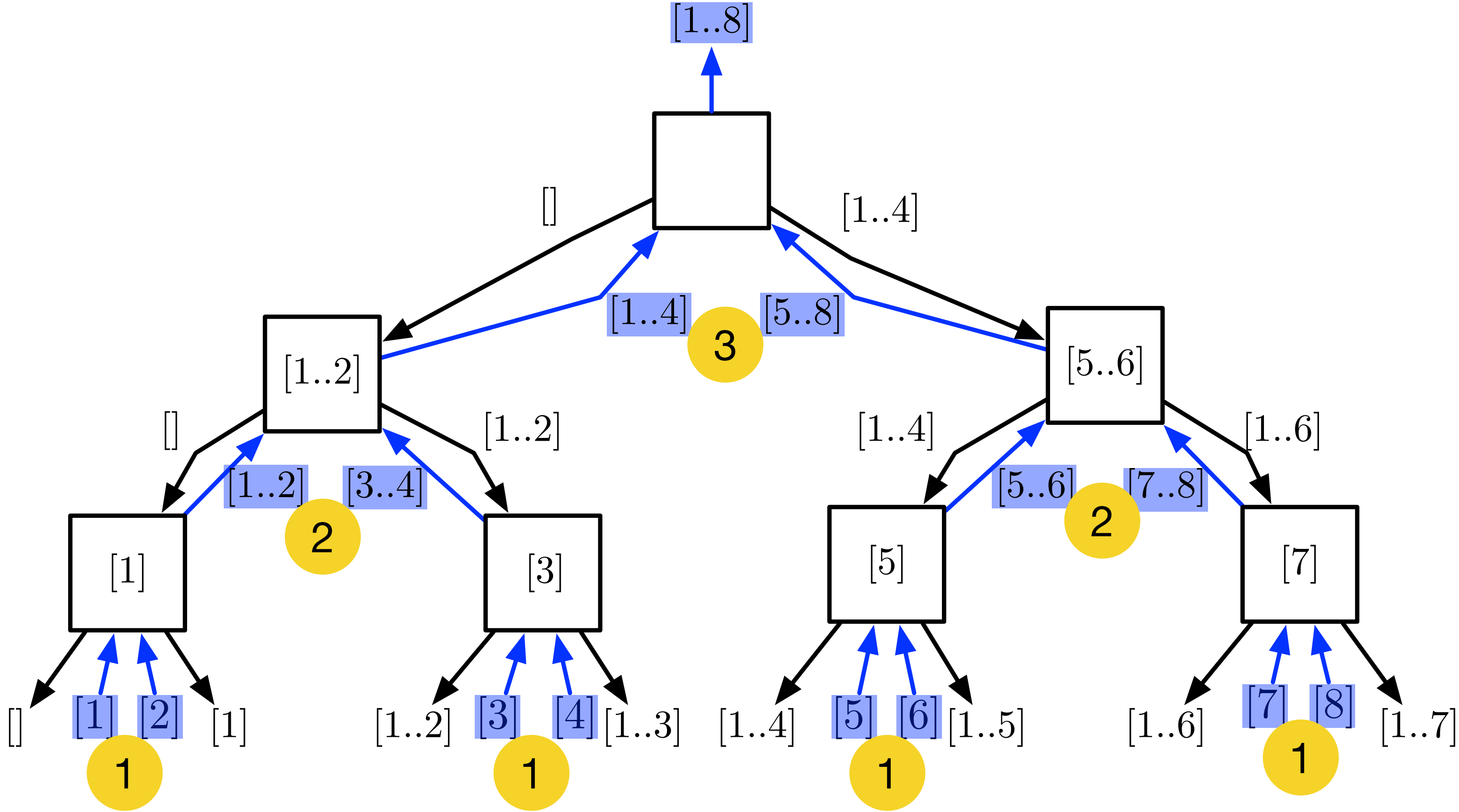$[\,]$ $[1]$ $[1..2]$ $[1..n-1]$

X_1 X_2 X_3 X_n
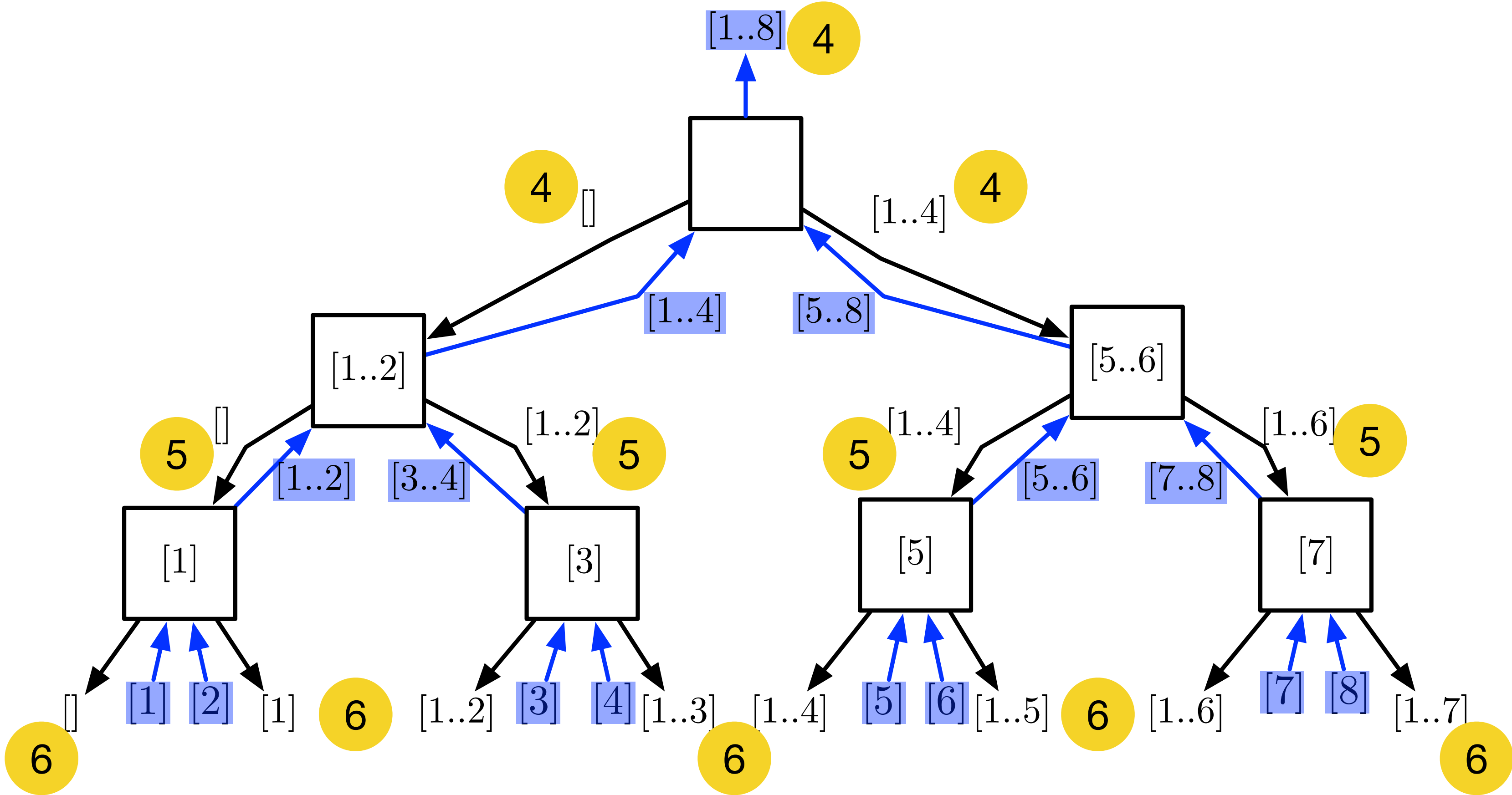
In some cases, we don't need to wait!

Theorem [von Neumann, 1946]. The average-case latency for a ripple carry binary adder is O(log N) for i.i.d. inputs

Theorem [Winograd, 1965]. The worst-case latency for binary addition is Ω(log N)
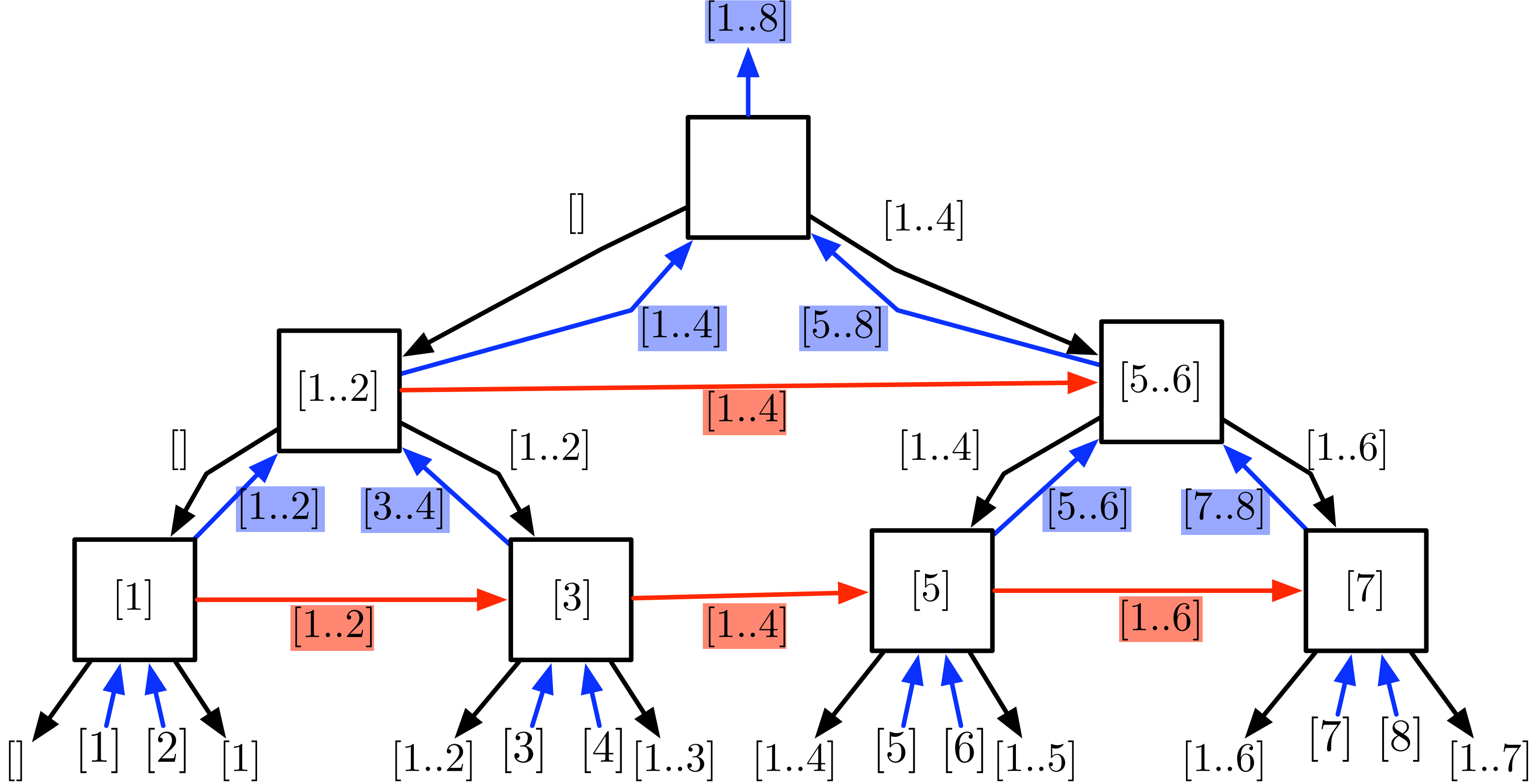
Yale ENGINEERING

AVLSI

Tree structure for computing carry information

Tree structure for computing carry information

# Compute the carries in two ways, pick the first one!

The average-case latency for this structure is O(log log N) for i.i.d. inputs!

Exploiting asynchrony

- Exploiting the gap between the average-case and worst-case
  - ❖ ... but you have to design the underlying *computation structure* (**algorithm**) to exploit it

- Power management
  - ❖ Dynamic switching activity only where computation is occurring

- Robustness
  - ❖ Circuit families that are timing / delay insensitive can be robust to process, voltage, and temperature changes

- Continuous-time operation
  - ❖ Bandwidth-adaptive signal processing in continuous time

- Mixed-signal electronics
  - ❖ Substrate noise can be less of an issue

Yale ENGINEERING
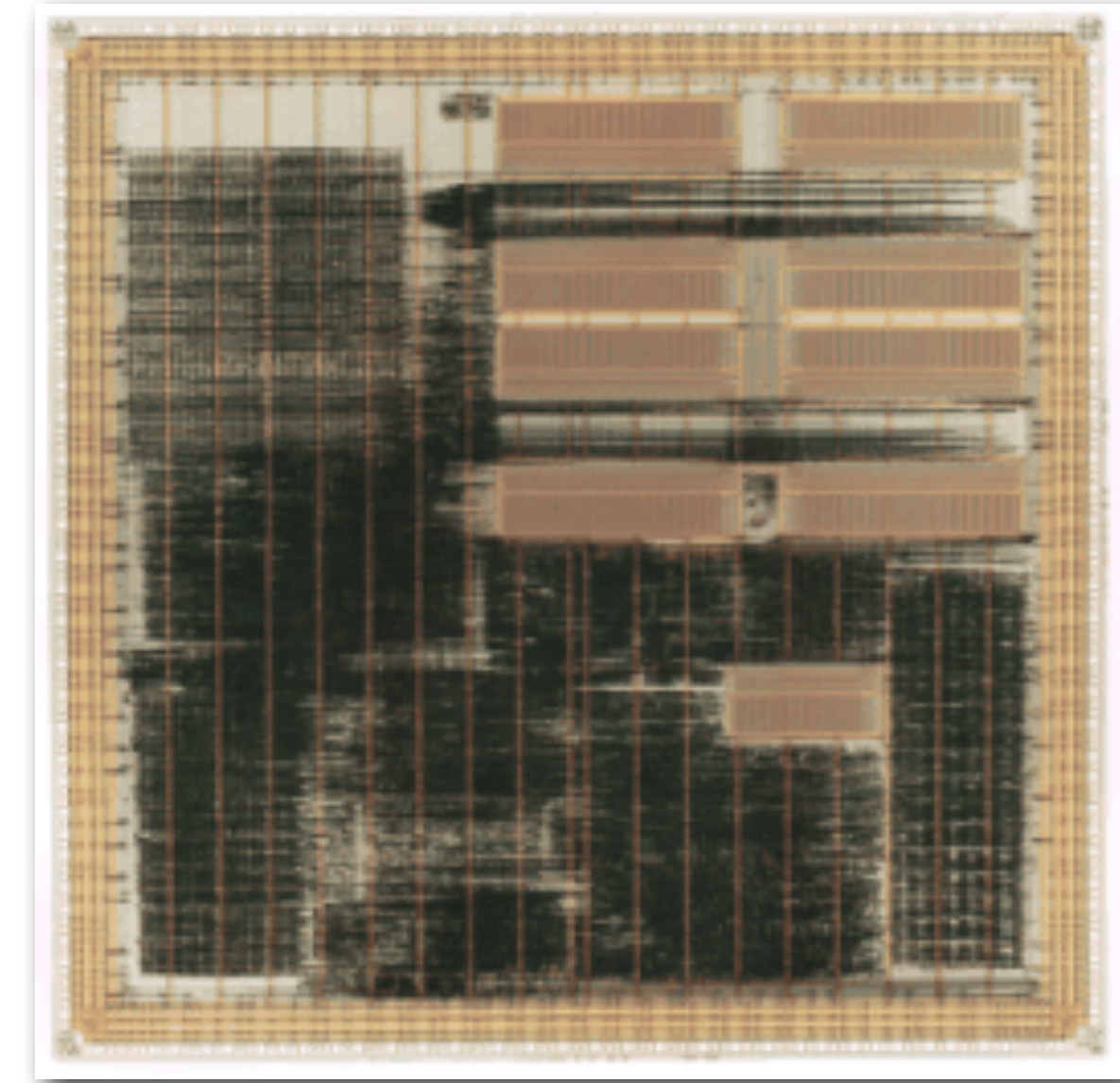
AVLSI

Example asynchronous microprocessors

The first asynchronous microprocessor
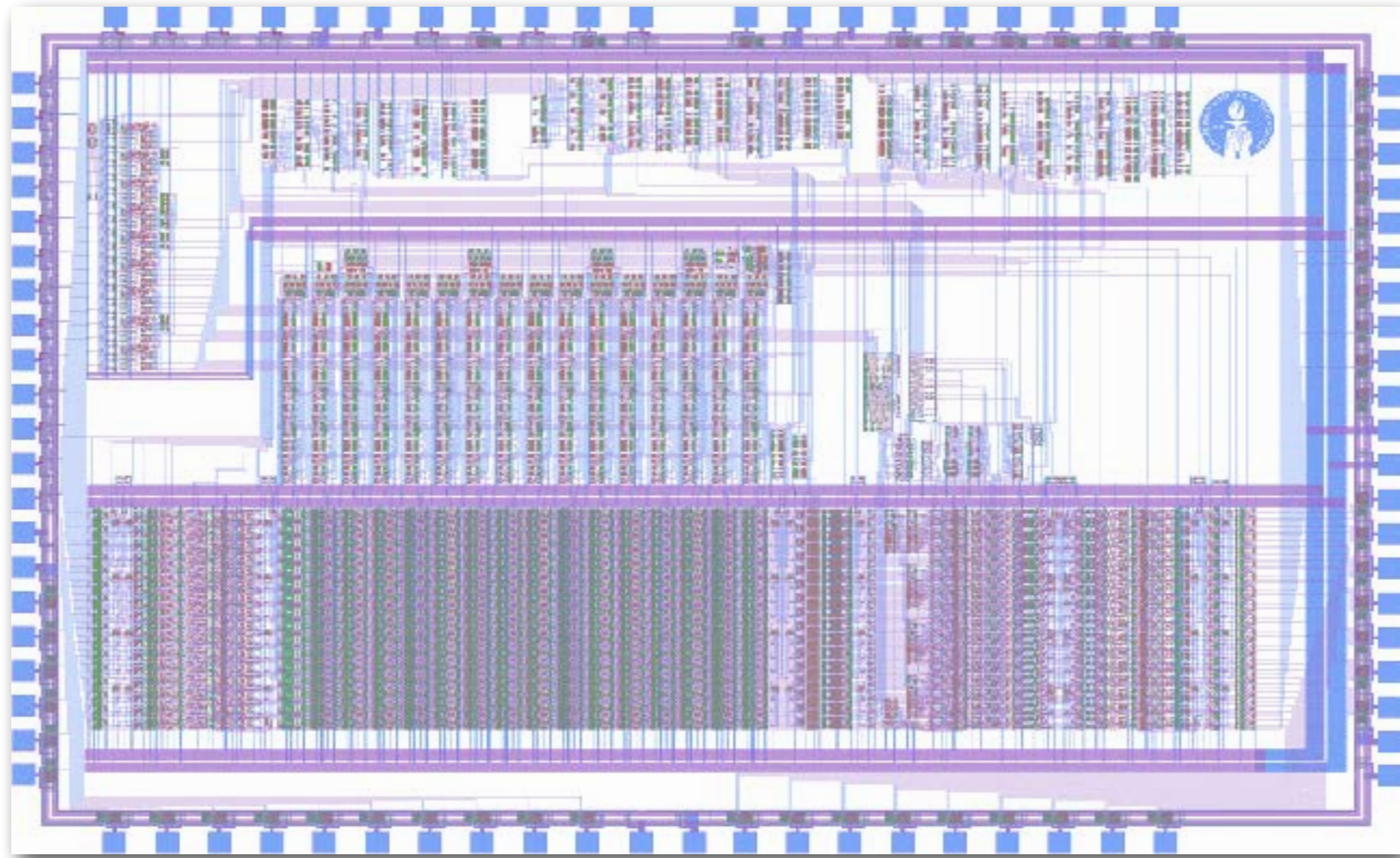
- ~20K transistors
- 16-20 MIPS (1.6µm feature size)
- 1989, Caltech

A. J. Martin, S. M. Burns, T.K. Lee, D. Borkovic, P.J. Hazewindus

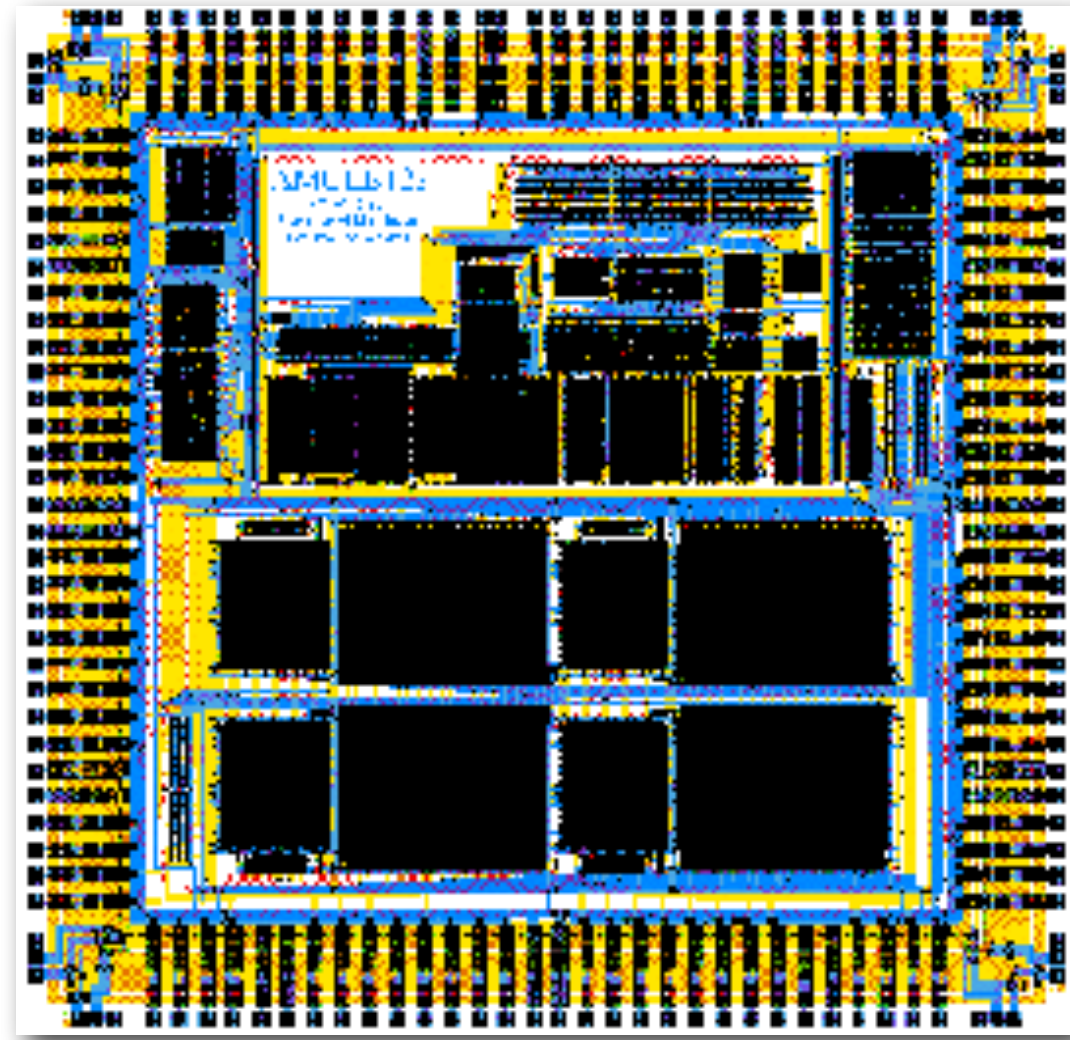A 32-bit MIPS R2000 microprocessor (TITAC-2)

- ~ 500K transistors
- 50 MIPS (0.5µm)
- U. of Tokyo (1996)

A. Takamura, M. Kuwako, M. Imai, T. Fujii, M. Ozawa, I. Fukasaku, Y. Ueno, T. Nanya

Yale ENGINEERING
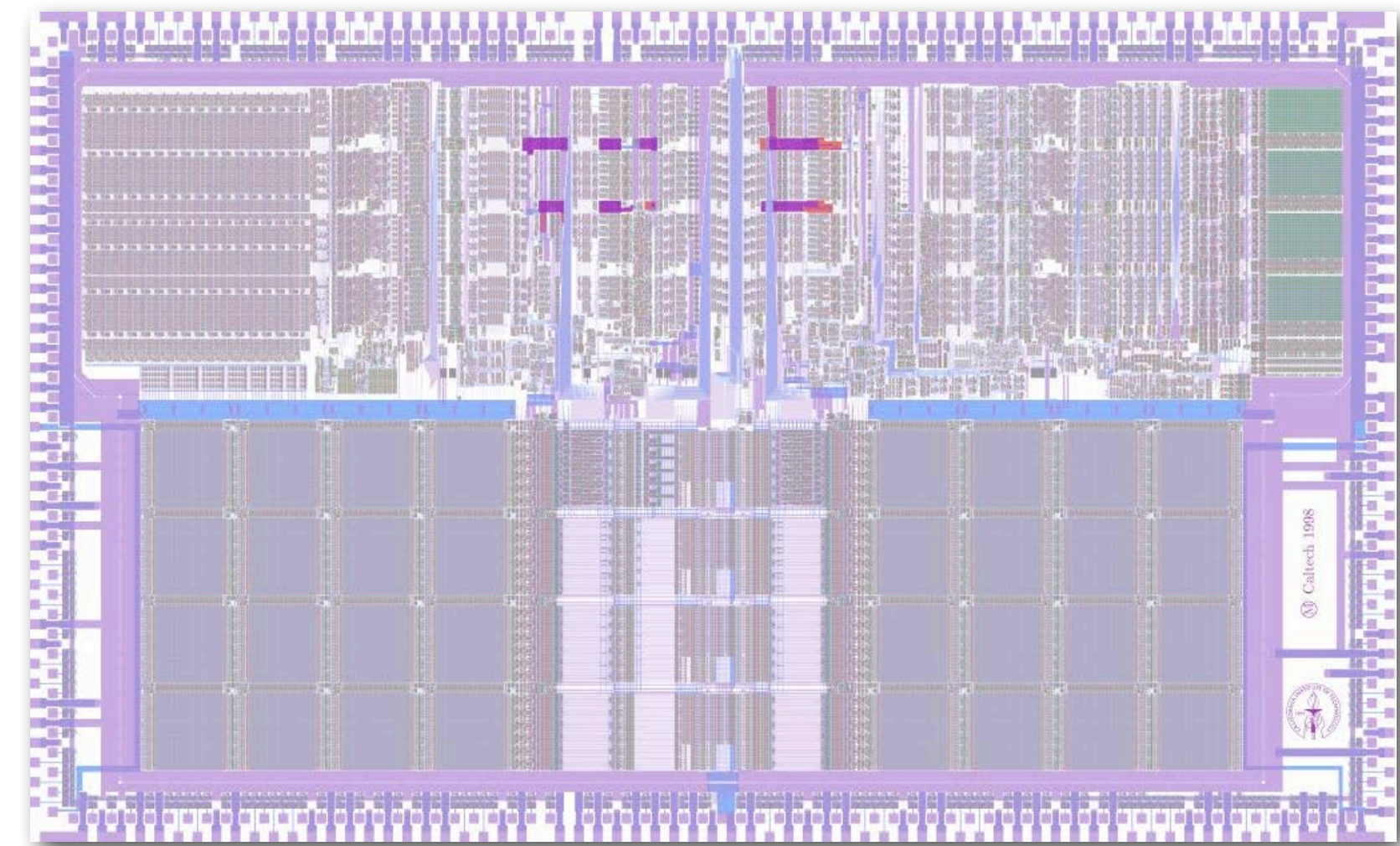
AVLSI

Example asynchronous microprocessors

Amulet2e (ARM microprocessor)

- ~ 454K transistors
- 32-bit microprocessor
- 27 MIPS (0.5µm)
- U. Manchester (1996)

S.B. Furber, J.D. Garside, S. Temple, J. Liu, P. Day, N.C. Paver
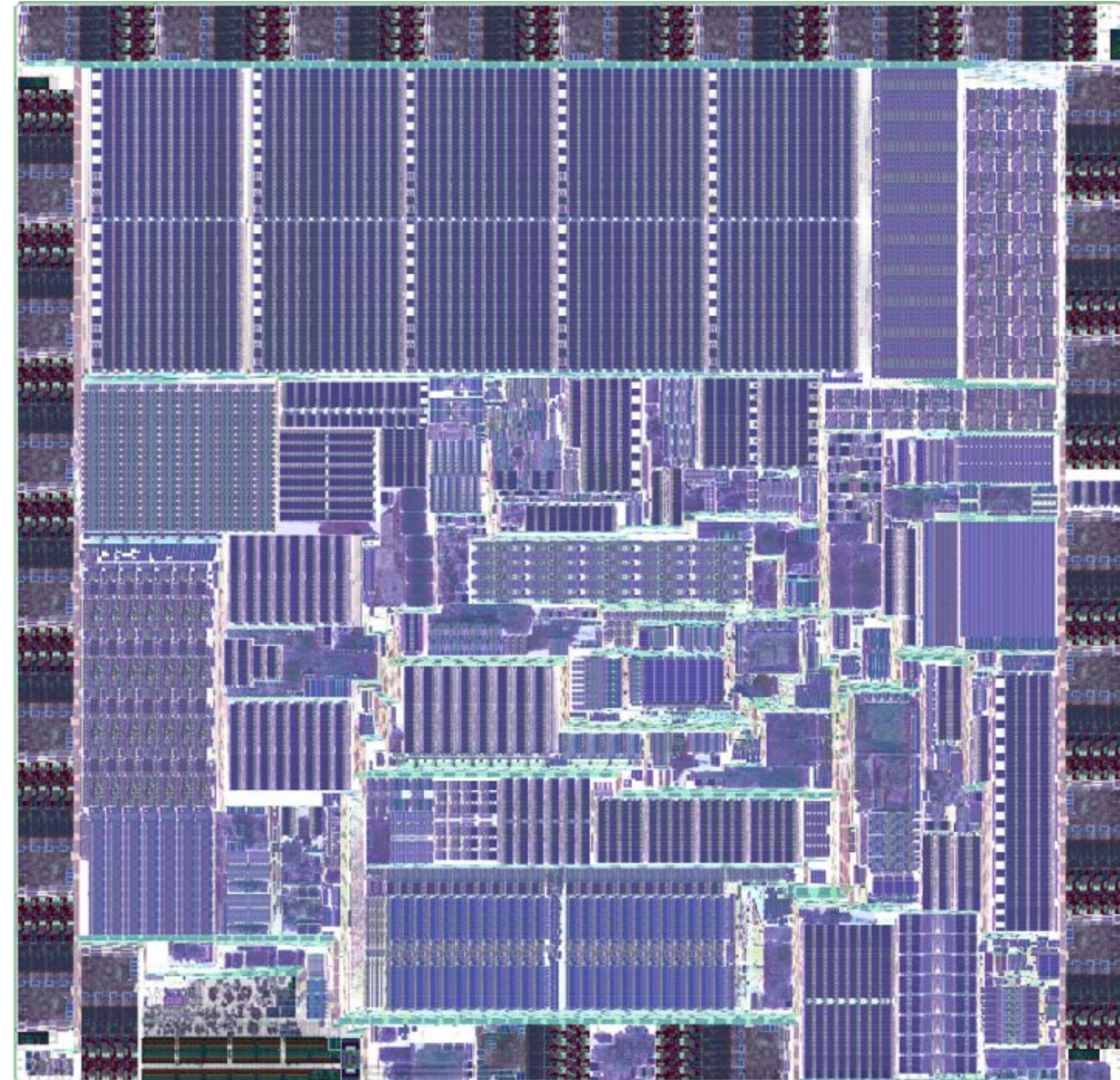
MIPS R3000 (MiniMIPS) microprocessor

- ~ 2.1M transistors
- 32-bit microprocessor
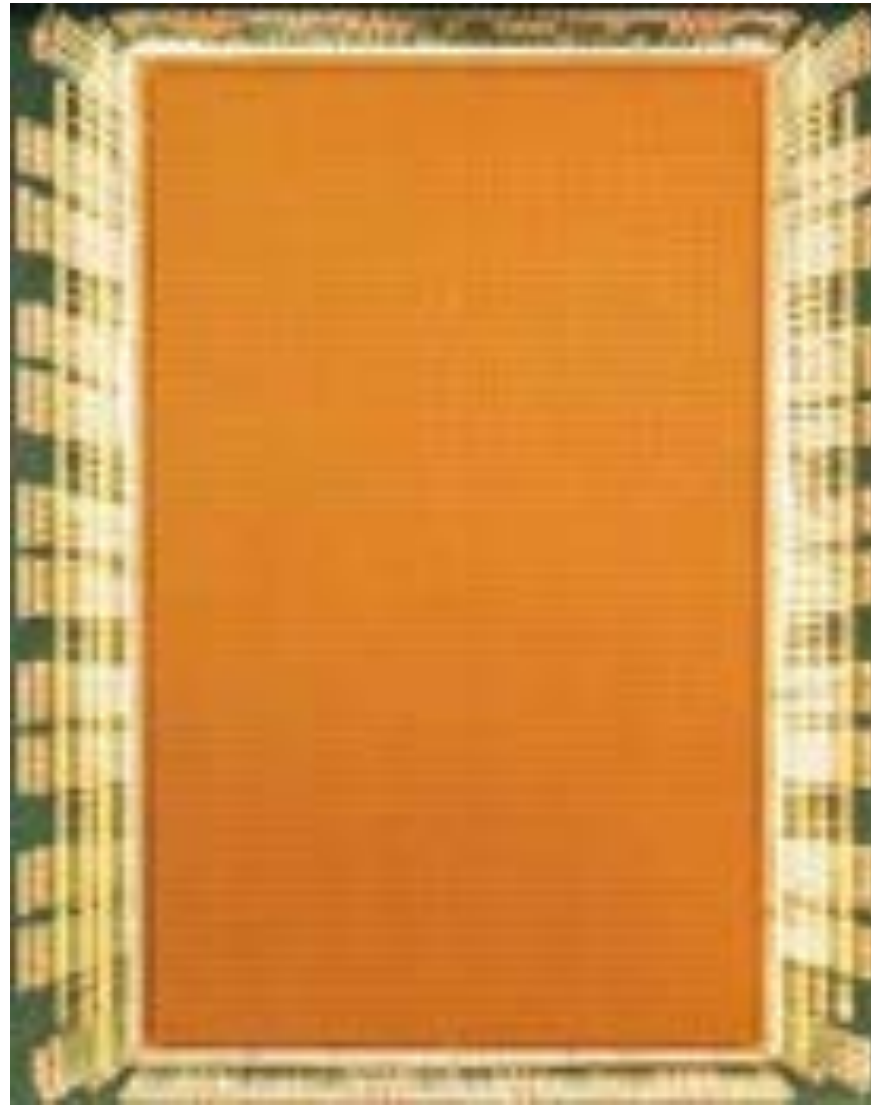- 250 MIPS (0.6µm) / 180 MIPS (RC)
- Caltech (1998)

A. J. Martin, A. Lines, R. Manohar, M. Nystrom, P. Penzes, R. Southworth, U. Cummings, T.K. Lee

Yale ENGINEERING
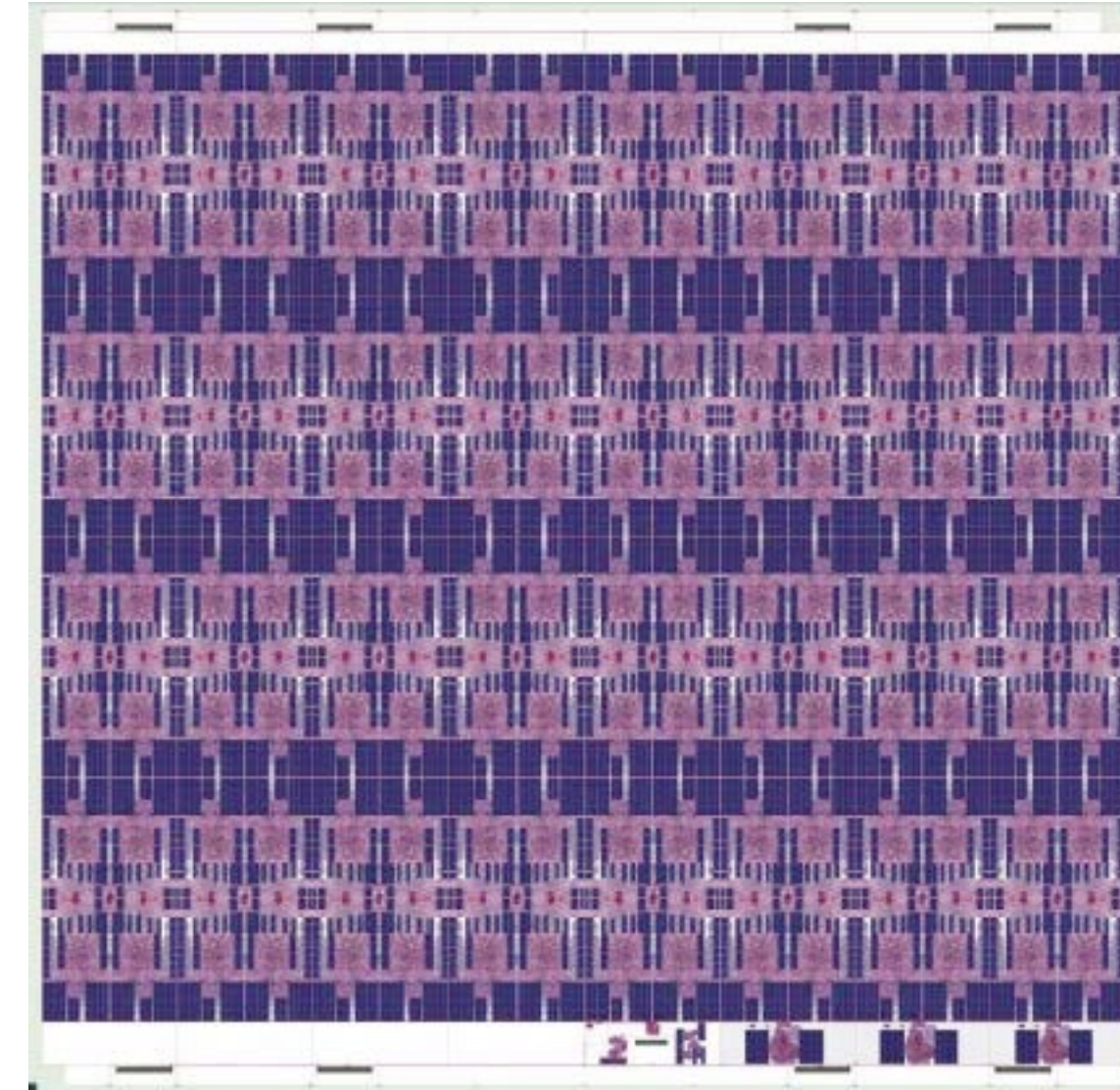
AVLSI

Commercial ethernet switch chips

- Ethernet switch chip (FM6000)

- 1.28Tbps switching

- NAT / VXLAN support

- 65nm technology

- Fulcrum (2012)

Neuromorphic chips

- "TrueNorth" chip
- Fully digital neuromorphic architecture
- 5.4B transistors
- 28nm technology
- IBM/Cornell (2014)

- "Loihi" neuromorphic chip
- Fully digital neuromorphic architecture
- 14nm technology
- Intel (2018)

Yale ENGINEERING

AVLSI