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: summer2026

Mattermost self-signup link

- June 1-2, 9:00AM to 4:30PM CEST
  - Behavioral design
  - Gate-level design
  - Circuit and physical design

- Slides, videos: `https://avlsi.csl.yale.edu/act/doku.php?id=summer2026:start`

June 1 schedule

9:00 AM	Welcome	
9:10 AM	Overview + introduction to asynchronous design	Rajit Manohar
9:30 AM	Message-passing behavioral description	Rajit Manohar
10:45 AM	Coffee break	
11:00 AM	Dataflow models	Jens Sparsø
12:10 PM	Lunch break	
1:10 PM	Handshake protocols	Rajit Manohar
1:30 PM	Gate level models	Rajit Manohar
1:45 PM	From dataflow to gates	Montek Singh
2:45 PM	From CHP to gates	Rajit Manohar
3:00 PM	Coffee break	
3:30 PM	CHP to gates wrap-up; non-determinism	Rajit Manohar

June 2 schedule

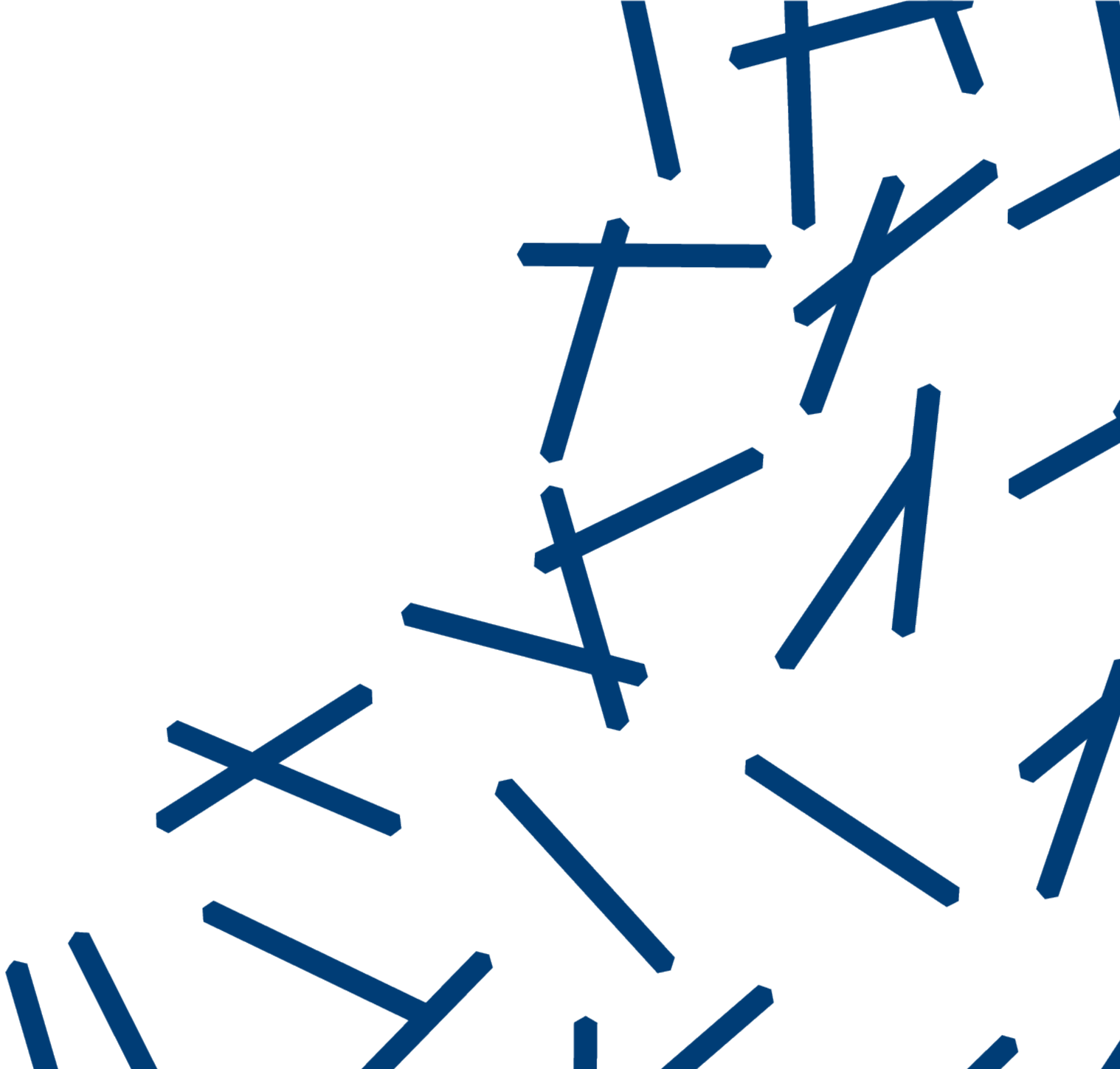
9:00 AM	Timing constraints	Rajit Manohar
9:30 AM	ASIC implementation flow I	Ole Richter
10:45 AM	Coffee break	
11:00 AM	ASIC implementation flow II	Rajit Manohar
12:10 PM	Lunch break	
1:10 PM	Petri net primer	Alex Yakovlev
2:00 PM	Controller design using Petri nets	Alex Yakovlev
3:00 PM	Coffee break	
3:30 PM	Custom circuit implementation	Rajit Manohar

Yale ENGINEERING

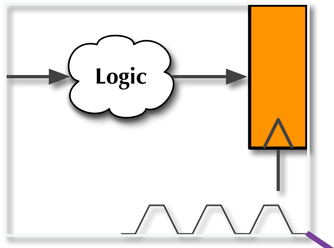
Introduction to asynchronous design

Rajit Manohar

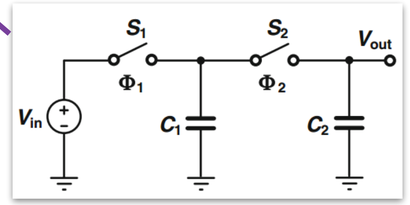
AVLSI

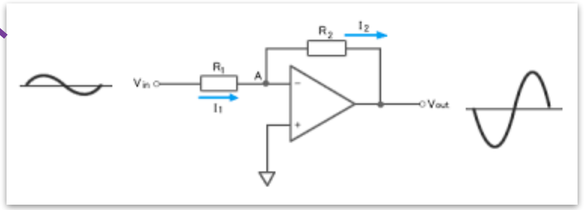


Approaches to computation



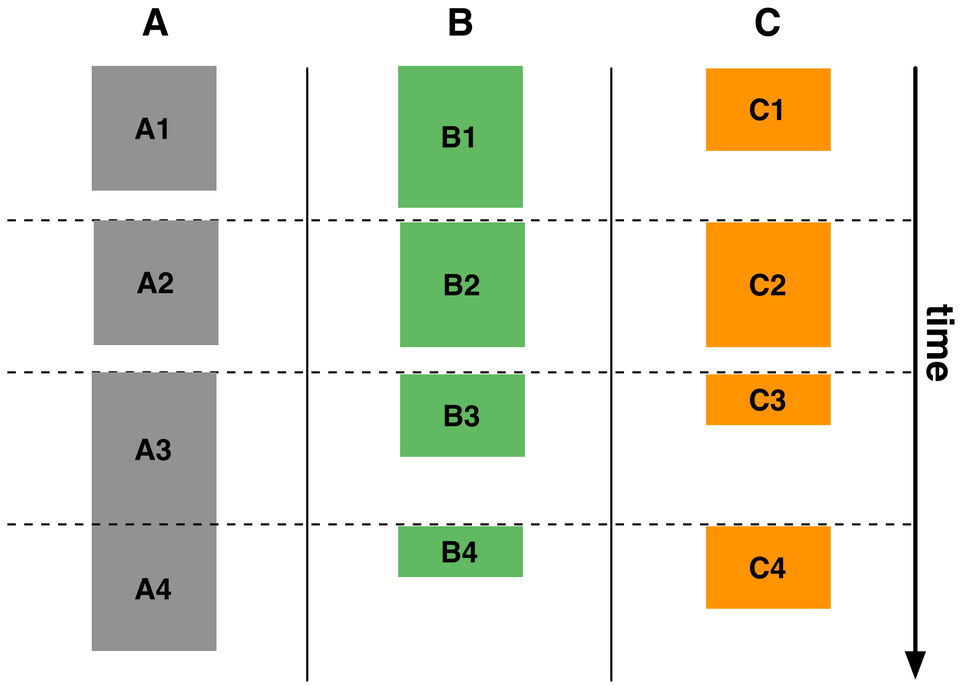
	Discrete Time	Continuous Time
Discrete Value	Digital, synchronous logic	Digital, asynchronous logic
Continuous Value	Switched-capacitor analog	General analog computation



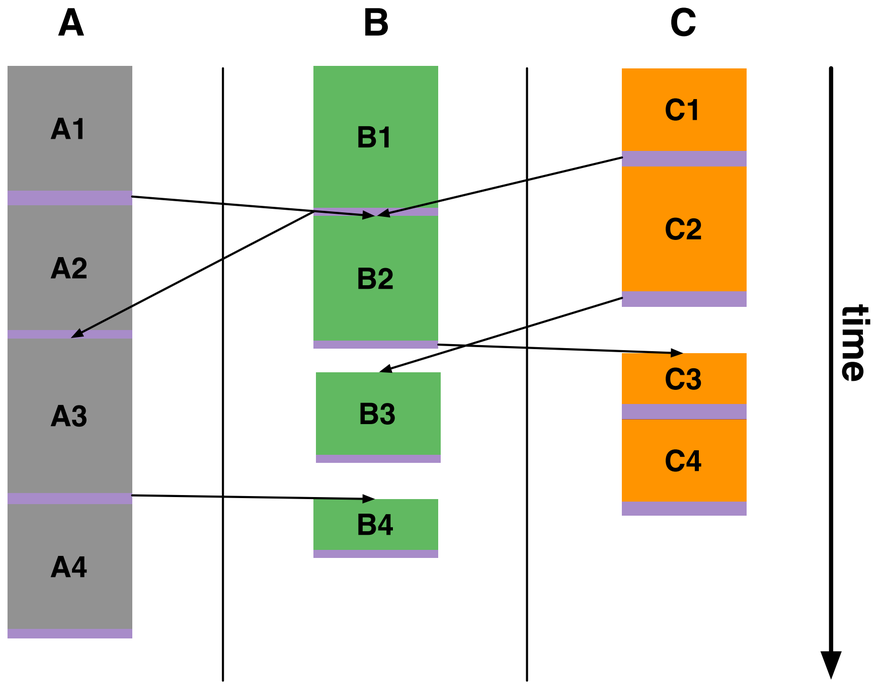


- Event-driven at the (digital) circuit level of abstraction
  - “Data-driven computing”
  - “Self-timed computing”
  - “Asynchronous computing”
  - “Clockless computing”

Discrete versus continuous time

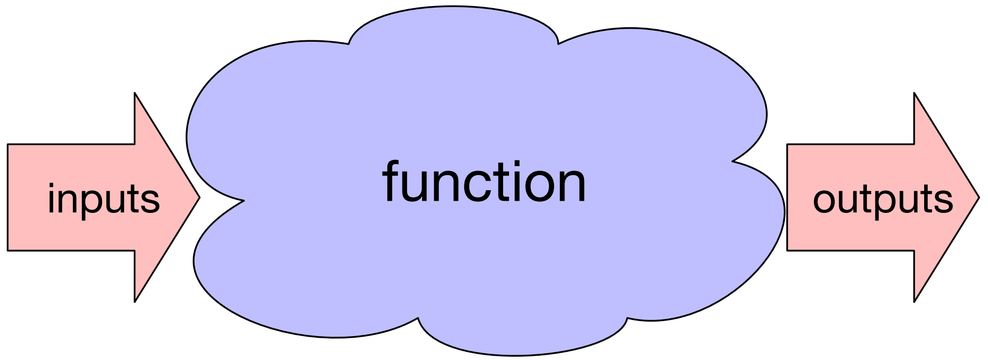


Discrete-time computation



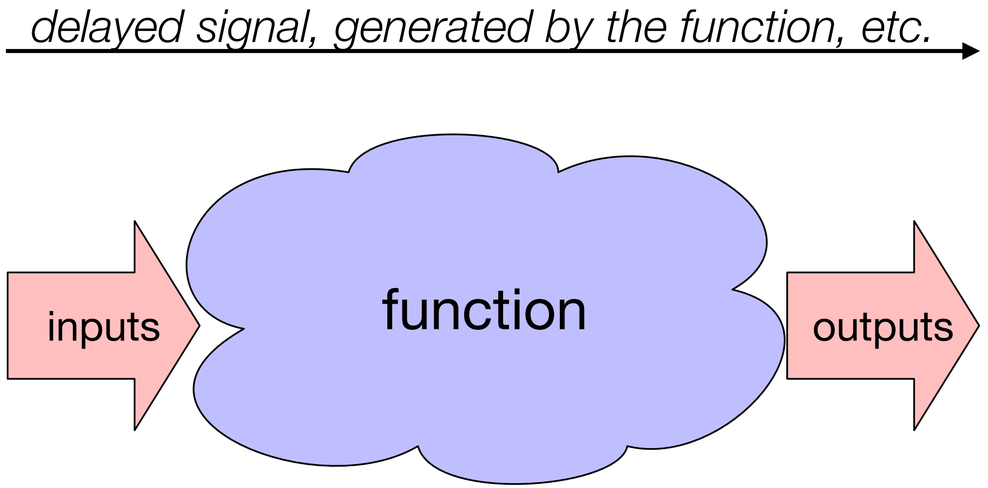
Continuous-time computation

Computing asynchronously



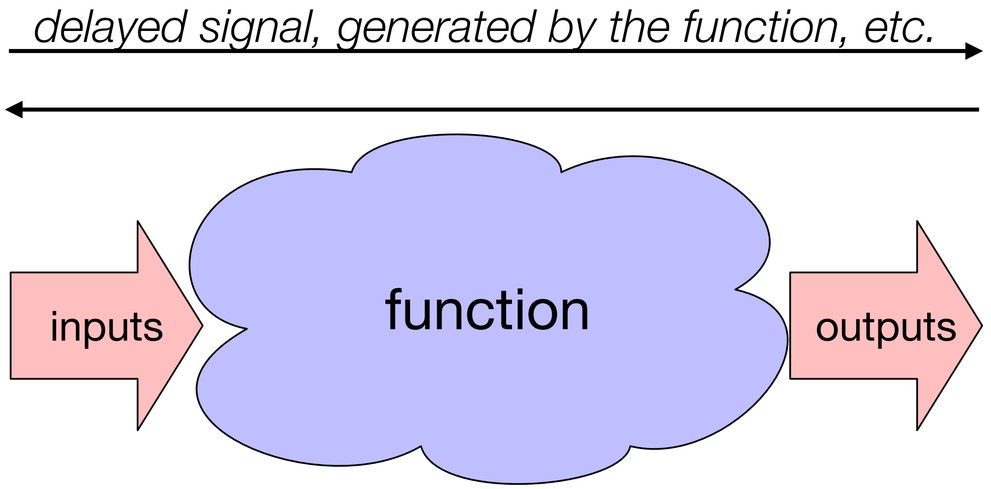
- When the input arrives, the function is triggered
- Eventually the output is produced

Computing asynchronously



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

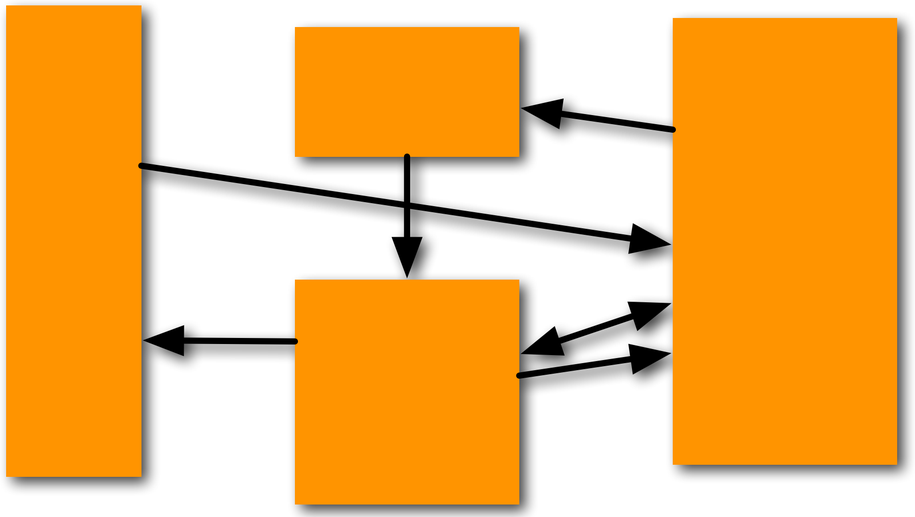
Computing asynchronously

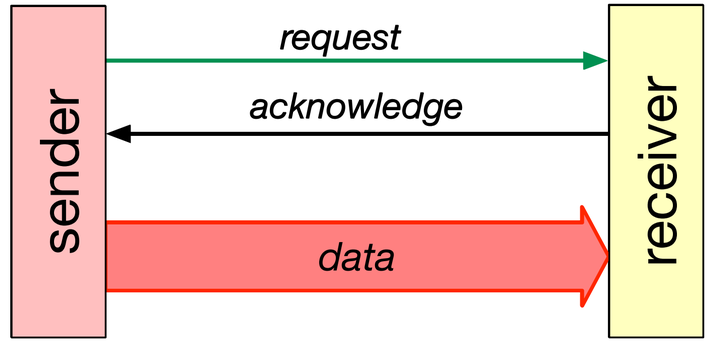


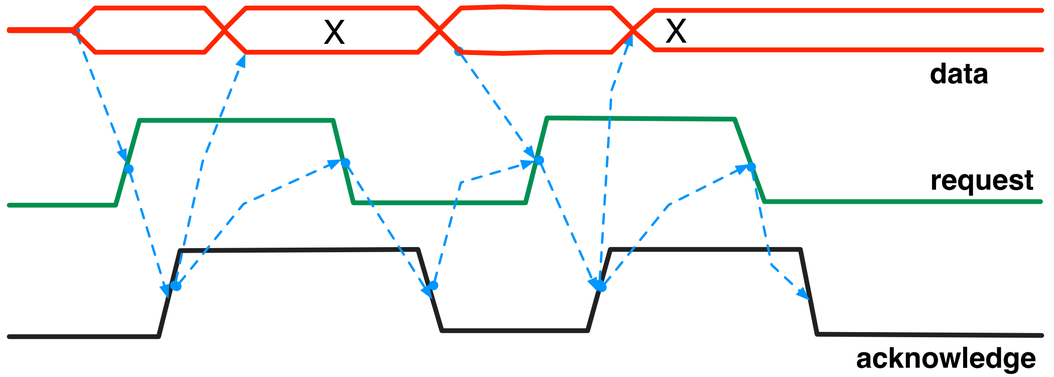
- When the input arrives, the computation is triggered
- Eventually the output is produced
  - We need to know **when** the output is ready
  - We need to know **when** we can produce the next output

Basic model

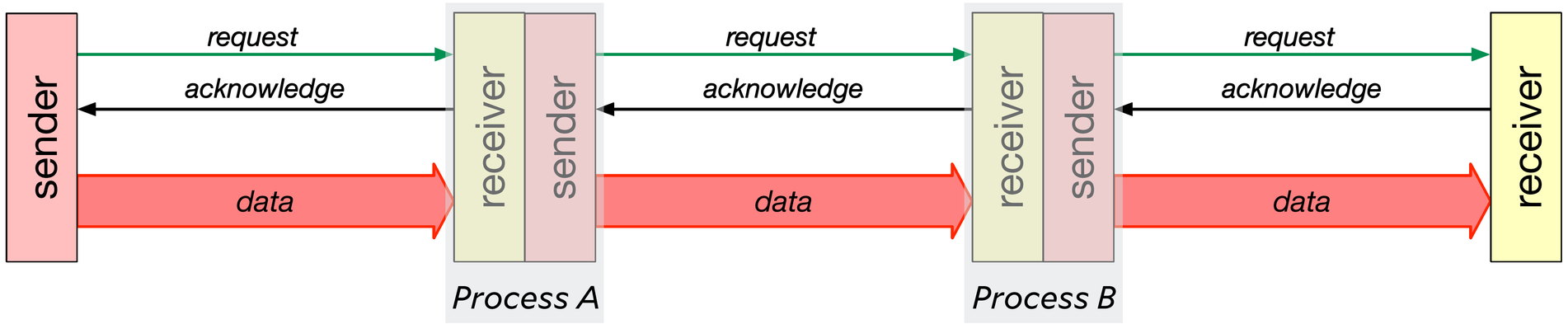
- Chip is a parallel program
  - ❖ Components: processes
  - ❖ Communication: via message-passing channels
  - ❖ Explicit synchronization through communication





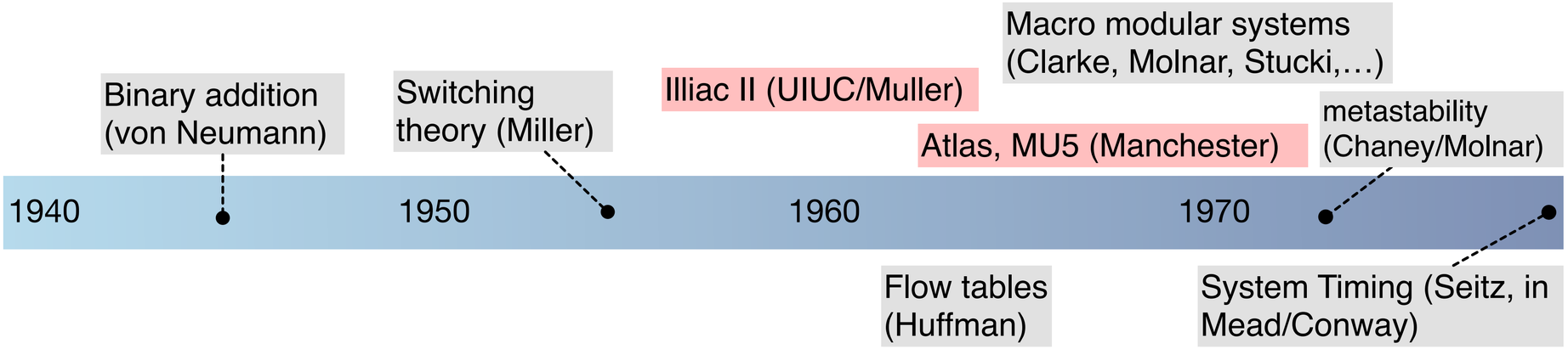


A compute pipeline



- Data moves through the pipeline at its own pace

A little bit of (biased) history...



"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)

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

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

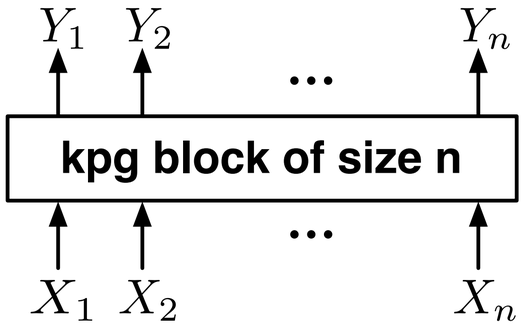
- Slow part: computing carry values
- Standard technique: classify input cases into three categories
  - kill
  - propagate
  - generate

An example: asynchronous binary addition

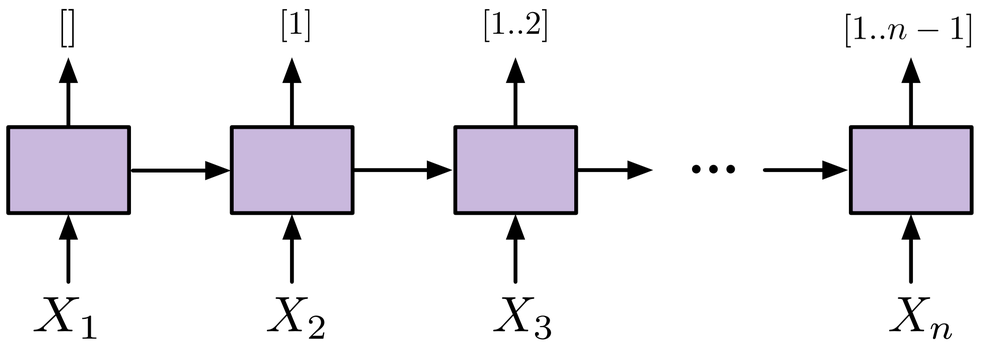
	1	1	0	0	0	
	1	0	1	1	0	0
+	0	1	1	0	1	0
	1	0	0	0	1	1	0

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"



Simple topology for combining the codes



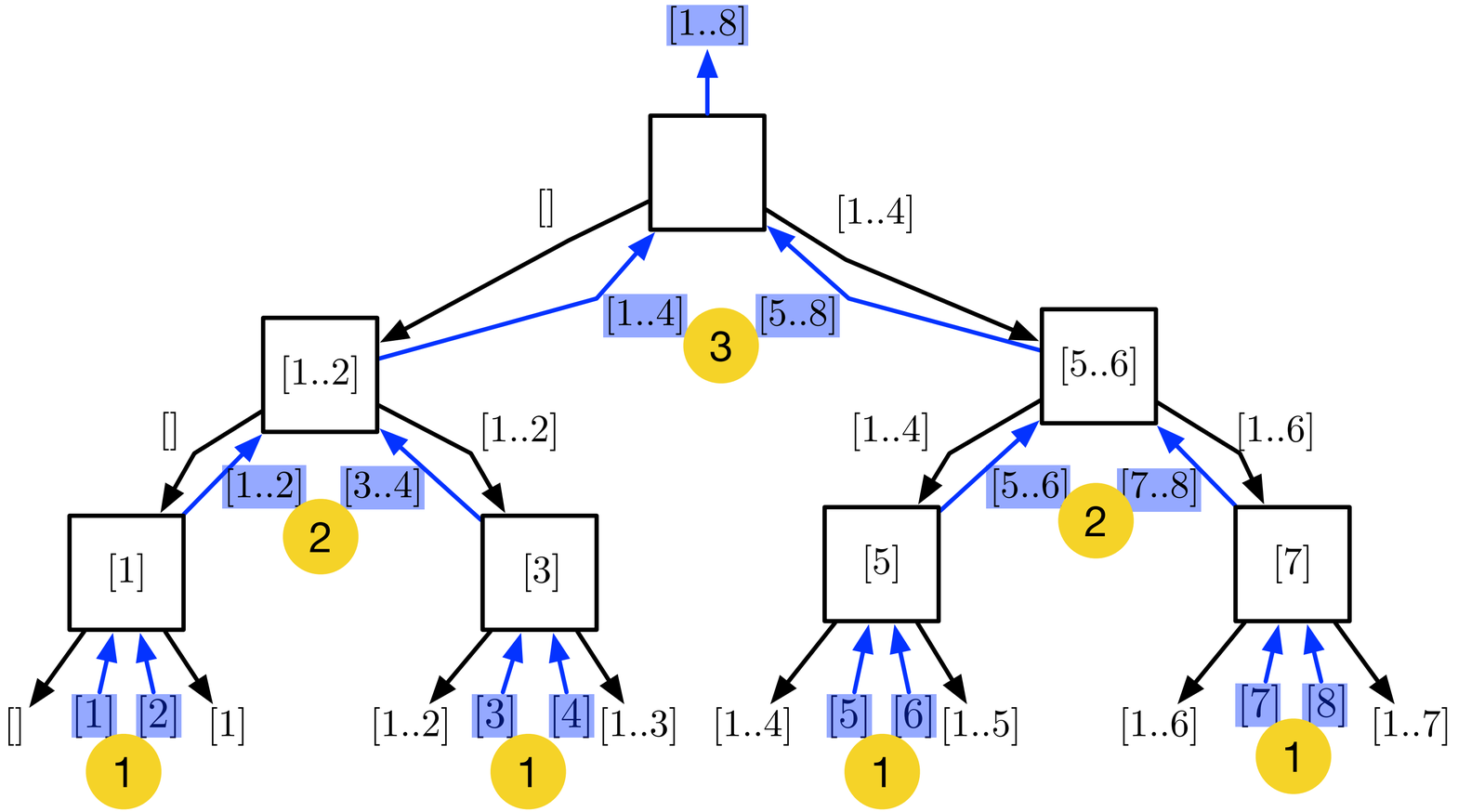
	1	1	0	0	0	
	1	0	1	1	0	0
+	0	1	1	0	1	0
1	0	0	0	1	1	0

In some cases, we don't need to wait!

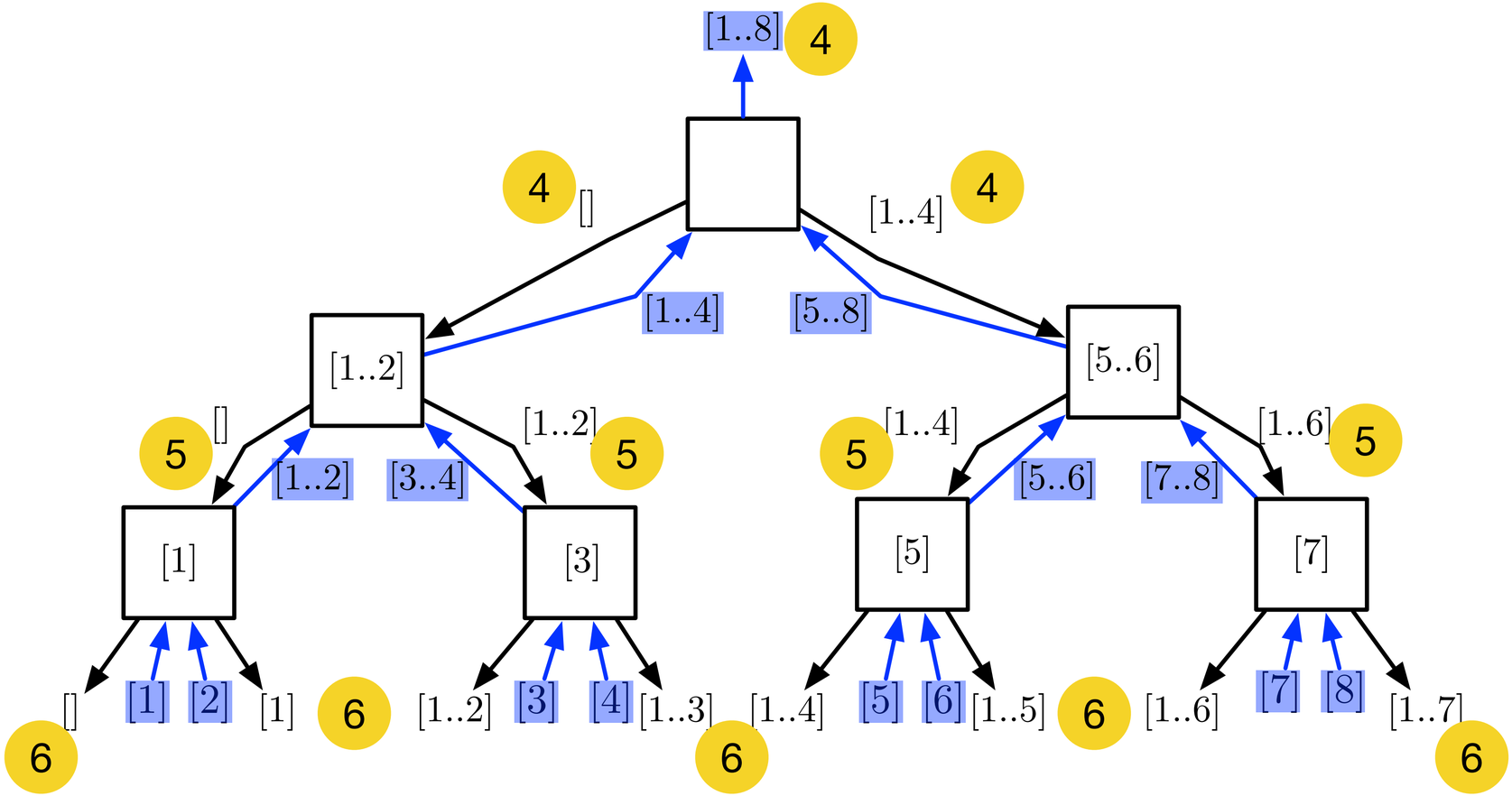
Theorem [von Neumann, 1946]. The average-case latency for a ripple carry binary adder is O(log N) for i.i.d. inputs

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

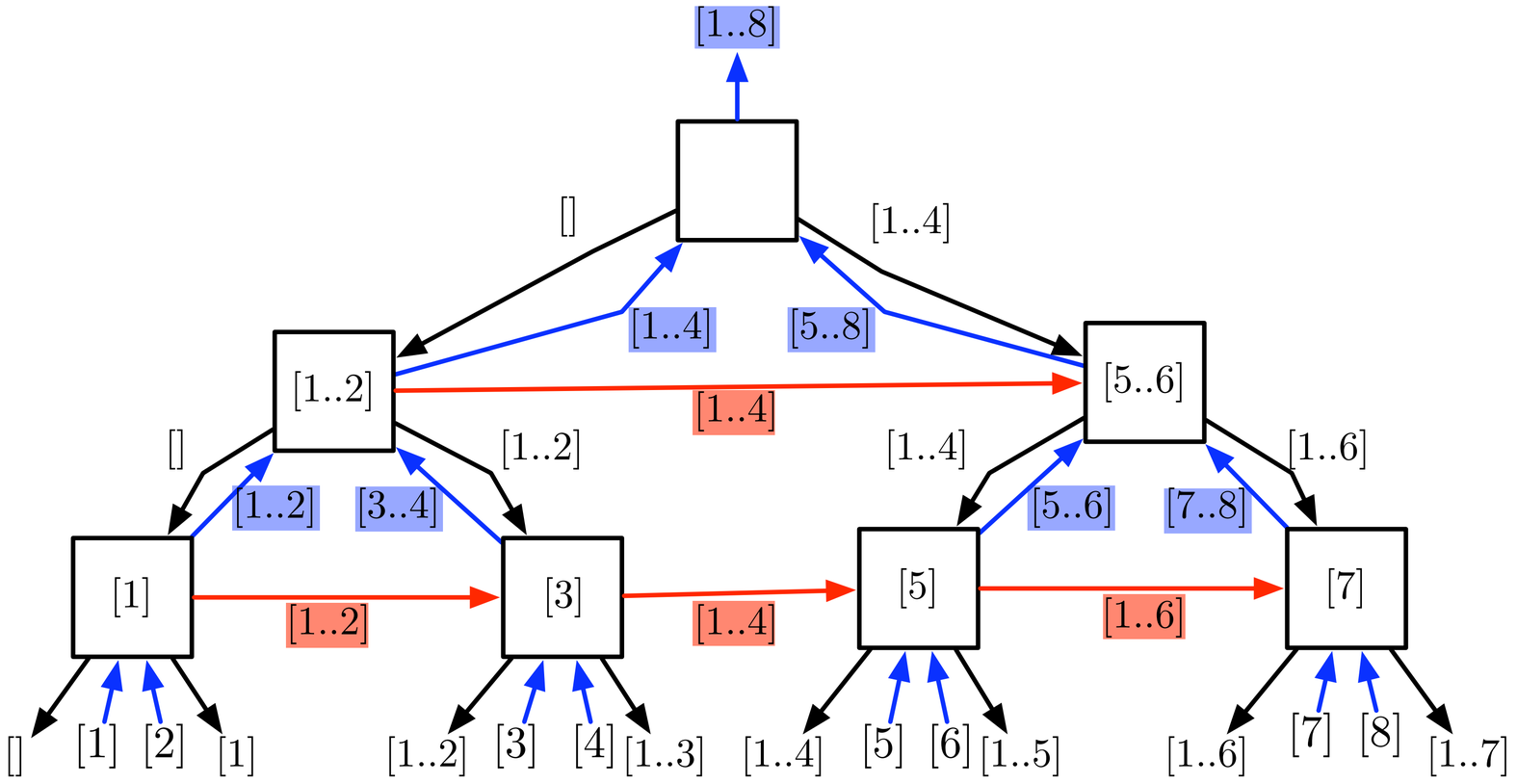
Tree structure for computing carry information



Tree structure for computing carry information



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



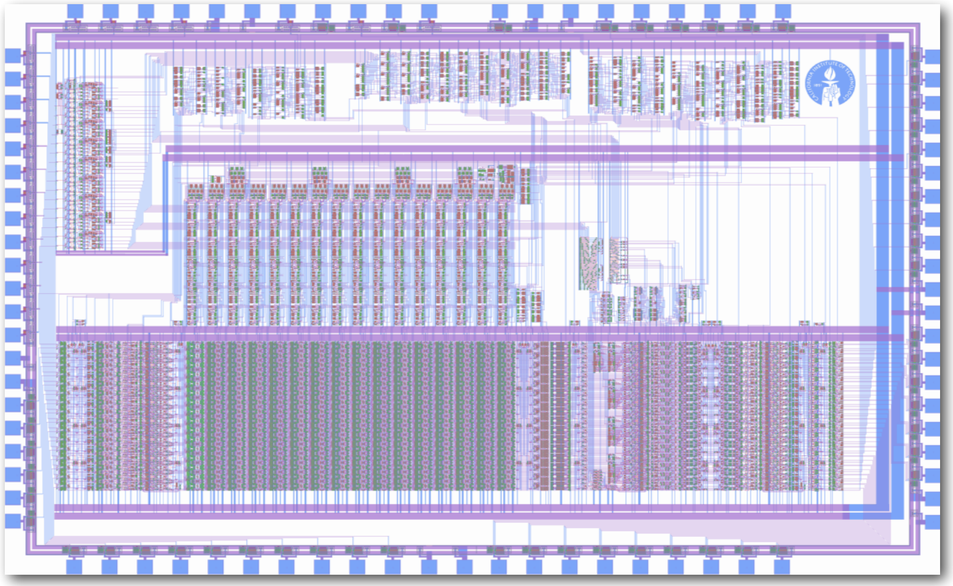
The average-case latency for this structure is O(log log N) for i.i.d. inputs!

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

Example asynchronous microprocessors

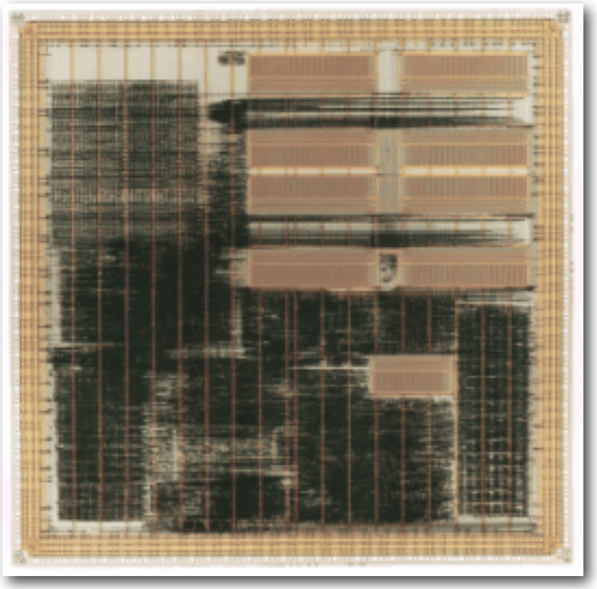
The first asynchronous microprocessor



- ~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)

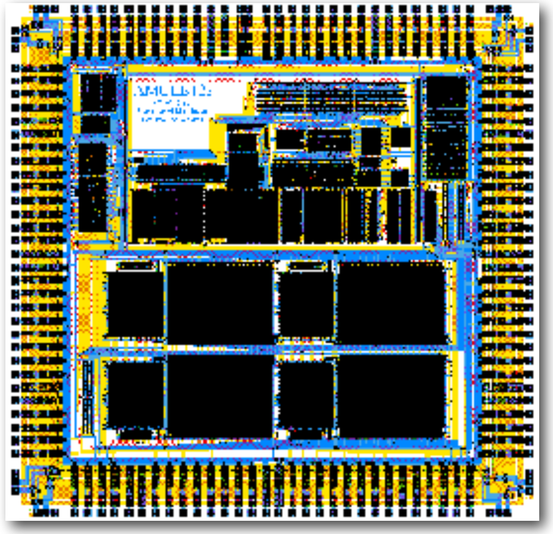


- ~ 500K transistors
- 50 MIPS (0.5µm)
- U. of Tokyo (1996)

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

Example asynchronous microprocessors

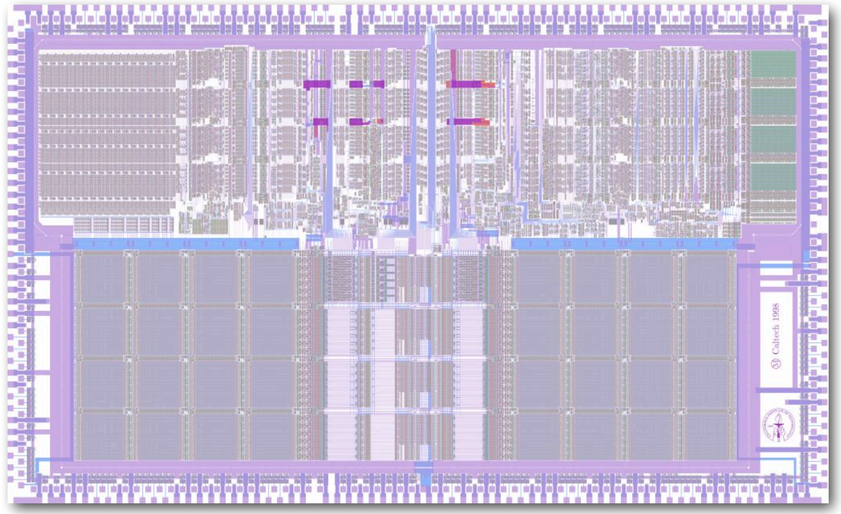
Amulet2e (ARM microprocessor)



- ~ 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

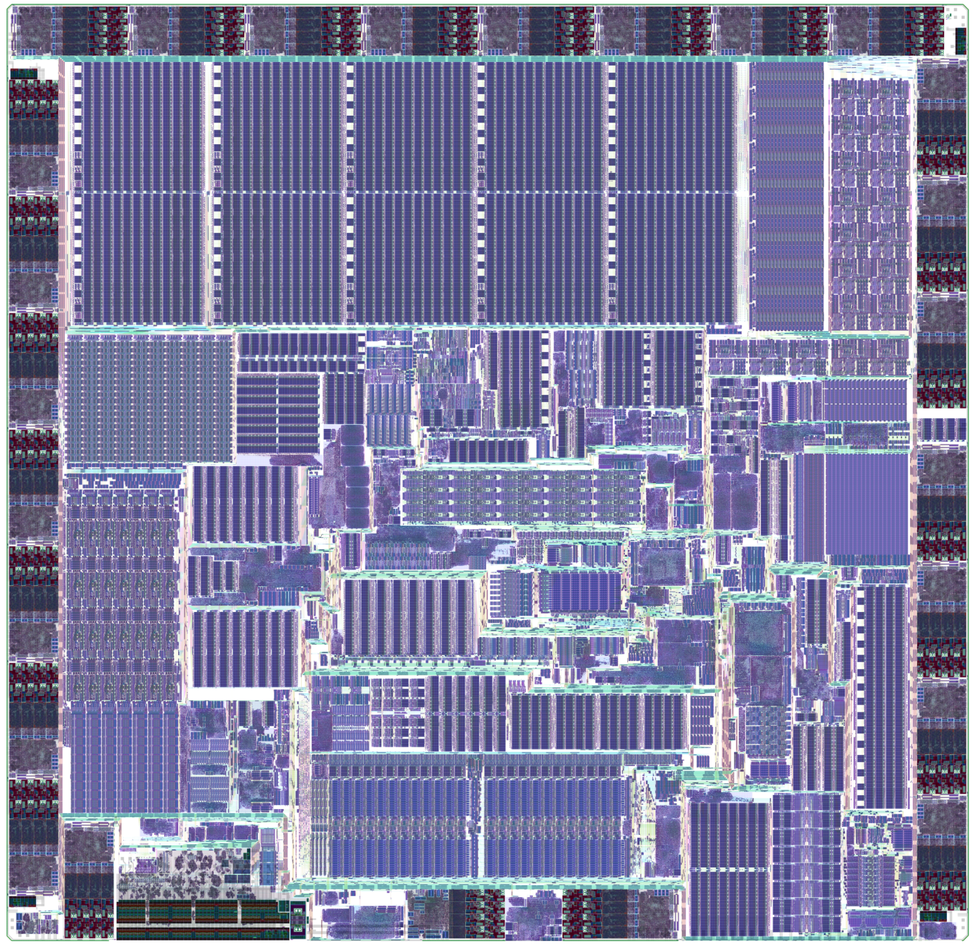


- ~ 2.1M transistors
- 32-bit microprocessor
- 250 MIPS (0.6µm) / 180 MIPS (RC)
- Caltech (1998)

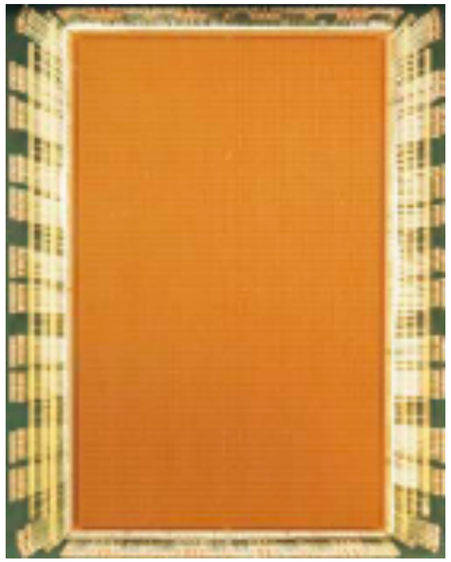
A. J. Martin, A. Lines, R. Manohar, M. Nystrom, P. Penzes, R. Southworth, U. Cummings, T.K. Lee

Commercial ethernet switch chips

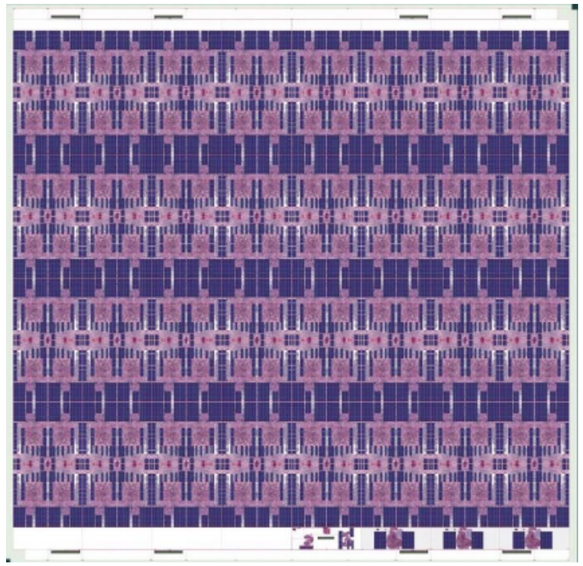
- Ethernet switch chip (FM6000)
- 1.28Tbps switching
- NAT / VXLAN support
- 65nm technology
- Fulcrum (2012)



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)