A Solder-Defined Computer Architecture for Backdoor and Malware Resistance

Examinee: Marc W. Abel

Committee: Travis Doom, Ph.D. (chair)
Jack Jean, Ph.D.
Michael Raymer, Ph.D.
Krishnaprasad Thirunarayan, Ph.D. (T.K. Prasad)
Vincent Schmidt, Ph.D. (Air Force Research Laboratory)

For slides: https://wakesecure.com

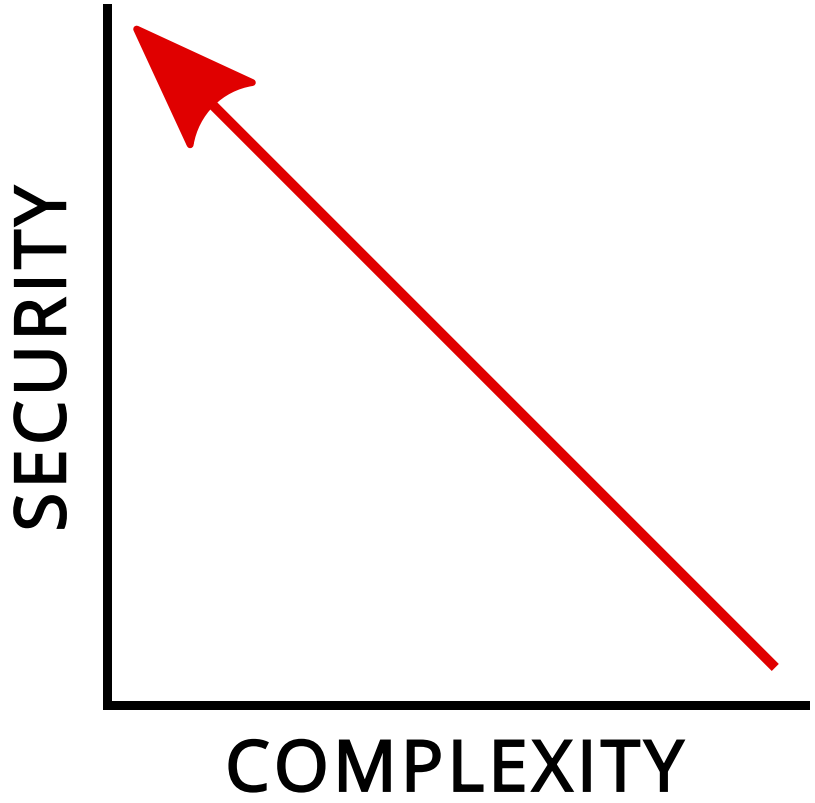
Department of Computer Science and Engineering

WRIGHT STATE UNIVERSITY

Today's conversation

1. Introduction
   Why computer hardware is a serious problem
2. Aspiration
   A logic family for solder-defined architectures
3. Combinational logic
   Arithmetic logic unit
4. Sequential logic
   Central processing unit
5. Closing
   Contributions, milestones, and questions

To increase security, reduce complexity



What's wrong with our hardware?

- Too complex
- Not well controlled by buyers
- Does not consider needs of software
- Not enough alternatives

Categories of vulnerability-inducing hardware irregularities

Category	I	II	III
Origin	purposeful	unexpected	malicious
Example	arithmetic wrap	RowHammer	hidden backdoor
Software fix?	yes	no	no
VLSI fix?	yes	yes	no
Manufacturing fix?	yes	yes	yes

Category I example: Integer wraparound

(Category I irregularities exist for a purpose.)

C programmers used to write:

```
c = a + b;
```

Today, they would need to write:

```
if (b > 0 && a > (INT_MAX - b) || b < 0 && a < (INT_MIN - b))
    longjmp(CATCHIT, SIGNED_ADD_OVERFLOW);
else
    c = a + b;
```

Some well-known Category II irregularities

(Category II irregularities are unplanned and unexpected.)

When	Architecture	Name	Synopsis
1985	80386	multiply bug	arithmetic error
1994	Pentium	FDIV	arithmetic error
1998	Pentium	F00F	lockup
2003	Via C3	God mode	privilege escalation
2008	Intel AMT	Silent Bob	full control of everything
2015	DRAM	RowHammer	memory corruption
2017	x86	Spectre	read others' memory
2017	x86, POWER, ARM	Meltdown	read all memory
2020	Intel SGX	load value inj.	inject data values
2020	Intel CSME	[M. Ermolov]	broken authentication

Actual and rumored Category III exploits

(Category III irregularities are intentionally malicious.)

Who	Architecture	Synopsis
AMD	Platform Security Processor	hypothesized backdoor
Apple	iPhone 6 + iOS 10.2.1	sabotaged performance
Deere	8520T tractor	right to repair infringements
Huawei	5G cellular infrastructure	potential for China influence
Intel	Management Engine	hypothesized backdoor
Intel	RDRAND instruction	non-randomness suspicions
NSA	ANT Catalog	implantable surveillance products
VIA	C3 (x86 clone)	backdoors claimed by C. Domas
ZTE	5G cellular infrastructure	potential for China influence

Proposed Category III countermeasures

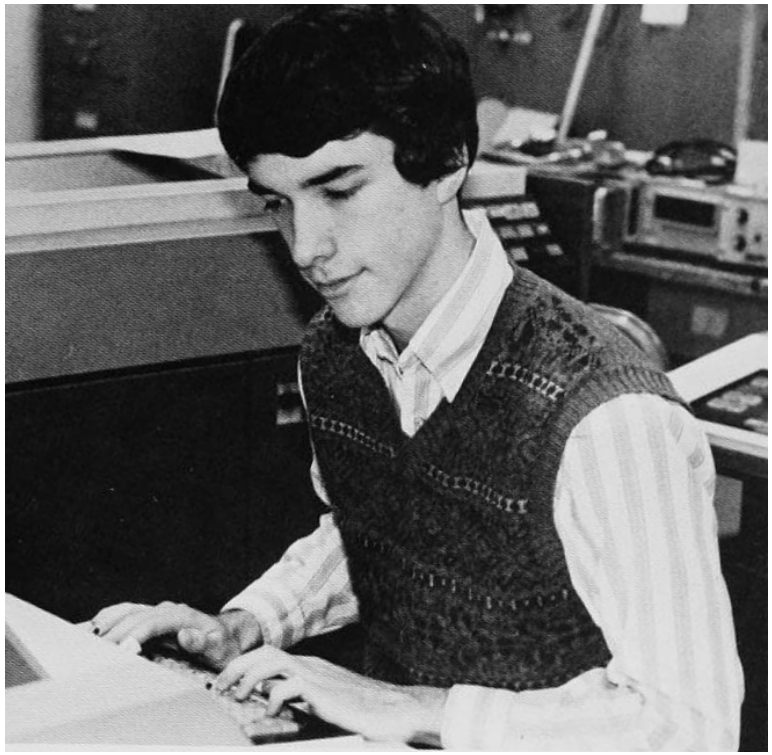
Proponent	Synopsis
Michael Pompeo	geopolitical controls
Adam Waksman	lock down VLSI supply chain
Eric Love	add formal proofs of security to hardware IP
Mirko Holler	X-ray ptychographic inspection

This proposal	
Marc Abel	complex logic to be built by end user

The work of this proposal targets all categories

Category	I	II	III
Origin	purposeful	unexpected	malicious
Example	arithmetic wrap	RowHammer	hidden backdoor
Software fix?	yes	no	no
VLSI fix?	yes	yes	no
Manufacturing fix?	yes	yes	yes

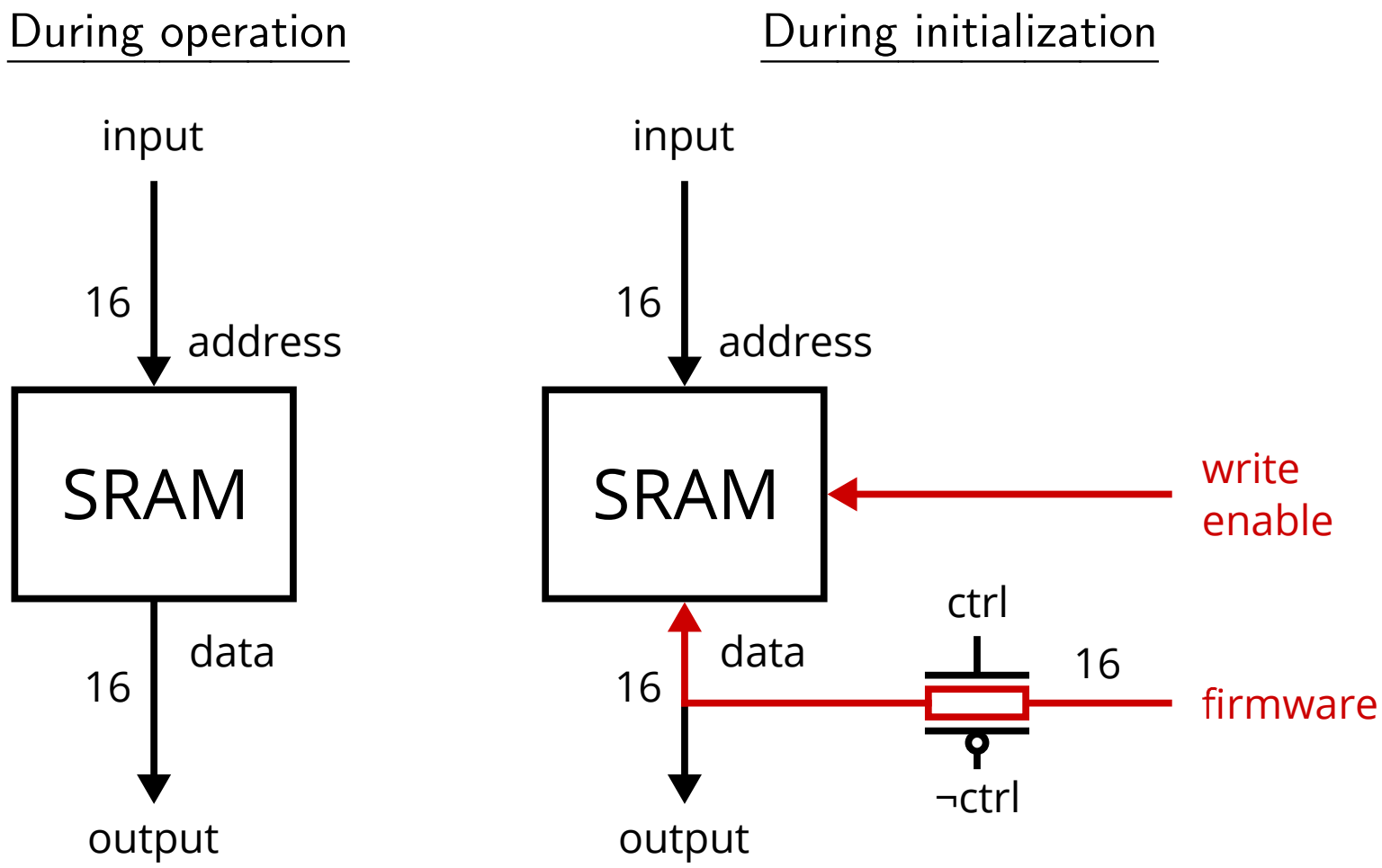
Computers were once BIG



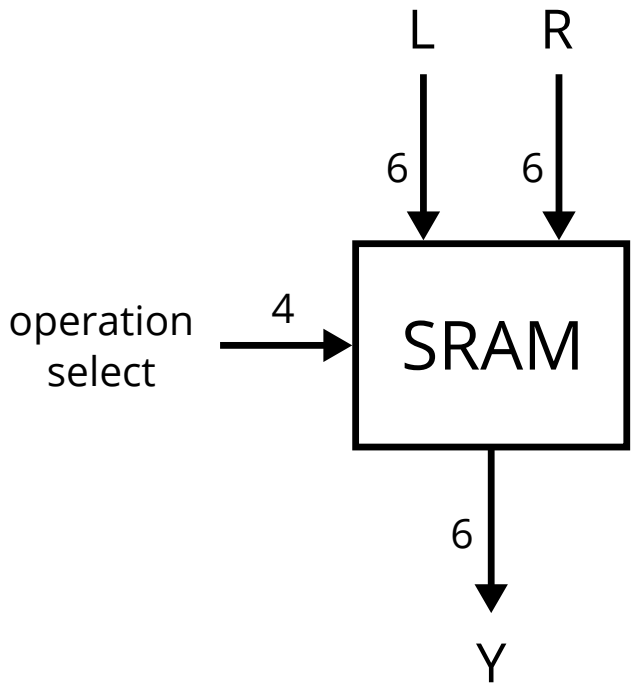
The speaker using an IBM 1130.

Norwester, 62, p. 73 (1986).
Used with permission.

SRAM logic gate

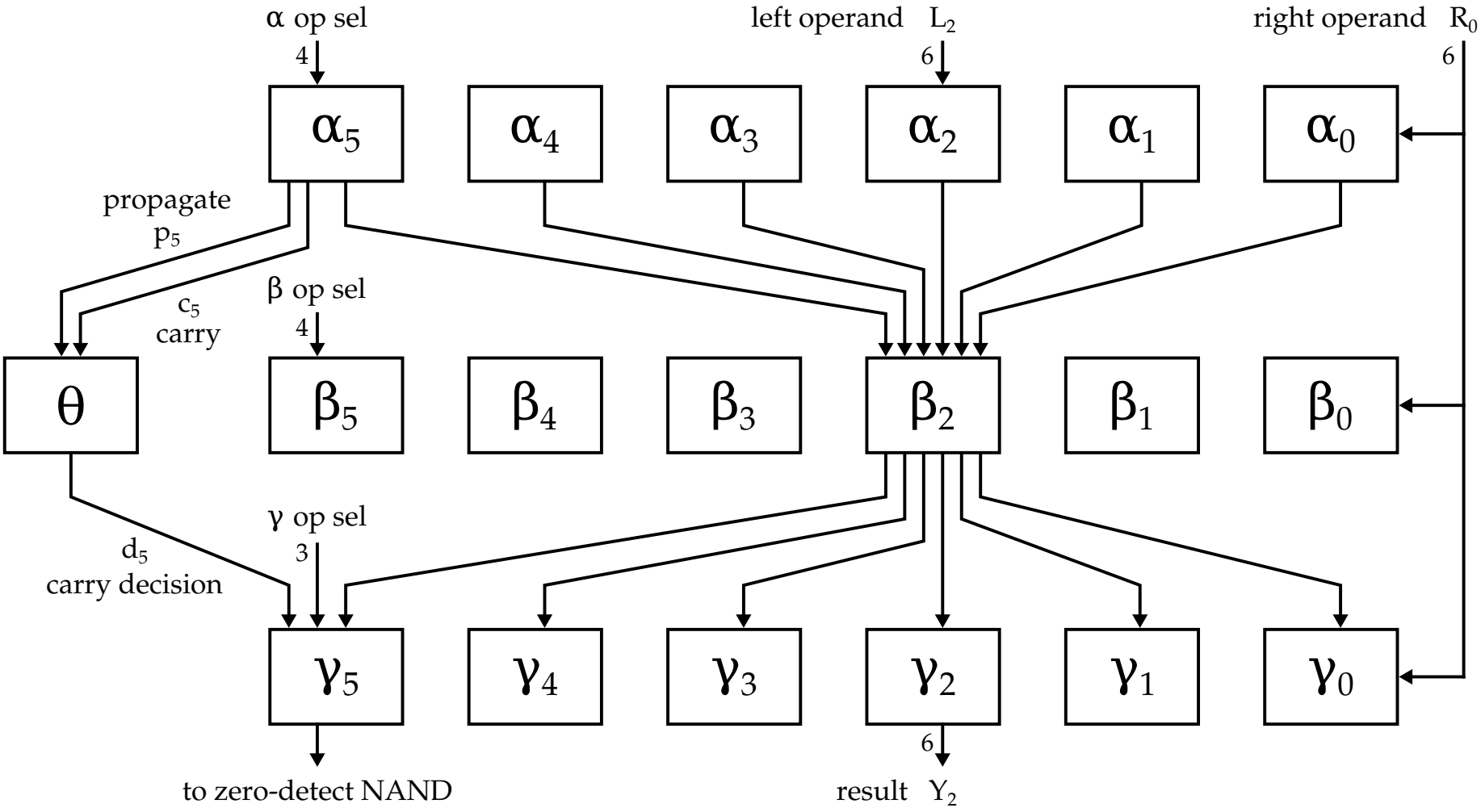


SRAM logic gate sample application

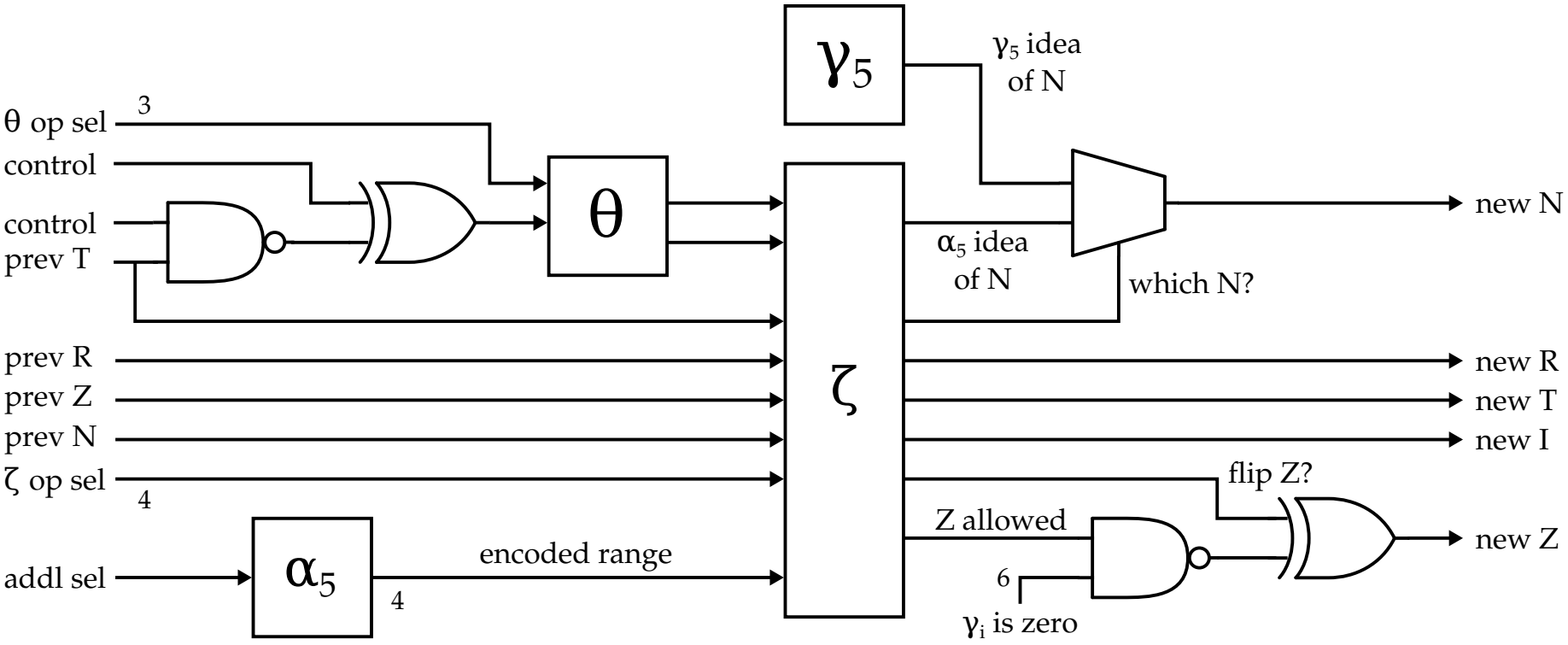


Slot	Use
0	L + R
1	L − R
2	R − L
3	NOT L
4	L AND R
5	L NAND R
6	L OR R
7	L NOR R
8	L XOR R
9	L AND NOT R
10	L × R, low 6 bits
11	L × R, high 6 bits
12	shifts/rotations of L (select by R)
13	permutations of L (select by R)
14	L encrypted with key R
15	L decrypted with key R

Block diagram of 36-bit ALU



Flag handling for 36-bit ALU



CPU flag meanings

Flag	Name	Purpose
I	Interrupt	array bound check failed
N	Negative	arithmetic: result < 0; logic: bit 35 set
R	Range	a previous result did not fit destination
T	Temporal range	the most recent result did not fit destination
Z	Zero	result is all zeros

32 supported subtraction operations

```
; The assembler for the ALU simulation uses this syntax.

unsigned au bu cu
signed   as bs cs

; ordinary        ; subtract        ; reverse         ; reverse subtract
; subtract        ; with carry      ; subtract        ; with carry

cu = au - bu      cu = au -- bu     cu = au ~- bu     cu = au ~-- bu
cu = au - bs      cu = au -- bs     cu = au ~- bs     cu = au ~-- bs
cu = as - bu      cu = as -- bu     cu = as ~- bu     cu = as ~-- bu
cu = as - bs      cu = as -- bs     cu = as ~- bs     cu = as ~-- bs
cs = au - bu      cs = au -- bu     cs = au ~- bu     cs = au ~-- bu
cs = au - bs      cs = au -- bs     cs = au ~- bs     cs = au ~-- bs
cs = as - bu      cs = as -- bu     cs = as ~- bu     cs = as ~-- bu
cs = as - bs      cs = as -- bs     cs = as ~- bs     cs = as ~-- bs
```

Screenshot from ALU simulation

```
$ # Example: logical shift left 5 bits
$ ./vm 000077007777 lsl 050505050505

        L 000000 000000 111111 000000 111111 111111      t -> t
        R 000101 000101 000101 000101 000101 000101
 =  alpha 000000 000000 111111 000000 111111 111111    p 001011
 L   beta 010110 010110 010110 010110 010110 001011    c 010100
 lL gamma 000000 011111 100000 011111 111111 100000    d 000000 0
    flags i n r t z
 unsigned 528613344
   signed 528613344
$
```

Projected CPU capabilities and metrics

- 36-bit architecture
- 10 MIPS
- preemptive multitasking
- memory protection
- I/O bus protection
- 20×20 cm approximate size
- first unit cost $<$ \$1000

Potential applications

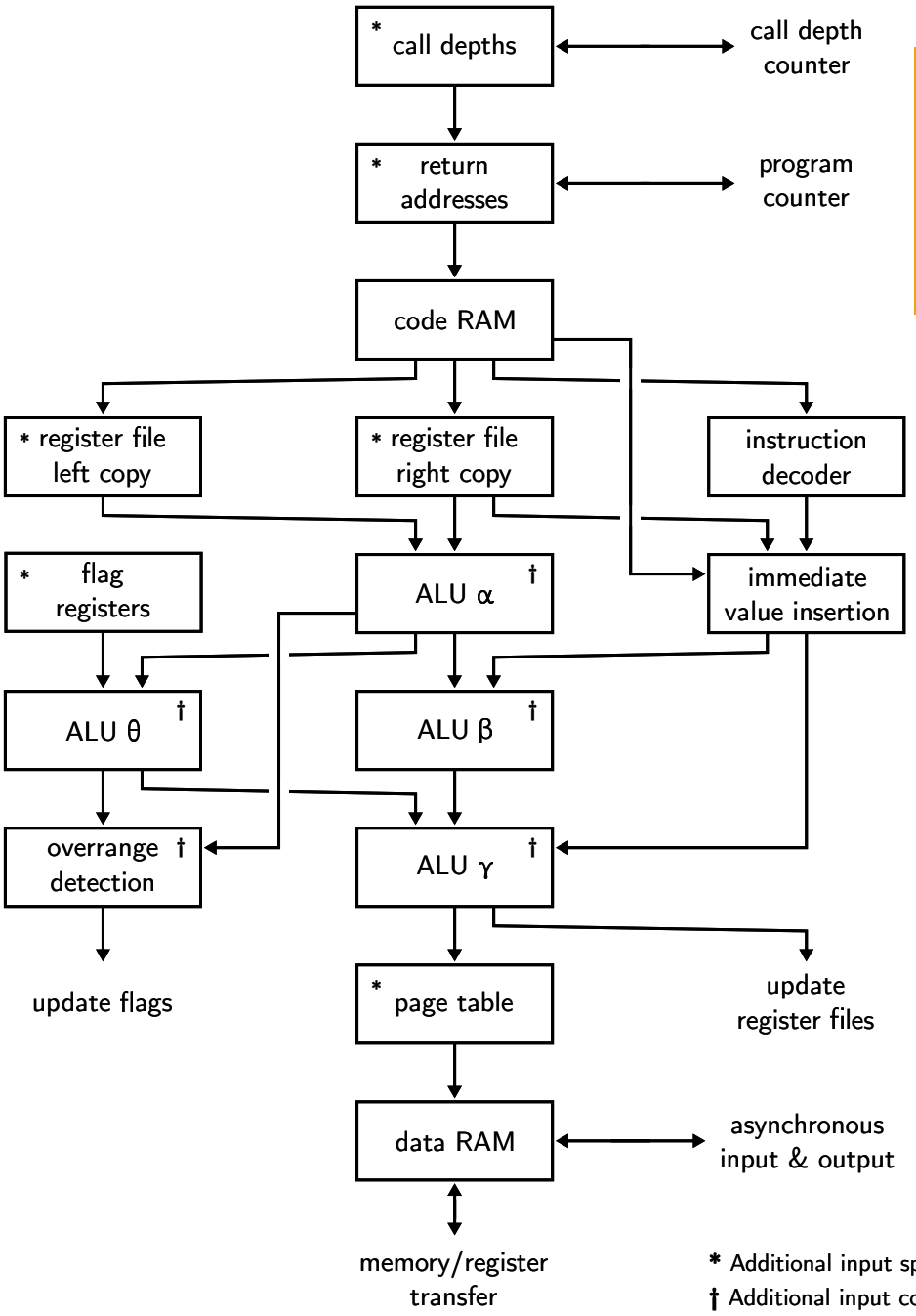
Fast enough for

- hardened desktop applications
- electronic mail
- light- to moderate-use servers
- control objects that move
- process controls
- peripheral and device controllers
- telephony
- modest Ethernet switches

Too slow for

- most Web surfing
- machine learning
- image and video processing
- self-driving vehicles
- fast raster or vector graphics
- fast symmetric cryptography
- fast asymmetric cryptography
- micro air vehicles

CPU cycle depiction with static RAM drawn as boxes



* Additional input specifies the program that is running.

† Additional input comes from the instruction decoder.

Figure enlarged on proposal page 29

Register file highlights

- 128 programs $\times$ 512 registers available all at once
- registers are fully orthogonal
- not accessed by number in programs, but declared like variables
- assembler and linker can help allocate and consolidate registers
- no save/restore needed to switch programs
- duplicate (left and right operand) copies for speed
- immediate value (numeric constant) insertion via right operand

Primary storage ("main memory") highlights

- Harvard architecture
  (physically separate code, data, and call return stack memory)
- 100% SRAM (no DRAM)
- page table for data memory
- no user control of return address stack
- no data co-located with return addresses
- no direct support for recursion
- branch destinations always hard-coded in instructions
- no `longjmp`, pointers to functions, etc.
- no user control of program memory or contents

Privilege mechanisms

- restrictive, inflexible instruction format
- stack and code not addressible via LOAD or STORE
- OS to filter privileged instructions at program load time
- OS to filter branches to library code at program load time
- data memory accessed via page table

Implementation challenges to come

- high pin count (capacitance) at many nodes
- provision needed for OS writes to many subsystem RAMs
- I/O subsystem must balance security, succinctness, and speed
- many components necessary for firmware loading
- pipelining and nanosecond budget to reach 10 MIPS
- layout and assembly for real-world prototype
- potential need to troubleshoot assembled system
- scope to pupil ratio

Original contributions

ALU Incremental contributions associated with SRAM implementations of carry-skip adders, fast multipliers, logarithmic shifters, and superpositions thereof

CPU World's first "gold standard" for transparently functioning, fully auditable, user-constructable computers for integrity- and confidentiality-critical missions

When is Marc done?

- This will take 12 months.
- Requirements for the machine are on page 32 of the proposal. They align with what you have heard today.
- I will submit an article to *IEEE Computer Architecture Letters* no later than October 2 (23 days from today).
- If *CAL* does not accept this article, I could extend it for:
  - *ACM Transactions on Information and System Security*
  - *ACM Transactions on Computer Systems*
  - *IEEE Transactions on Dependable and Secure Computing*
  - *42nd IEEE Symposium on Security and Privacy* (not preferred)

Questions and answers

Thank you for spending this time with us!