Computer Architecture

Daniel Page

Department of Computer Science,
University Of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB. UK.
⟨csdsp@bristol.ac.uk⟩

September 17, 2026

Keep in mind there are *two* PDFs available (of which this is the latter):

1. a PDF of examinable material used as lecture slides, and
2. a PDF of non-examinable, extra material:
   - the associated notes page may be pre-populated with extra, written explaination of material covered in lecture(s), plus
   - anything with a "grey'ed out" header/footer represents extra material which is useful and/or interesting but out of scope (and hence not covered).

Notes:

Notes:

- Agenda: recalling that COMS10015 comprises 3 high-level themes, i.e.,

  Theme #1 $\Rightarrow$ "from Mathematics and Physics to digital logic"
  Theme #2 $\Rightarrow$ "from digital logic to computer processors"
  Theme #3 $\Rightarrow$ "from computer processors to software applications"

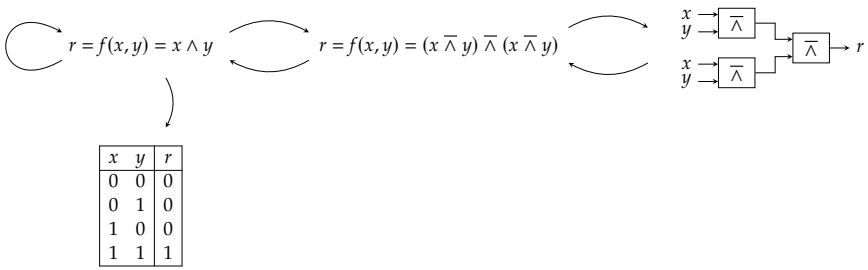
  the aim is to summarise (or wrap-up), by

  1. looking *backward* $\Rightarrow$ what we *have done* in TB1
  2. looking *forward* $\Rightarrow$ what we *will do* in TB2

Notes:

Unit summary (1)

Looking backward, low-level perspective: arc #1 = "no remaining magic between abstract and concrete computation"

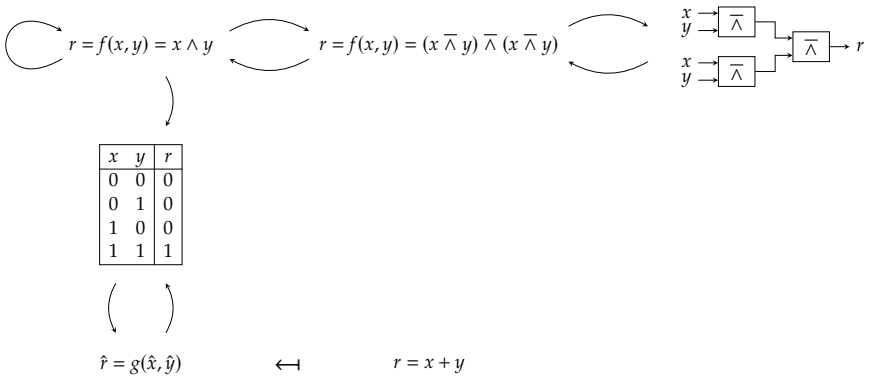


x	y	r
0	0	0
0	1	0
1	0	0
1	1	1

Notes:

Unit summary (1)

Looking backward, low-level perspective: arc #1 = "no remaining magic between abstract and concrete computation"



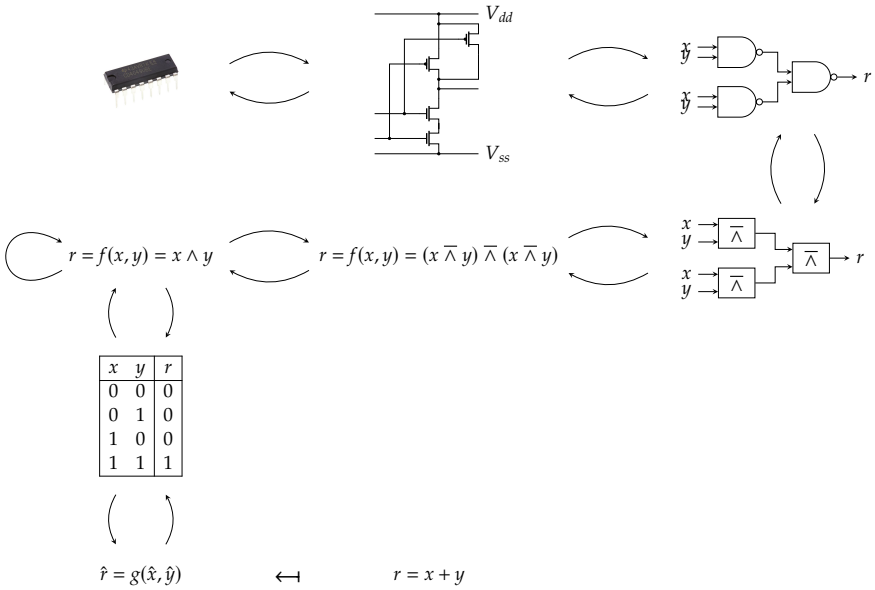
x	y	r
0	0	0
0	1	0
1	0	0
1	1	1

$\hat{r} = g(\hat{x}, \hat{y})$ $\longmapsto$ $r = x + y$

Notes:

Unit summary (1)

Looking backward, low-level perspective: arc #1 = "no remaining magic between abstract and concrete computation"



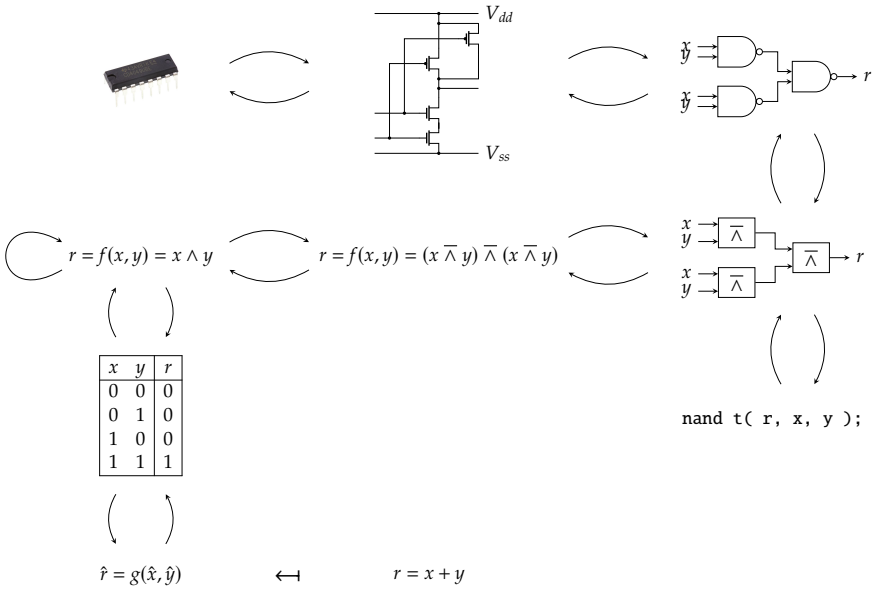
x	y	r
0	0	0
0	1	0
1	0	0
1	1	1

$\hat{r} = g(\hat{x}, \hat{y})$ $\longmapsto$ $r = x + y$

Notes:

Unit summary (1)

Looking backward, low-level perspective: arc #1 = "no remaining magic between abstract and concrete computation"



x	y	r
0	0	0
0	1	0
1	0	0
1	1	1

Notes:

Unit summary (2)

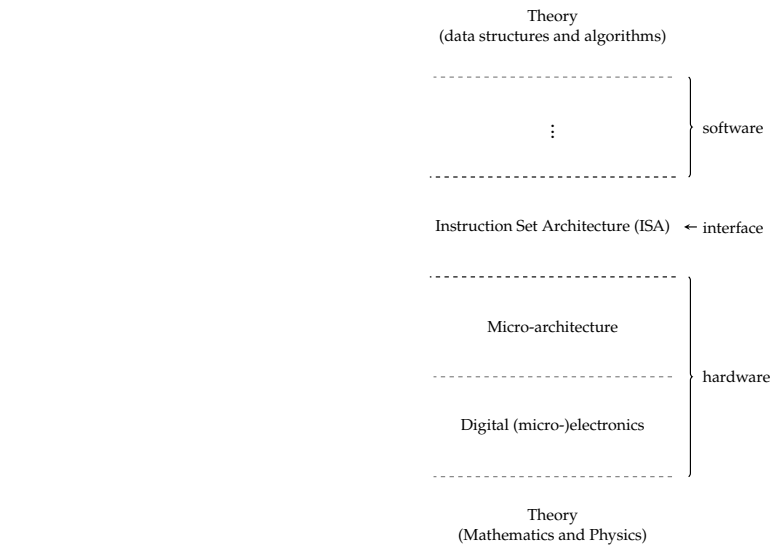
Looking backward, low-level perspective: arc #2 = "progressively more involved versions of addition"

theory	$\rightsquigarrow$	representation		unsigned and signed integer
theory	$\rightsquigarrow$	computation		"school-book" addition algorithm
⋮				
combinatorial logic	$\rightsquigarrow$	fixed function computation,	not stateful	full-adder cell, ripple-carry adder
sequential logic	$\rightsquigarrow$	fixed function computation,	stateful	counter
FSM	$\rightsquigarrow$	fixed function computation,	stateful	
RM	$\rightsquigarrow$	not fixed function computation,	stateful	counter machine
⋮				
micro-processor	$\rightsquigarrow$	not fixed function computation,	stateful	add instruction

Notes:

Unit summary (3)

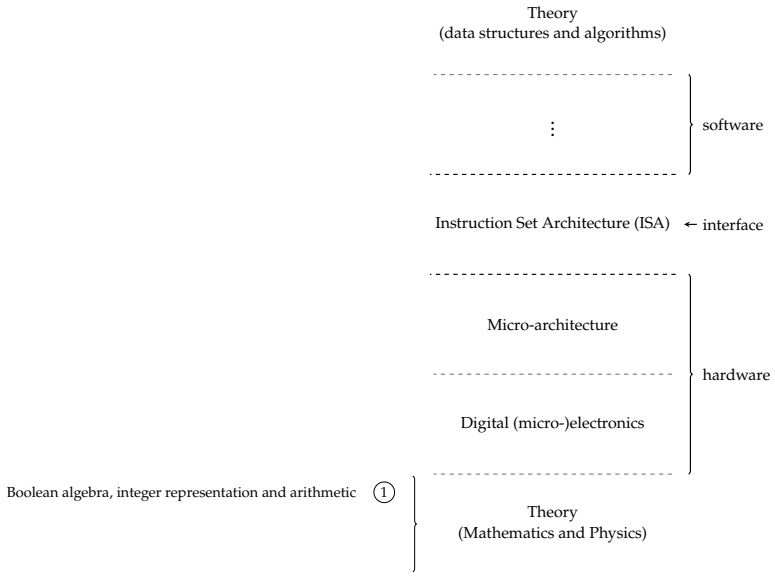
Looking backward, high-level perspective



Notes:

Unit summary (3)

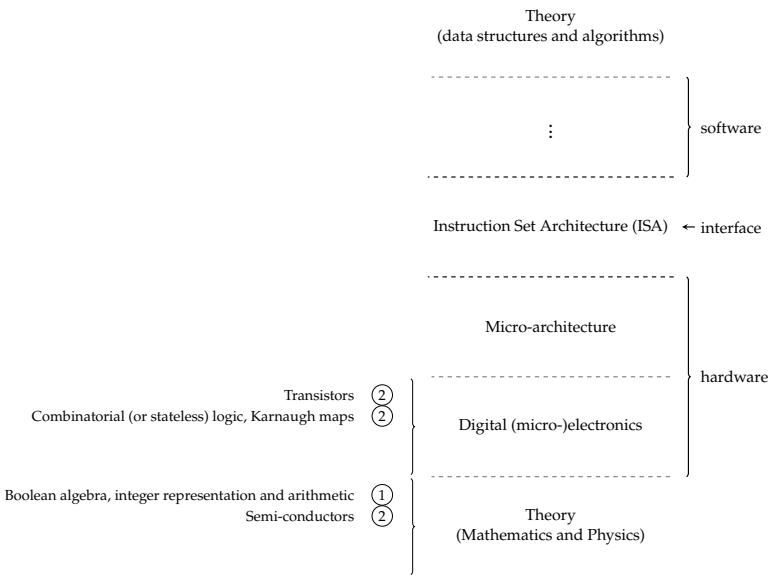
Looking backward, high-level perspective



Notes:

Unit summary (3)

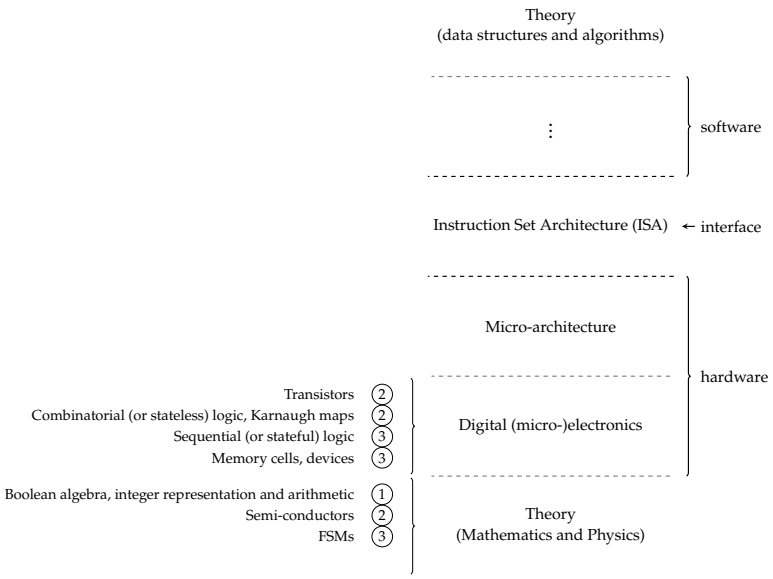
Looking backward, high-level perspective



Notes:

Unit summary (3)

Looking backward, high-level perspective

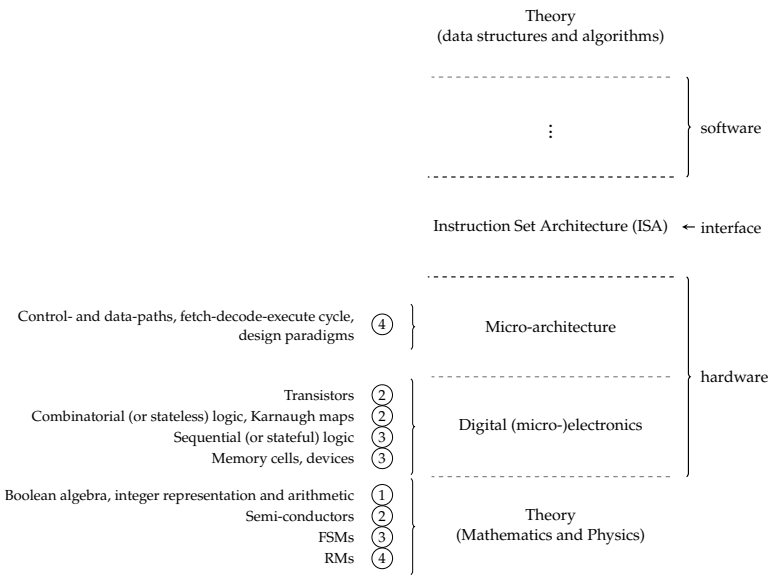


Notes:

© Daniel Page ⟨cs@bristol.ac.uk⟩
Computer Architecture
University of BRISTOL
git # 0073a8663 @ 2026-09-15

Unit summary (3)

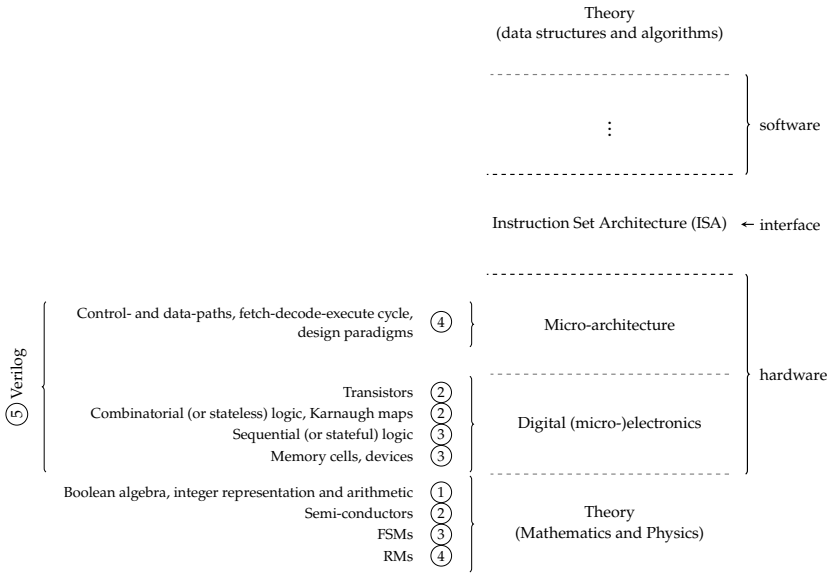
Looking backward, high-level perspective



Notes:

Unit summary (3)

Looking backward, high-level perspective



Notes:

Unit summary (4)
Looking forward

- ... yet to come, in TB2:

1. Instruction Set Architecture (ISAs):
   - instruction set design: instruction classes; addressing modes; instruction encoding and decoding
   - real-world examples: ARMv7-A
2. micro-architecture (revisited):
   - pipelined instruction execution
   - von Neumann bottleneck, memory hierarchy; cache memories
3. (system) software:
   - development tools: assembly language; assembly and linkage processes; debuggers; compilers
   - support for structured programming (e.g., function calls)
   - support for operating systems: interrupts; protection; virtual memory
4. ...

Notes:

Conclusions

- Take away points: *hopefully*, TB1 has delivered

1. some *understanding*,
   - Boolean algebra; integer representation and arithmetic
   - physical design of logic components (e.g., logic gates from transistors)
   - use of combinatorial logic components (e.g., Karnaugh maps)
   - use of sequential logic components (e.g., state machines)
   - processor paradigms: counter, accumulator, stack, and register machines; von Neumann vs. Harvard architecture; RISC vs. CISC
   - ...
2. some *skills*,
   - Verilog-based modelling and simulation of digital logic
   - ...
3. some *experience*,
   - hierarchical design (via abstraction, and "understand-design-implement" ethos)
   - debugging strategies
   - ...

which will be further extended and enhanced by TB2.

Notes:

References

Notes: