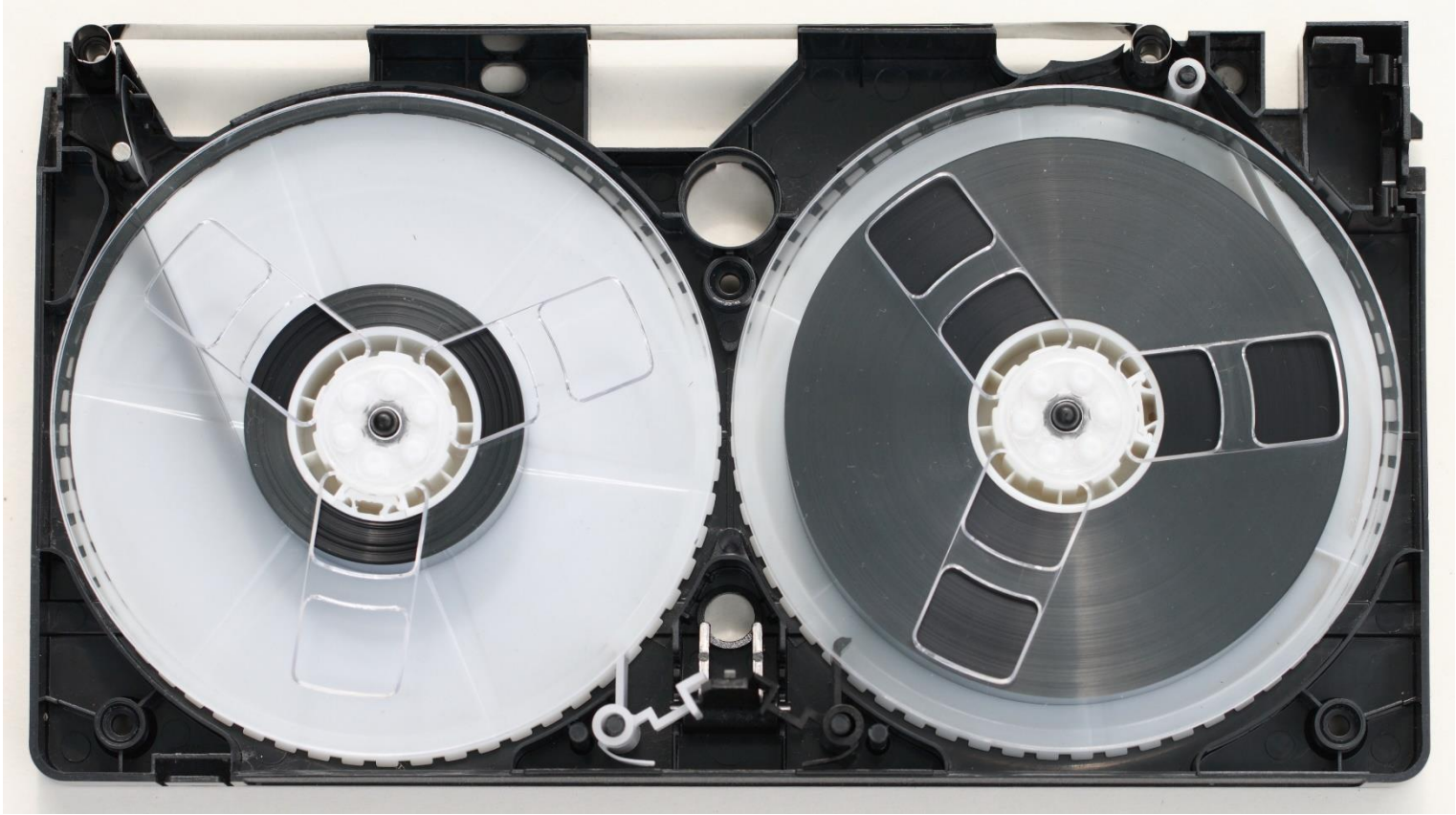
Advanced circuit design

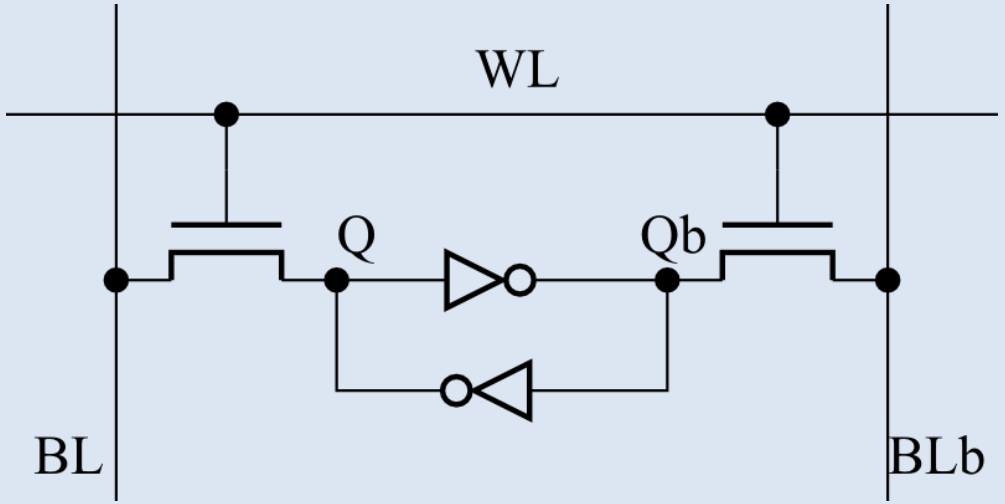
Andrey Mokhov, Alex Yakovlev
Danil Sokolov, Victor Khomenko

Newcastle University, UK
andrey.mokhov@ncl.ac.uk

Part I:
Asynchronous memory



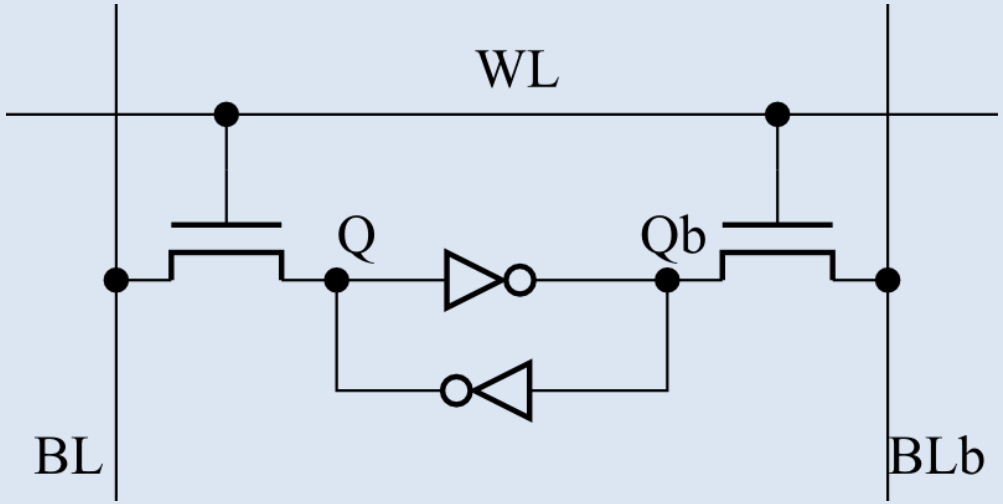
Conventional 6T SRAM



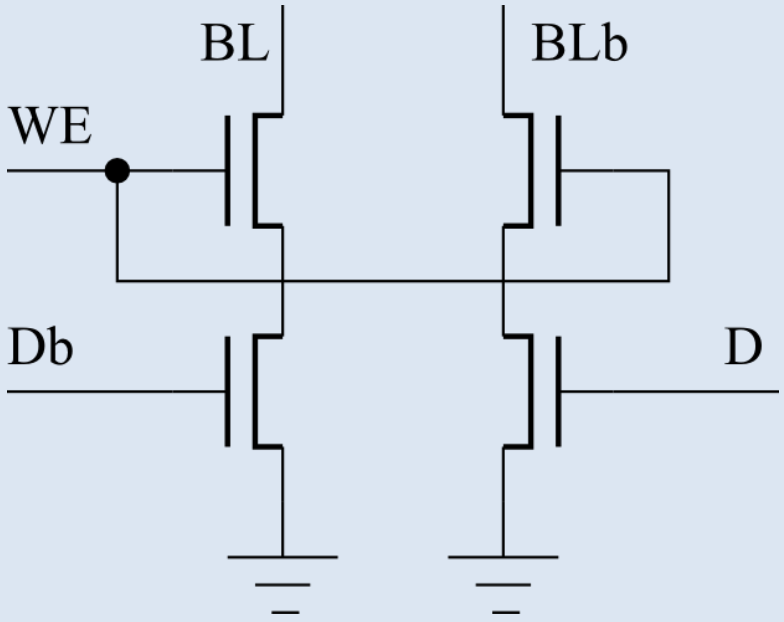
Bit cell

Reading: precharge bit lines, assert WL, **sense bit line changes**

Conventional 6T SRAM



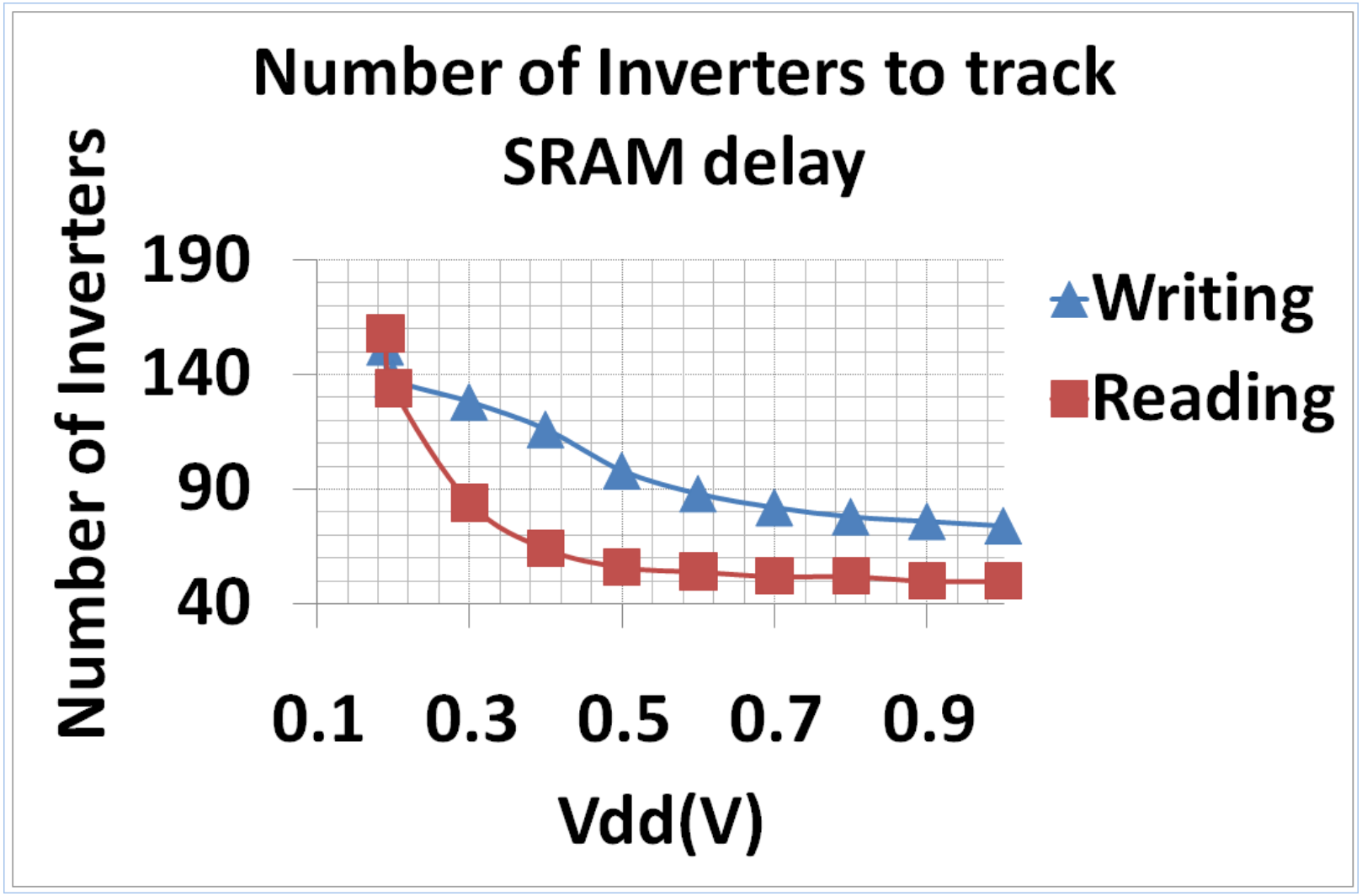
Bit cell



Write driver

Writing: set data lines, assert WL and WE, **wait for a while…**

Problem: how long to wait?



Problem: how long to wait?

SRAM 6T cell delays are difficult to match accurately

- When V_{dd} = 1V, SRAM read delay is ≈50 inverters
- When V_{dd} = 190mV, SRAM read delay is ≈158 inverters
- Read and write delays scale differently with V_{dd}

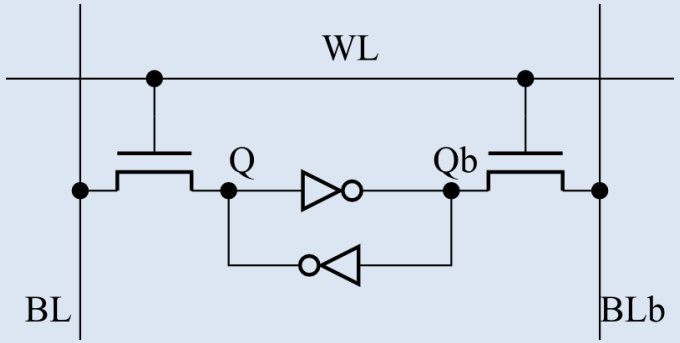
Conventional solutions

- Use different delay lines for different ranges of V_{dd}
- Duplicate an SRAM line to act as a reference delay line
- Need voltage references, costly in area and energy

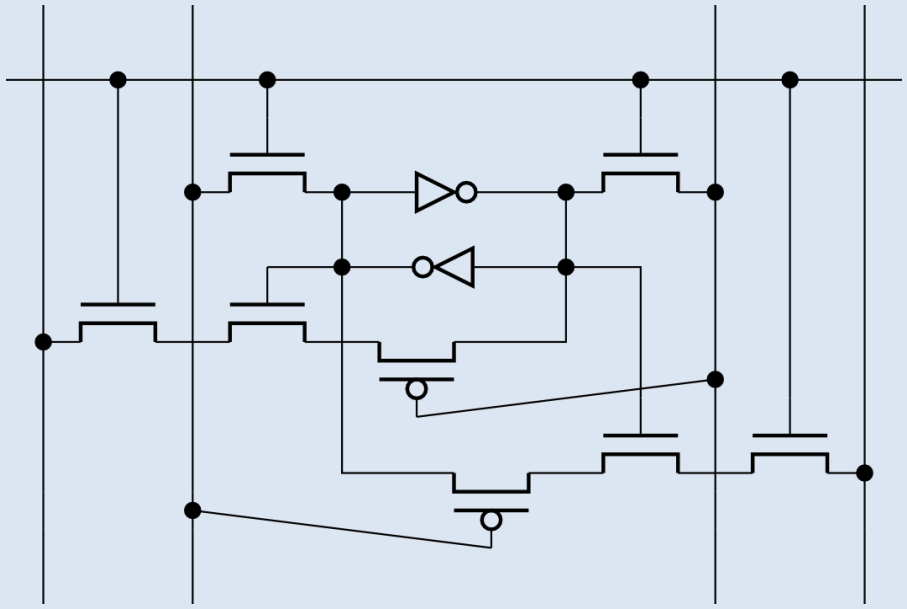
Asynchronous solution with completion detection

- Speed-independent, free from voltage references
- Developed by A. Baz *et al.* (PATMOS 2010, JOLPE 2011)

Low-level completion detection



Bit cell

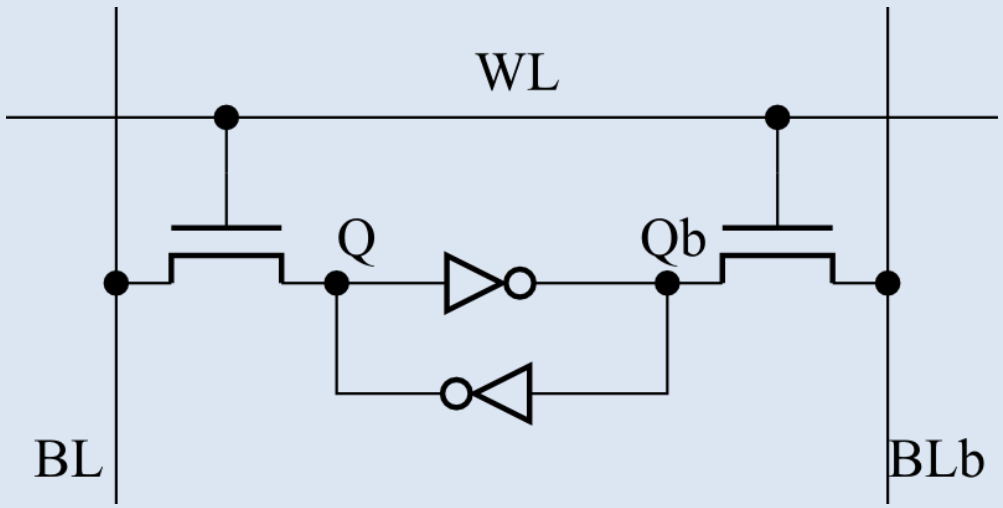


Bit cell with CD

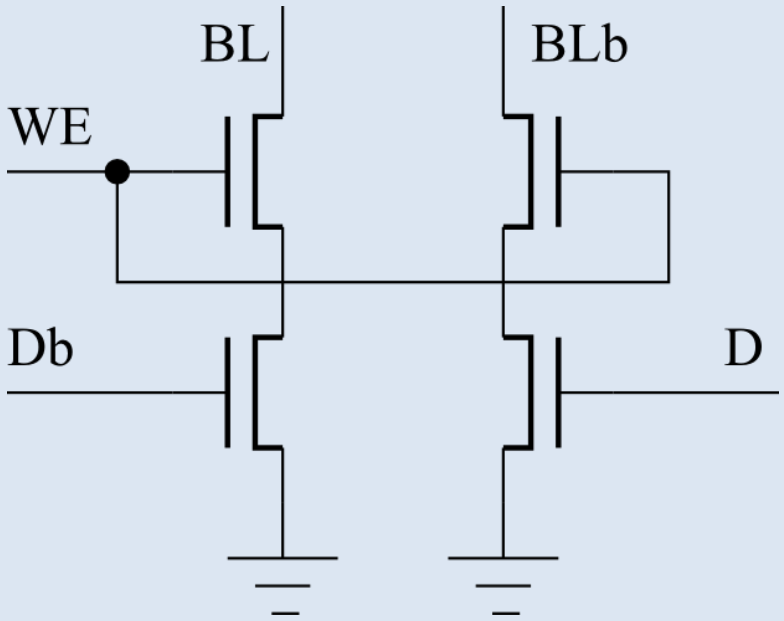
True completion detection **both for reading and writing**

Too costly!

Back to conventional 6T SRAM



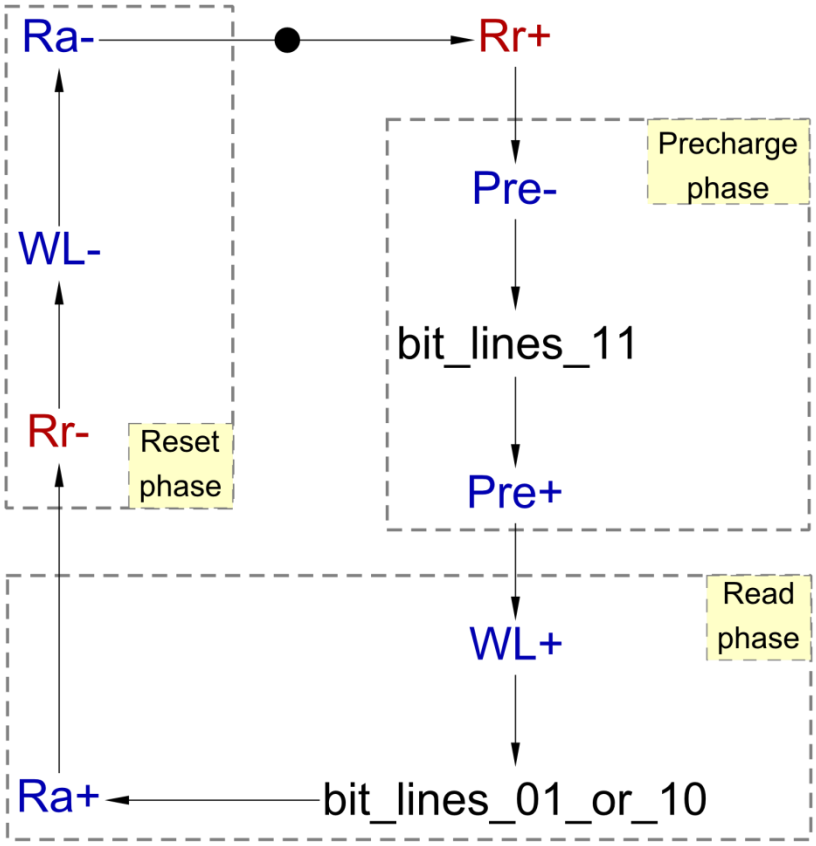
Bit cell



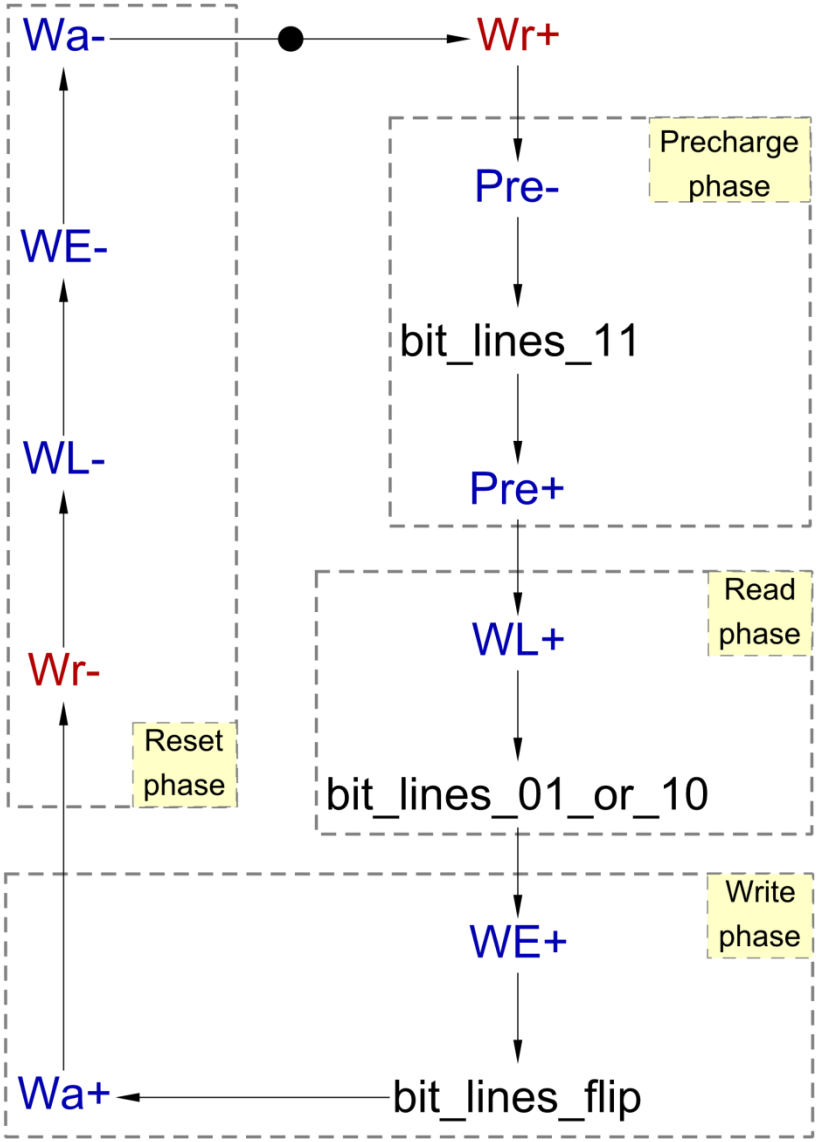
Write driver

Idea 1: completion detection is possible when the bit is flipped
Idea 2: read before writing to check if the bit will be flipped

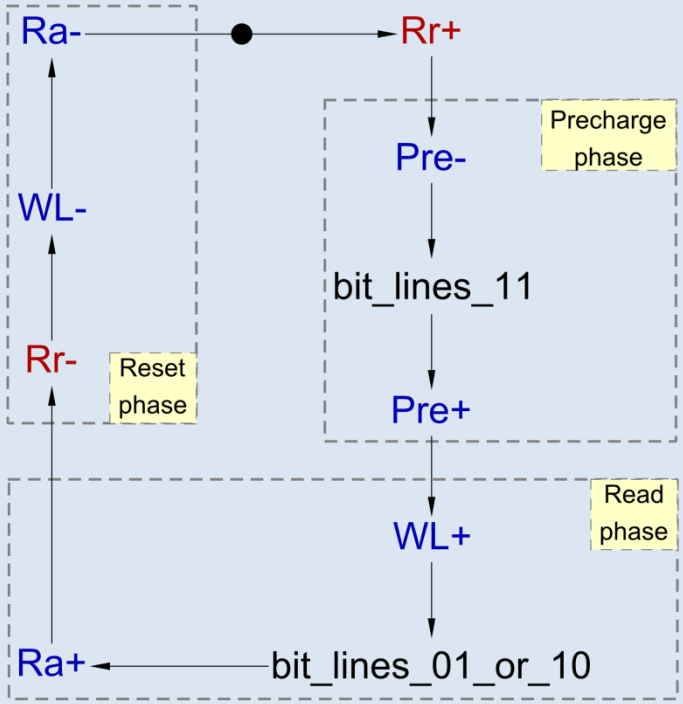
Specification: read scenario



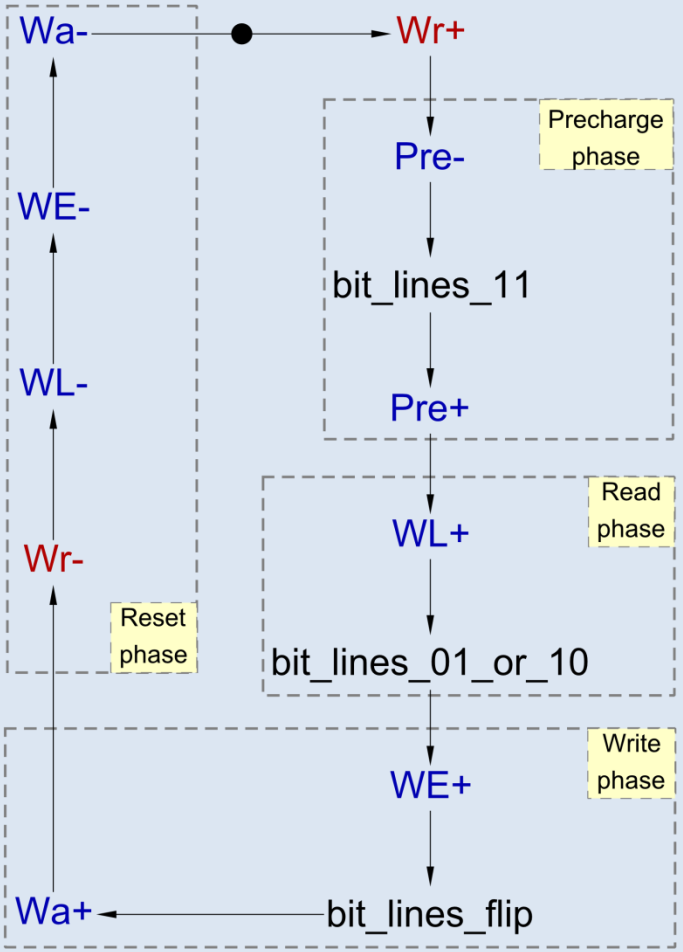
Specification: write scenario



Specification: composing scenarios

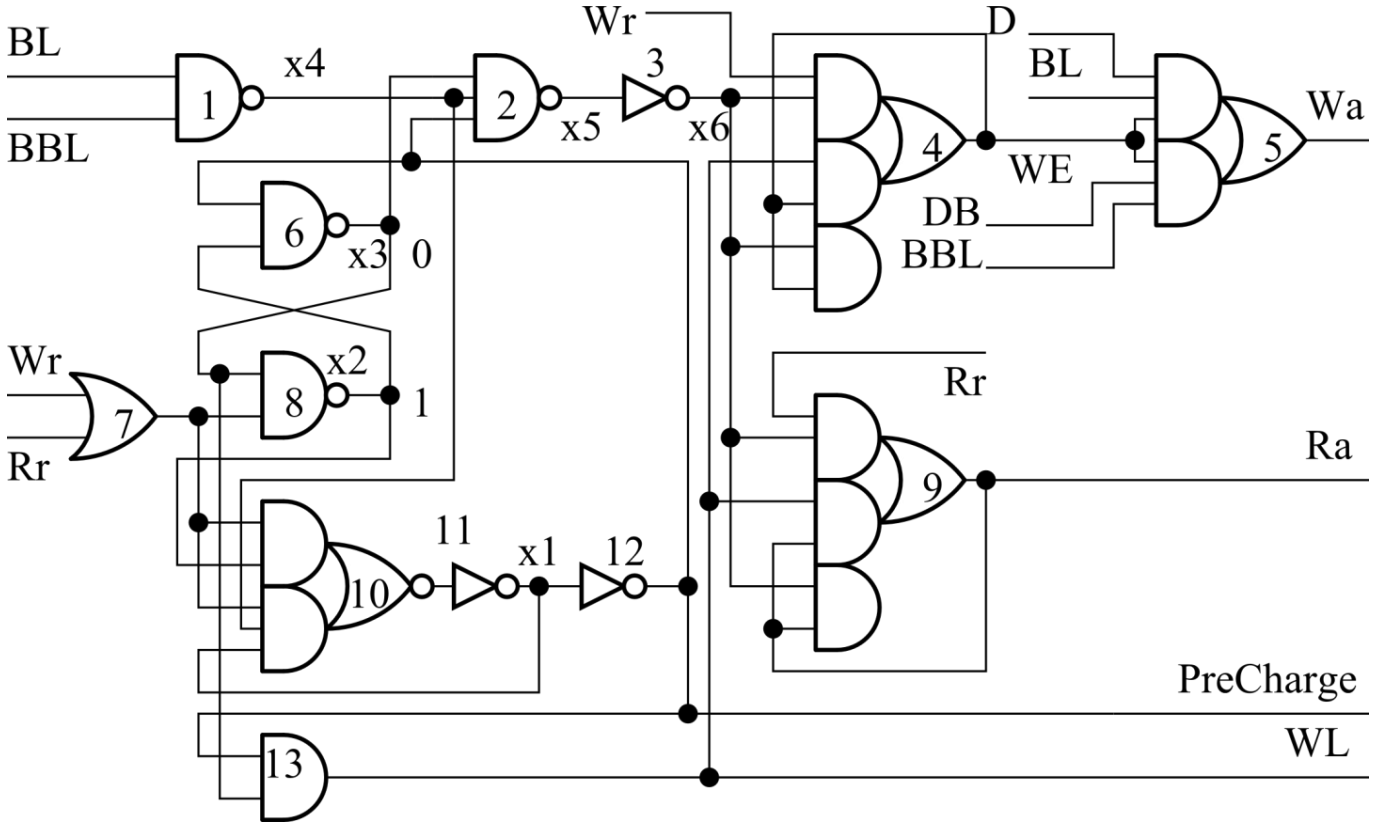


Read



Write

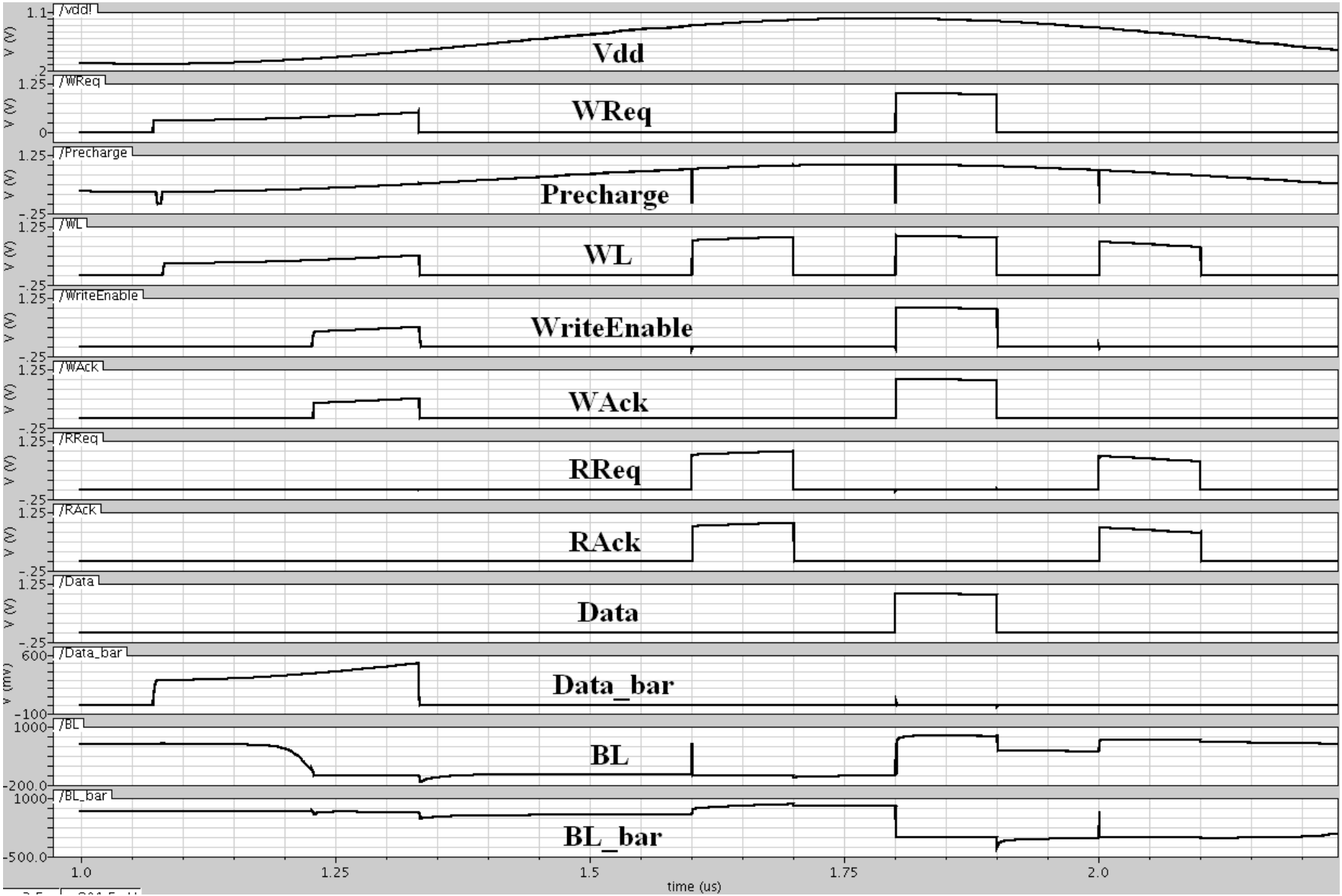
Asynchronous SRAM controller



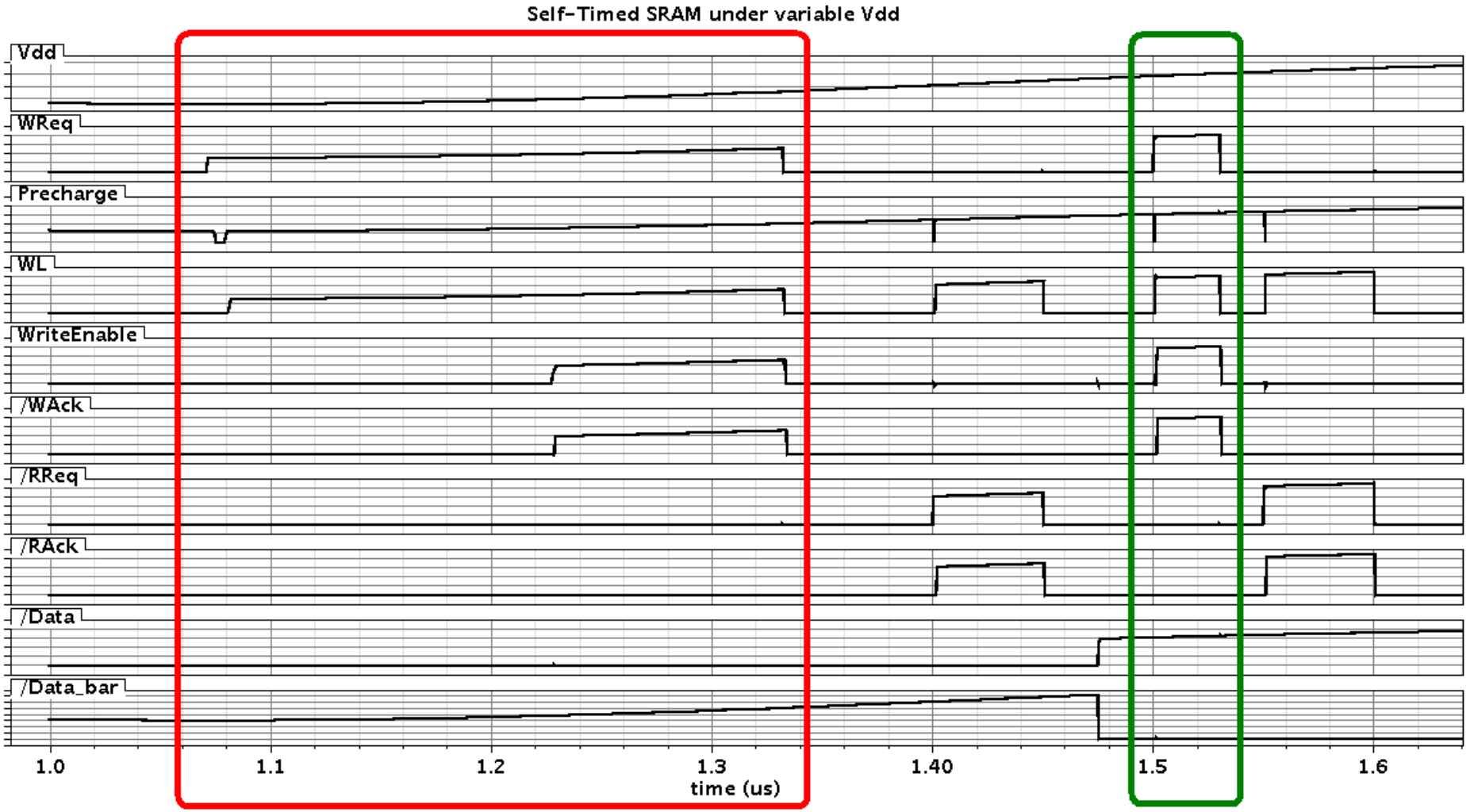
Hand made, hence not guaranteed to be speed-independent

We will design a provable correct implementation in Part II

Tolerating variable voltage supply



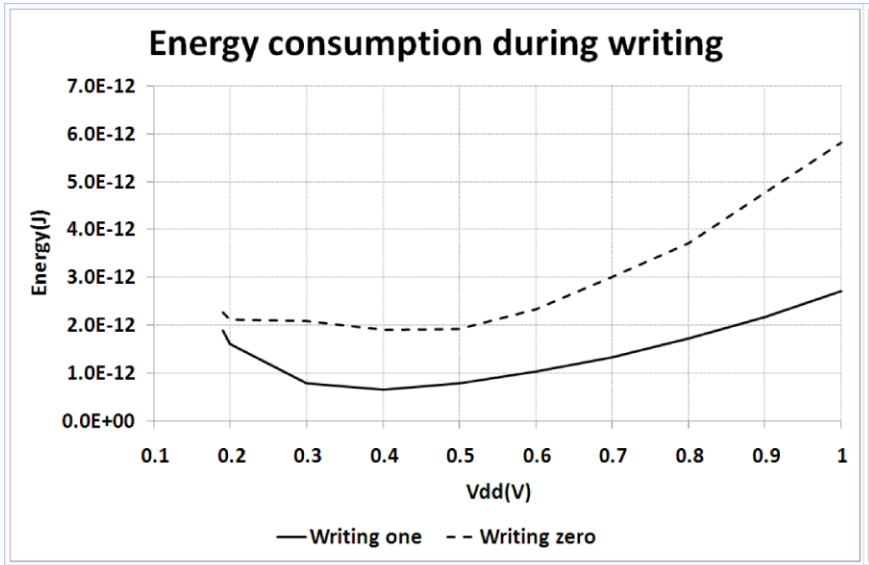
Tolerating variable voltage supply

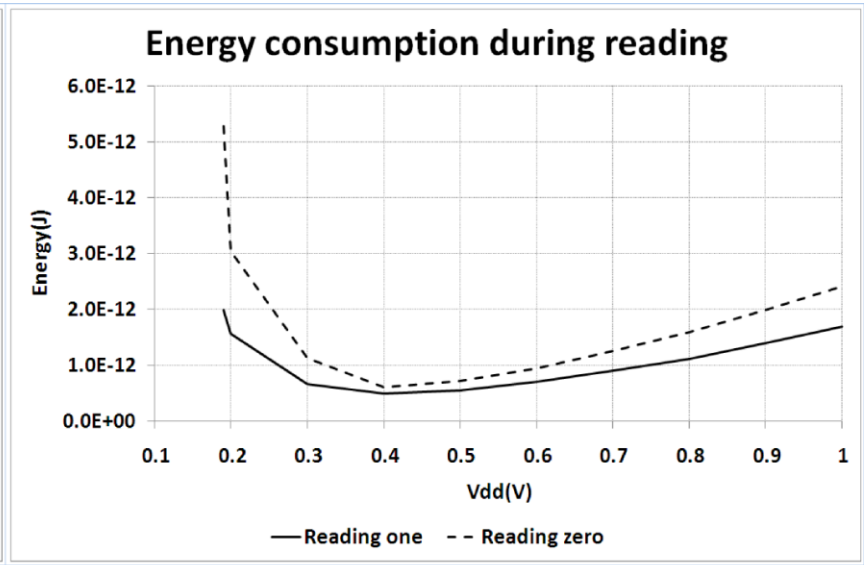


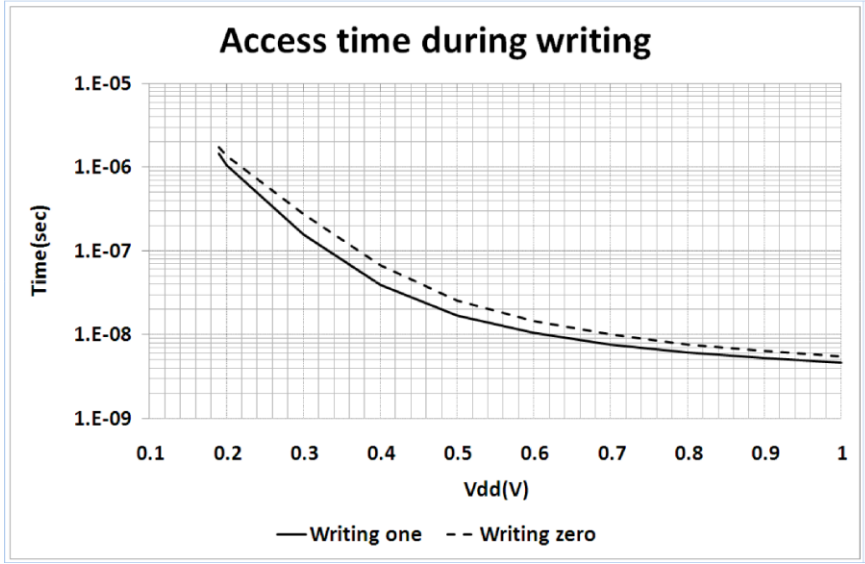
Low voltage, slow response

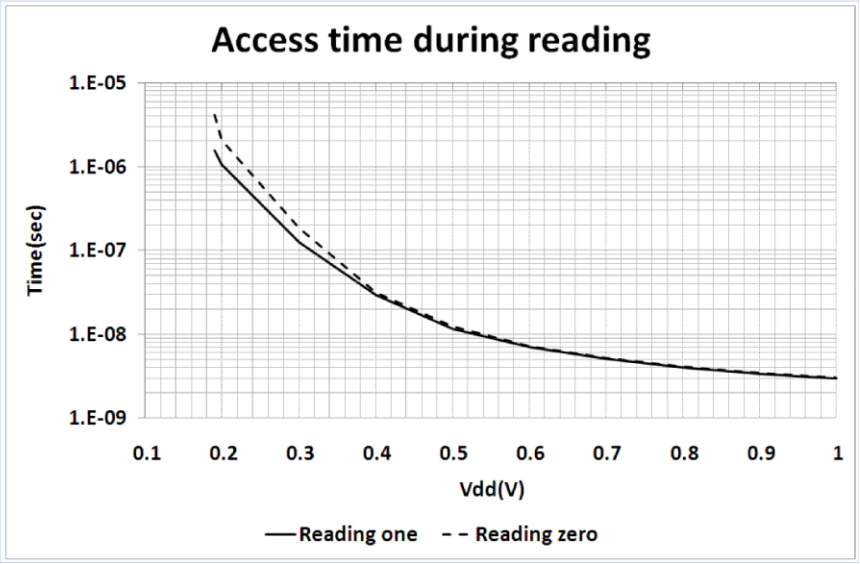
High voltage, fast response

Trading energy for performance









Part II: Design of a speed-independent SRAM controller

Design in Workcraft

Summary

Memory is inherently asynchronous

- Read & write completion can be reliably detected
- Conventional synchronous ‘handcuffs’ (matching delay lines) are clumsy and costly

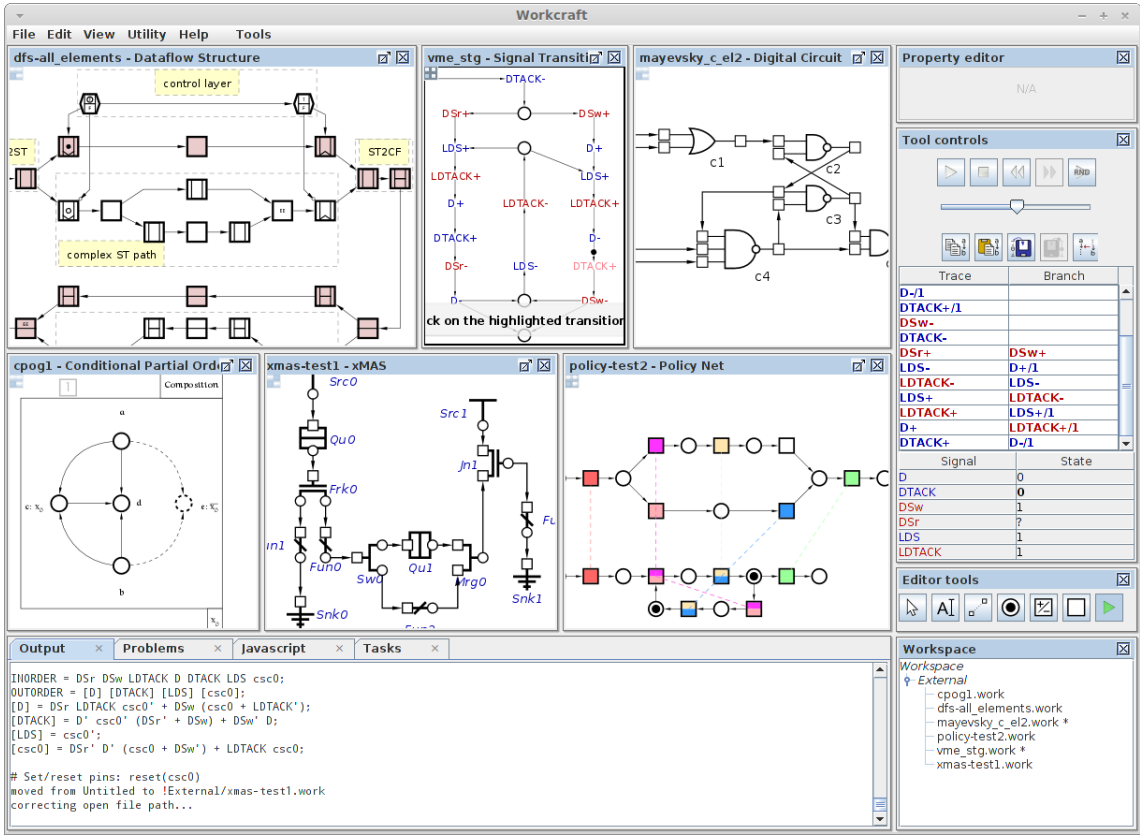
Typical ‘little digital’ control, fully supported by Workcraft design, synthesis and verification flow

Ongoing and future work – **you can contribute!**

- Integrate asynchronous SRAM into a real system
- Opportunity for new memory architectures

Part III:
A glimpse of the future

```
21 memory_access    = pciu → ifu + ma
22 cond_alu_Rn_Rn   = pciu → ifu + al
23 cond_alu_123_Rn = lt  ? ((alu + p
24                  + lt' ? ((alu + p
25 cond_alu_123_PC = lt  ? ((alu + p
26                  + lt' ? ((alu + p
27
28 (x, x') = literals "x"
29 (y, y') = literals "y"
30 (z, z') = literals "z"
31
32 processor = opcode [x',y',z'] ? a
33           + opcode [x ,y ,z'] ? a
34           + opcode [x ,y',z ] ? a
35           + opcode [x',y ,z'] ? a
36           + opcode [x ,y',z'] ? m
37           + opcode [x',y',z ] ? c
38           + opcode [x ,y ,z ] ? c
39           + opcode [x',y ,z ] ? c
```



Friends of asynchronous design

Analogue world
- Power regulation
- Sensing
- Synchronisers, memory

Variability
- Device parameters, stochastic effects
- Data dependency of components

Environmental **uncertainty**
- Unstable voltage supply
- Unpredictable service demand

Massive **concurrency**

Applications of asynchronous design

On-chip infrastructure

- Analogue liaisons
- SoC & NoC routers
- Little digital controllers (memory, network, I/O)

Internet of Things

- Batteries not included
- Unpredictable service demand

Dataflow computing

- Massive concurrency
- Optimised for throughput

Challenges in asynchronous design

Tool integration

- Mixing synchronous and asynchronous
- Mixing digital and analogue
- Reuse of legacy synchronous designs
- Testing

Scaling from little digital to big digital

Steep learning curve

- Better, more user friendly tools (like Workcraft)
- Simpler specification models and languages

Thank you!

Questions are welcome