ORRAM: An OpenROAD-Integrated RAM Generator Using Standard Cells

Brayden Louie*[1], Thinh Nguyen*[1], Matt Liberty[2], Austin Rovinski[1]

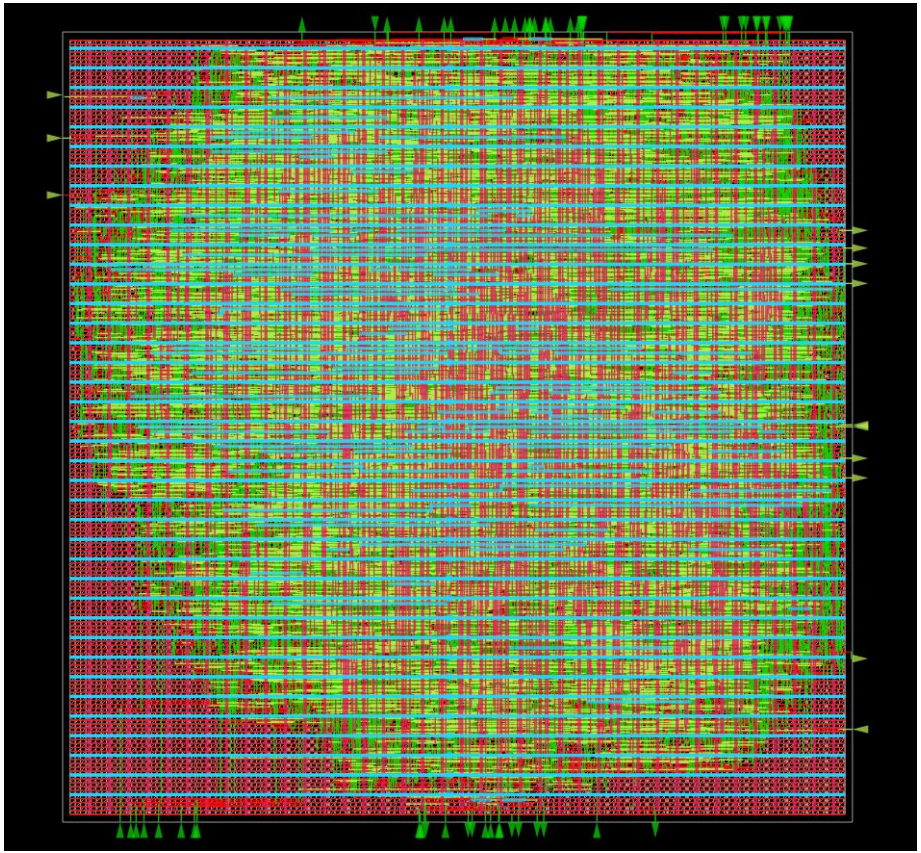
[1]New York University, [2]Precision Innovations, Inc.

*Equal contribution

06.28.26

Problem & Motivation

- Problem: Memory generation
- RTL Inference
  - Densely interconnected FFs
  - Degrades PPA
  - Slows down flow

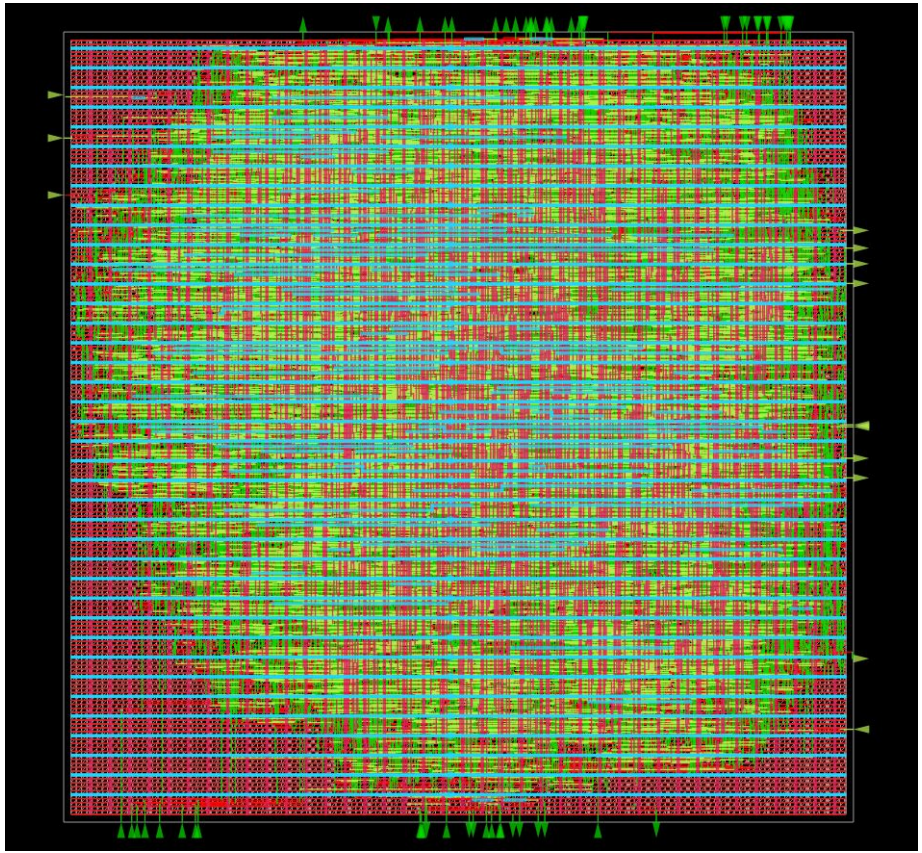


Memory generated from RTL Inference

NYU

Problem & Motivation

- Problem: Memory generation
- RTL Inference
  - Densely interconnected FFs
  - Degrades PPA
  - Slows down flow
- Memory compilers
  - Single optimized block
  - Proprietary
  - PDK-specific
  - Requires specialized cells
- Gap in open-source flows



Memory generated from RTL Inference

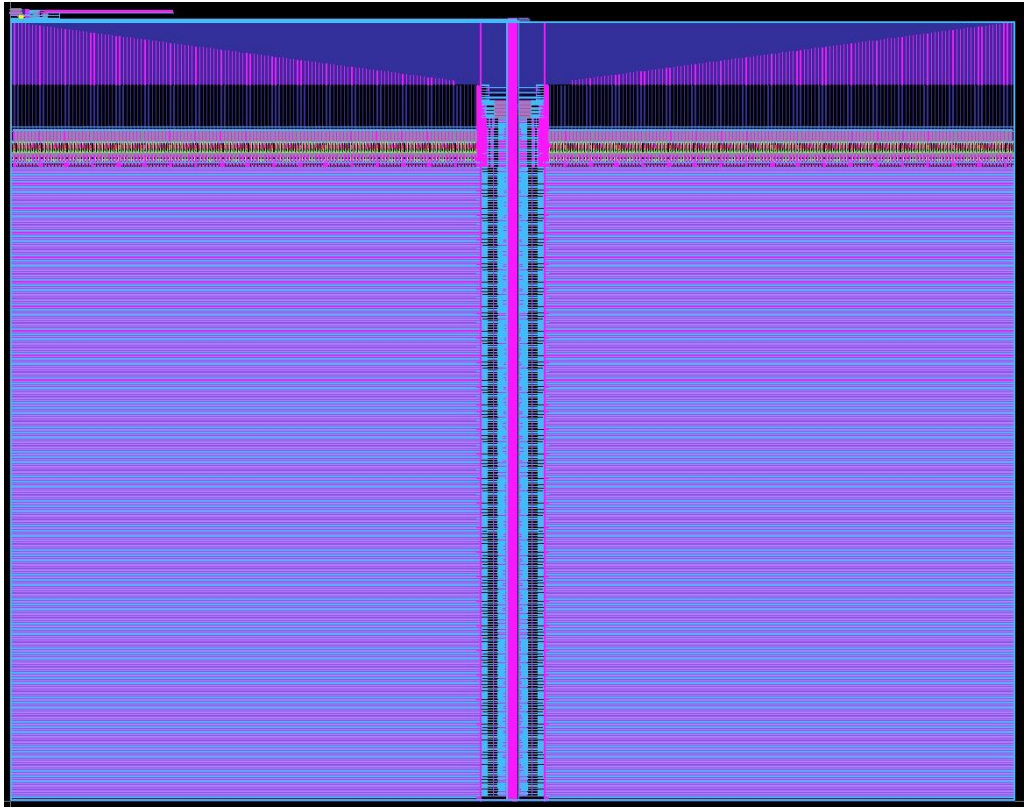
NYU

Prior Work

- Synthesized RTL
  - Low barrier of entry
  - Significant PPA + runtime penalty

Prior Work

- Synthesized RTL
  - Low barrier of entry
  - Significant PPA + runtime penalty
- OpenRAM
  - High density memory
  - Requires custom 6T bitcells + SPICE simulation for each PDK

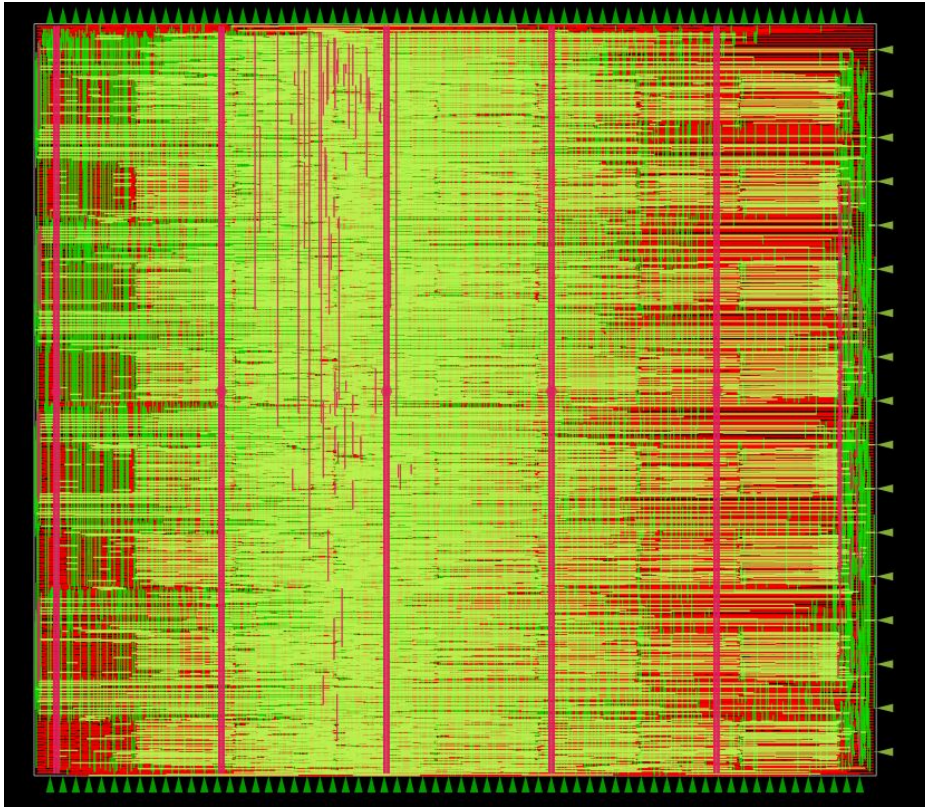


16KB SRAM from OpenRAM

NYU

Prior Work

- Synthesized RTL
  - Low barrier of entry
  - Significant PPA + runtime penalty
- OpenRAM
  - High density memory
  - Requires custom 6T bitcells + SPICE simulation for each PDK
- DFFRAM
  - Standard cell based
  - No custom placer (uses LibreLane)
  - Lacks important memory features

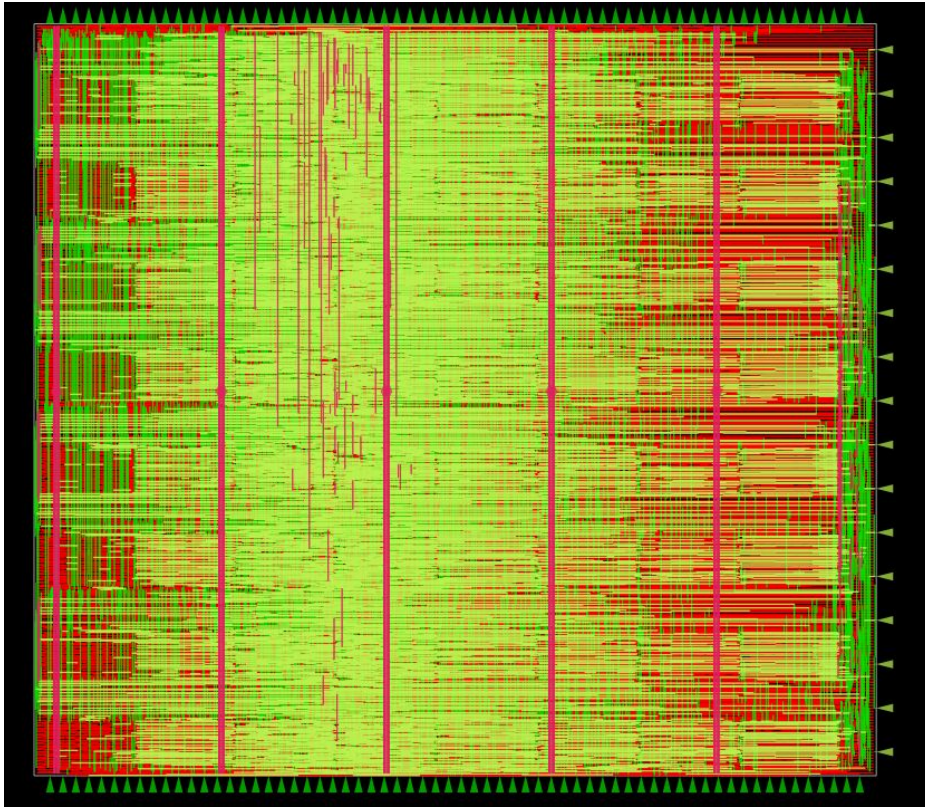


128x64 RAM from DFFRAM

NYU

Prior Work

- Synthesized RTL
  - Low barrier of entry
  - Significant PPA + runtime penalty
- OpenRAM
  - High density memory
  - Requires custom 6T bitcells + SPICE simulation for each PDK
- DFFRAM
  - Standard cell based
  - No custom placer (uses LibreLane)
  - Lacks important memory features
- Where ORRAM comes in



128x64 RAM from DFFRAM

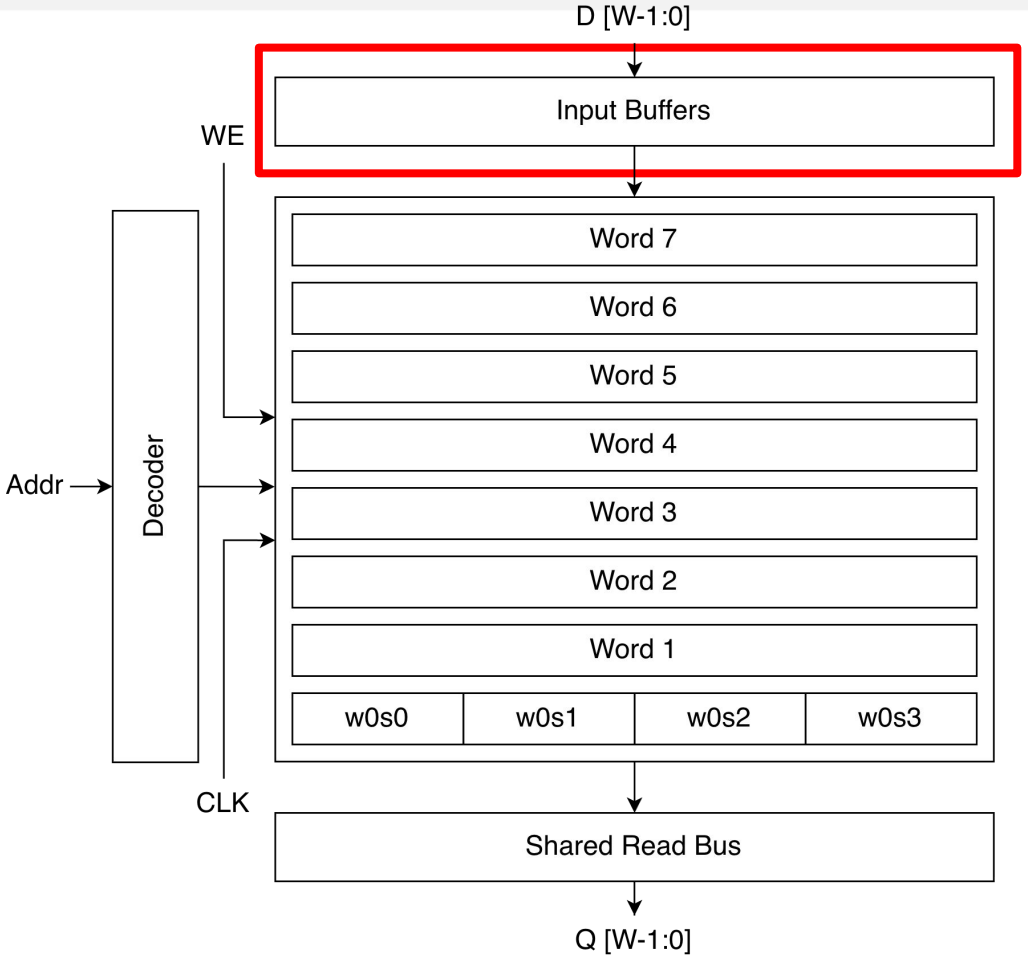
NYU

ORRAM Overview

- ORRAM Architecture
- ORRAM advanced features
- Comparison to prior works
- Conclusions & future work

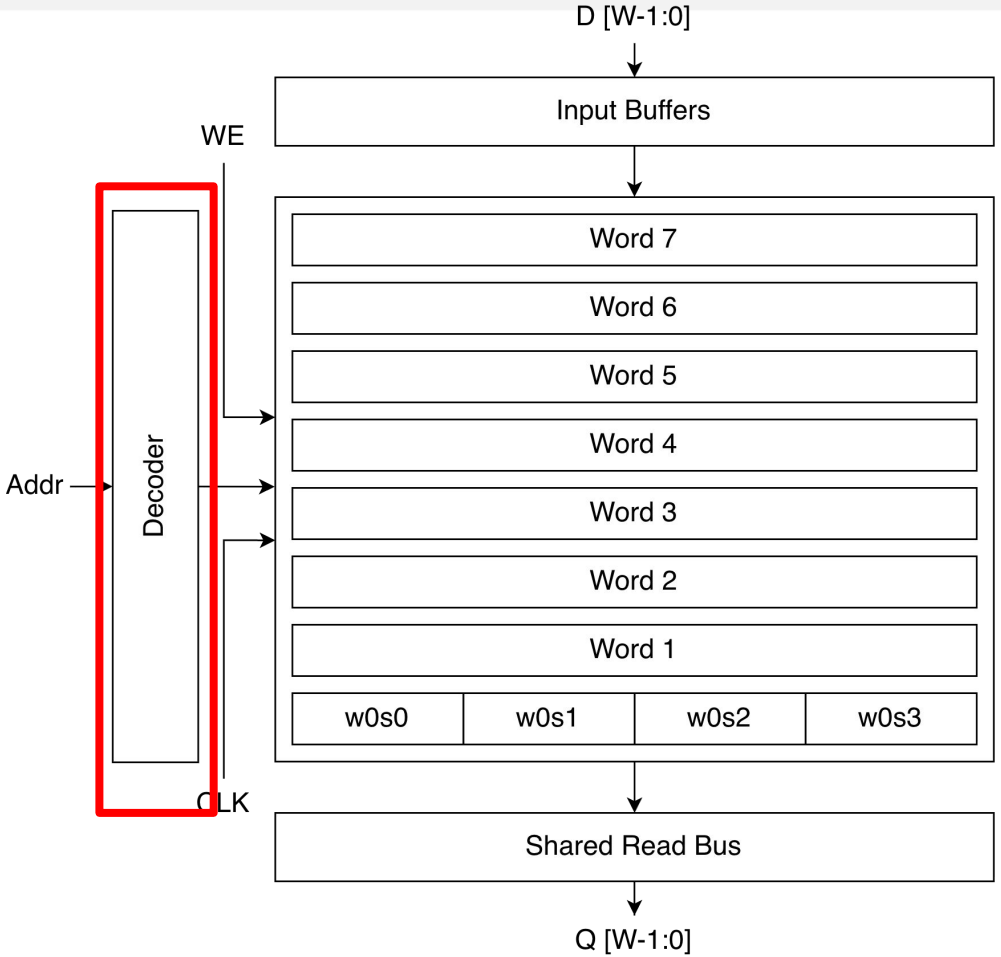
ORRAM Architecture

- Input buffer stage



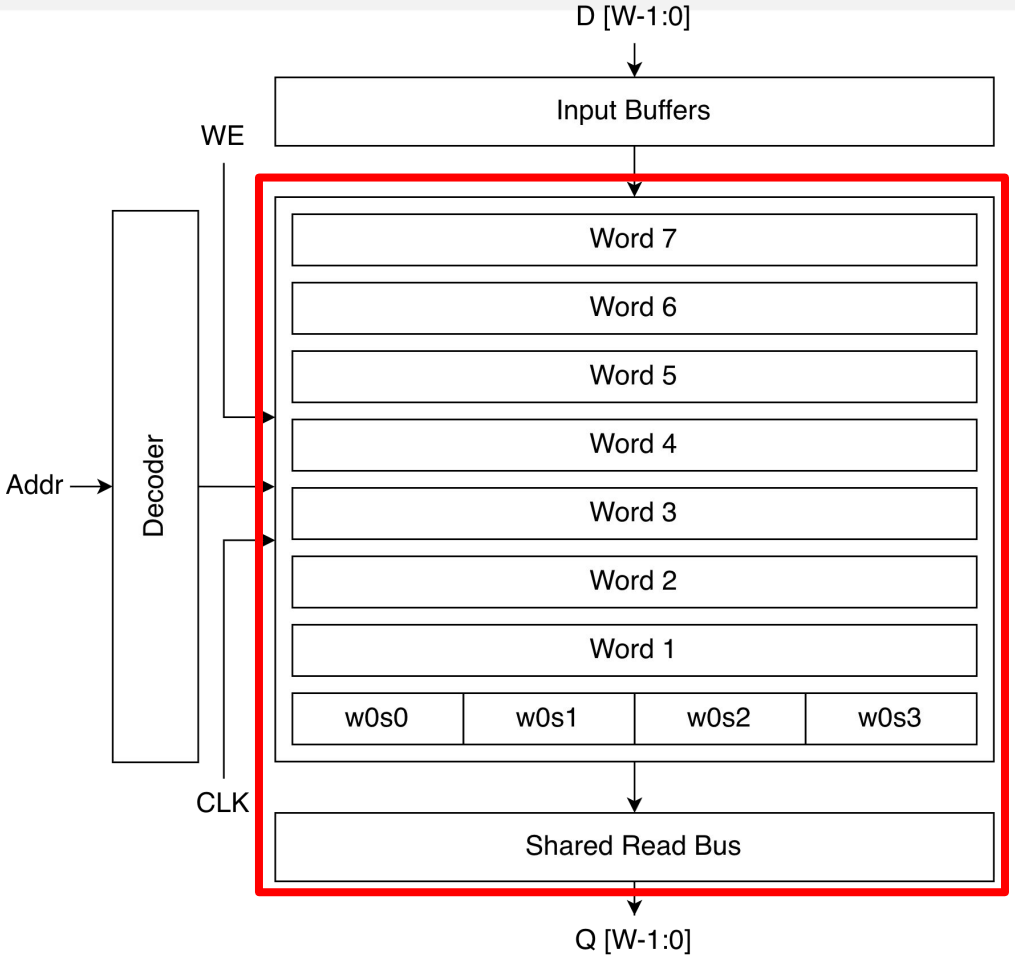
ORRAM Architecture

- Input buffer stage
- Per-port decoder generation
  - Per-word write enable
  - Per-port read select signals



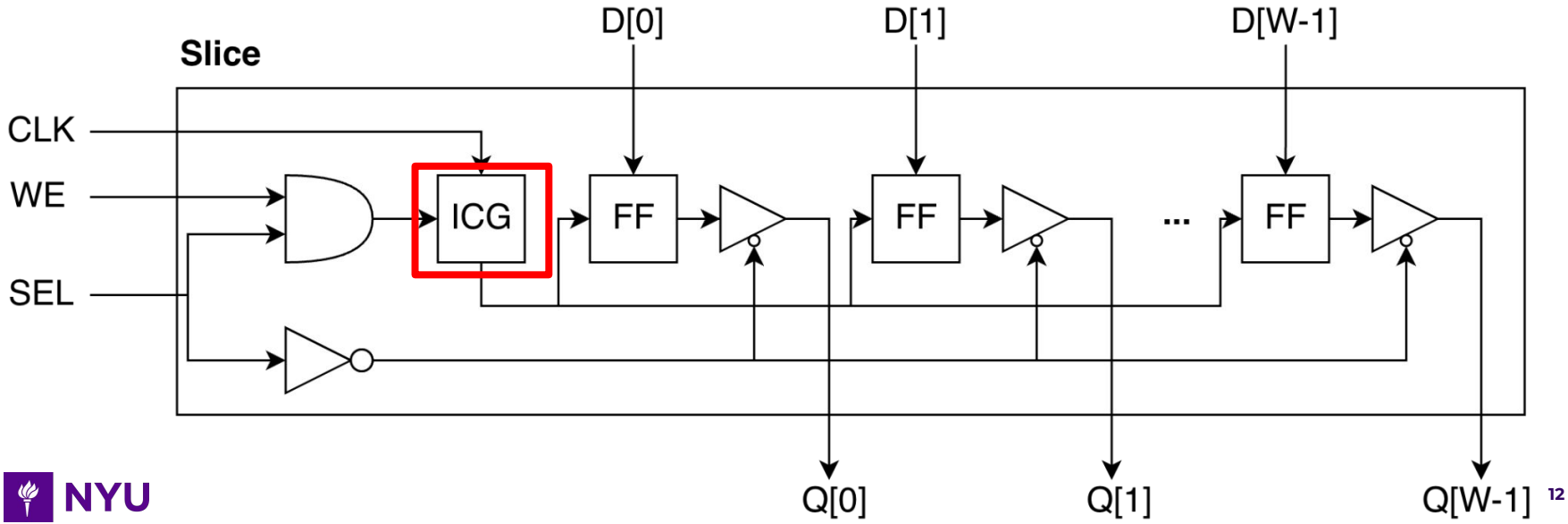
ORRAM Architecture

- Input buffer stage
- Per-port decoder generation
  - Per-word write enable
  - Per-port read select signals
- Output to shared read bus



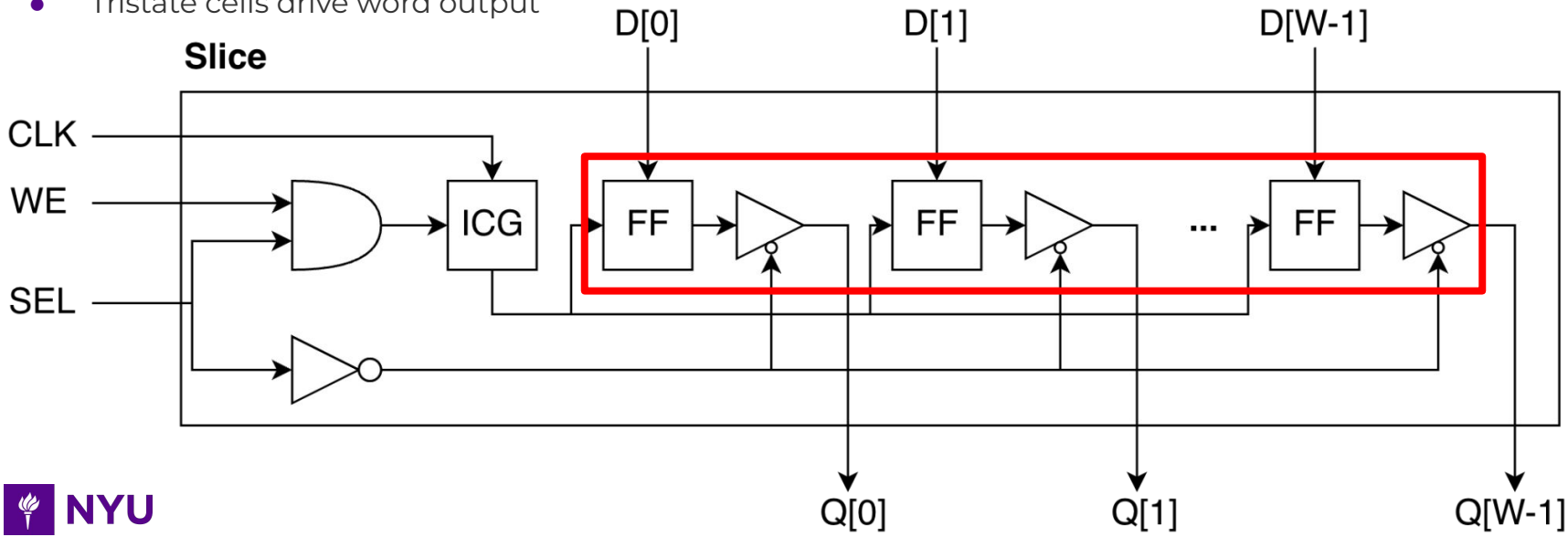
ORRAM Architecture

- Write-enable/port-select signals drive ICG

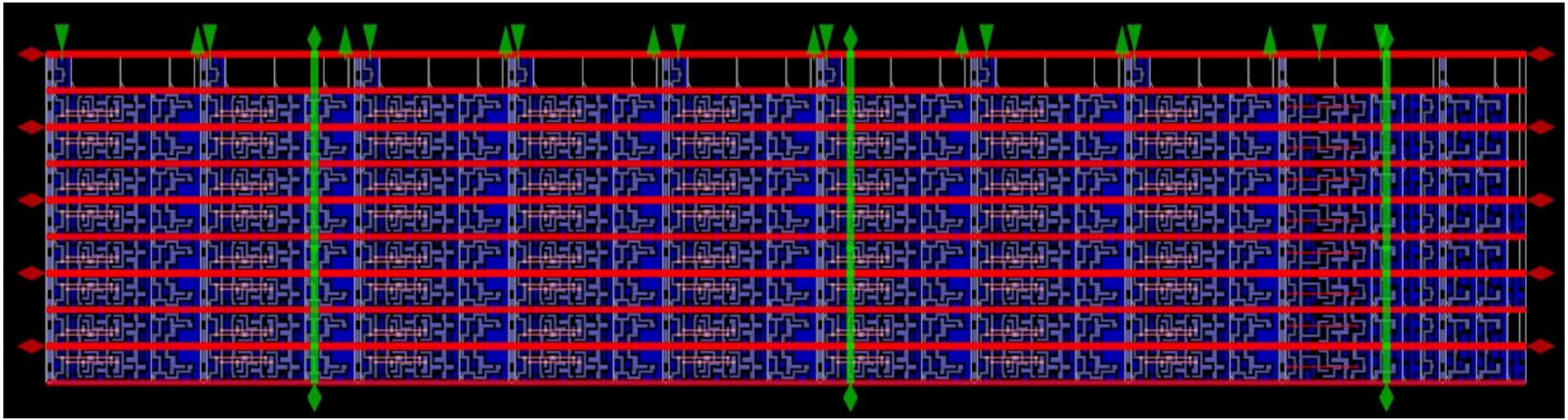


ORRAM Architecture

- Write-enable/port-select signals drive ICG
- Storage composed of flip-flops/latches
- Tristate cells drive word output



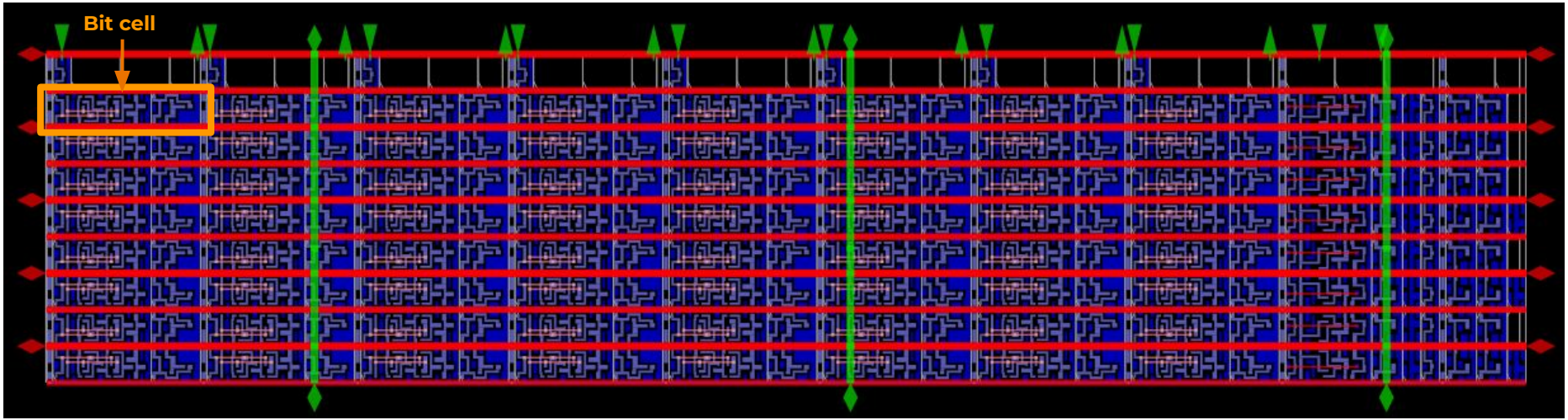
Cell Layout



8x8 RAM Cell Layout (default options)

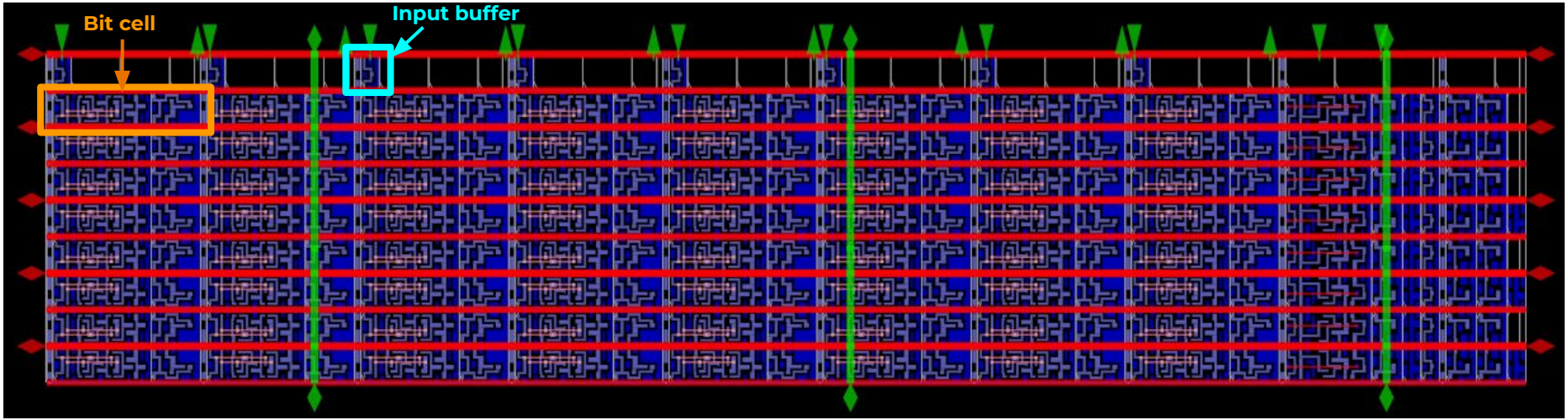
NYU

Cell Layout



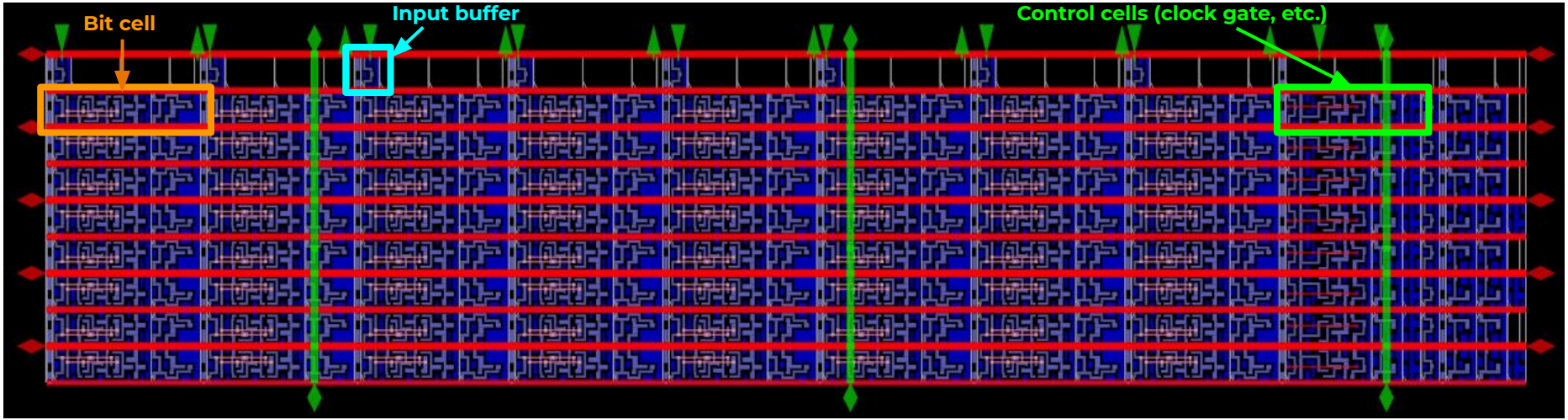
8x8 RAM Cell Layout (default options)

Cell Layout



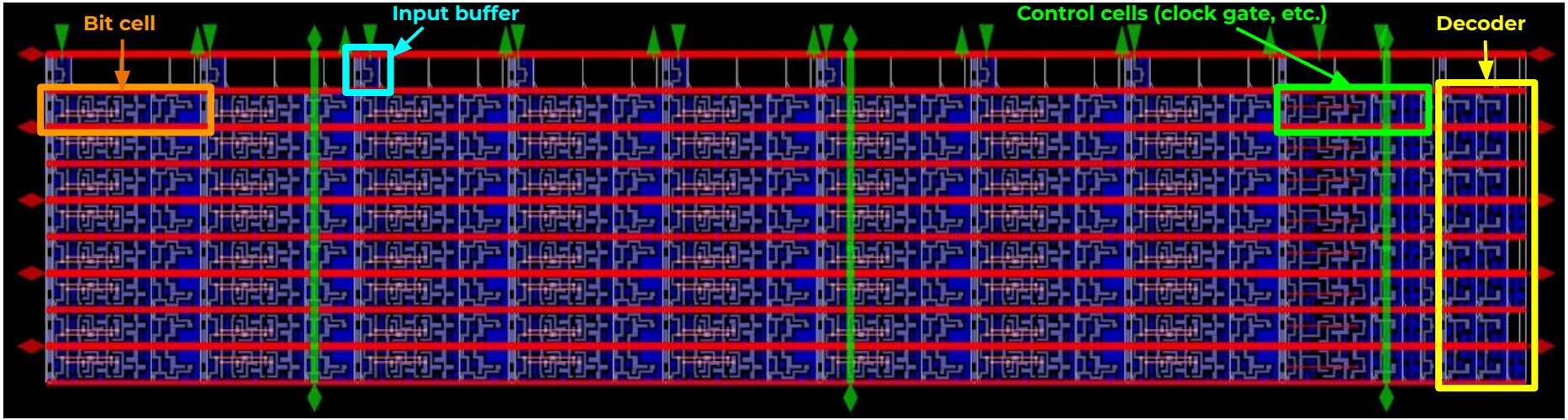
8x8 RAM Cell Layout (default options)

Cell Layout



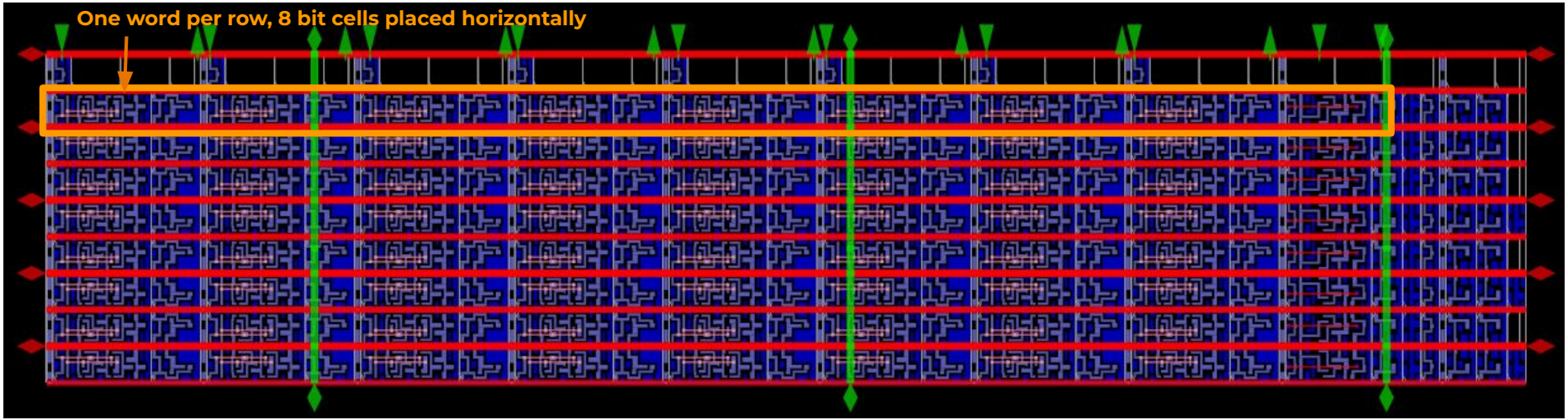
8x8 RAM Cell Layout (default options)

Cell Layout



8x8 RAM Cell Layout (default options)

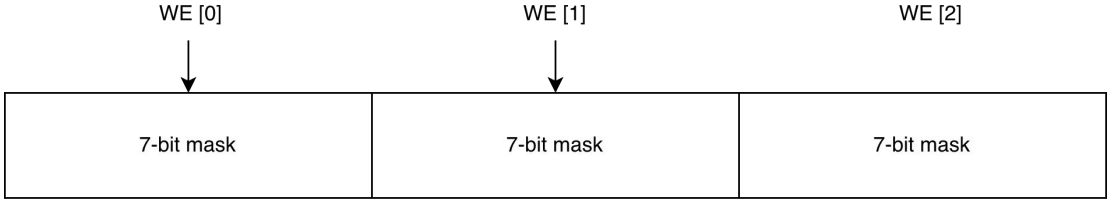
Cell Layout



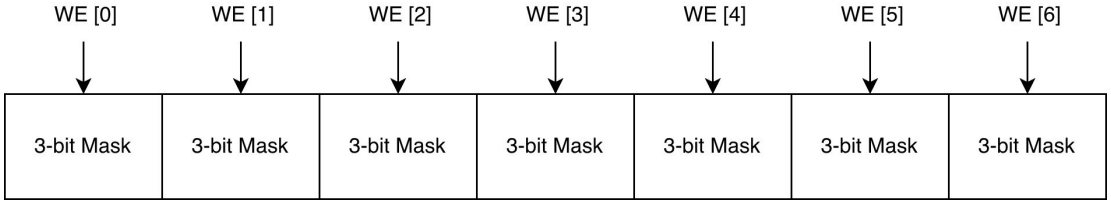
8x8 RAM Cell Layout (default options)

Arbitrary Masking

- Customizable memory
  - Word sizes + word counts
- Configurable write mask granularity
- Adaptable to design needs
  - Smaller register files
  - Larger memory modules



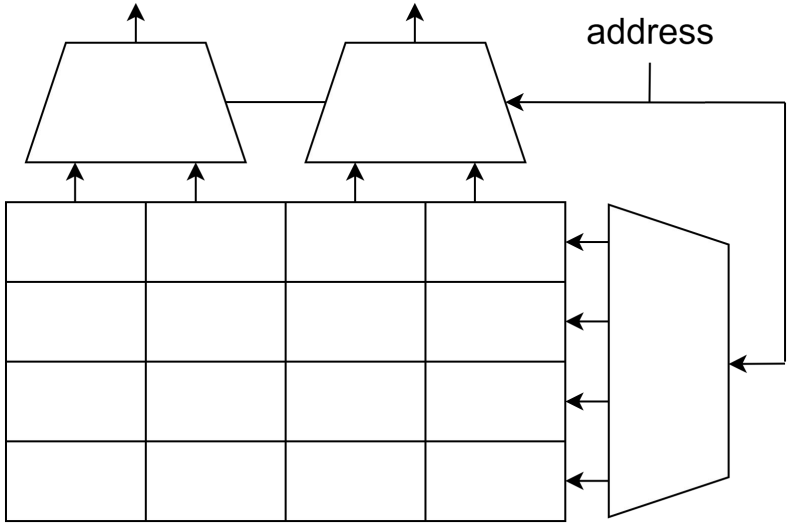
21-bit slice with 7-bit masks



21-bit slice with 3-bit masks

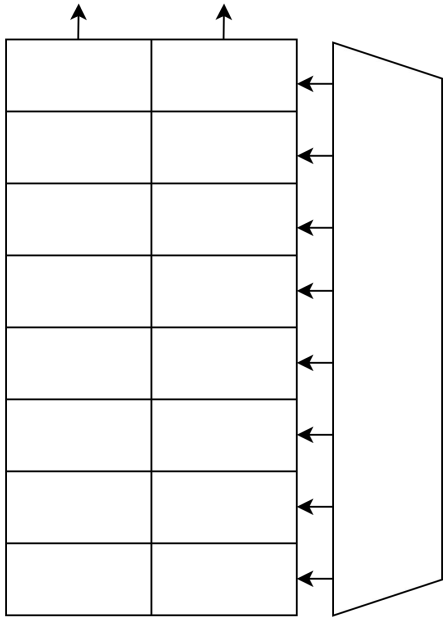
Column muxing

- Memory sizes can have bad aspect ratios
  - Results in long nets, high capacitance
  - Hard to place and route
- ***column_mux_ratio*** = 1, 2, or 4
  - word(s) placed in a single physical row
  - adjust aspect ratio by this specified ratio
- Lower ***log2(column_mux_ratio)*** bits select the word
- Remaining upper bit select the physical row

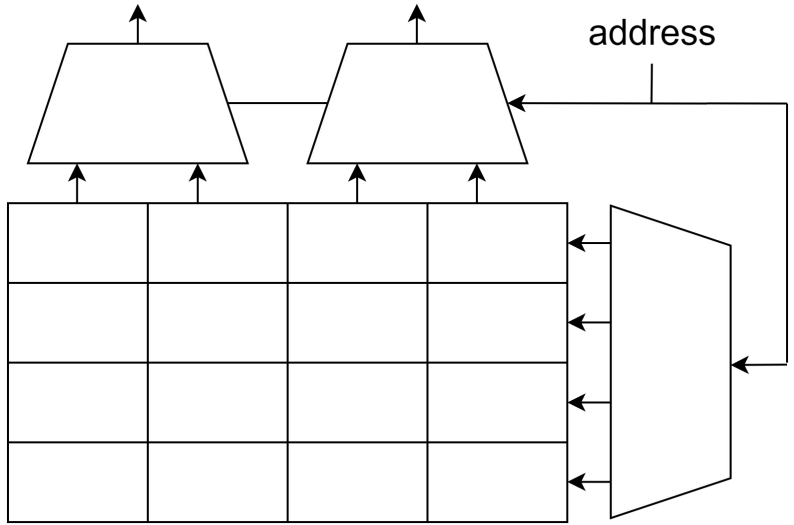


8x2 RAM (with ***column_mux_ratio*** =2)

Column muxing



8x2 RAM (no column muxing)

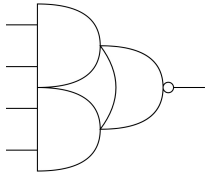


8x2 RAM (with ***column_mux_ratio*** =2)

NYU

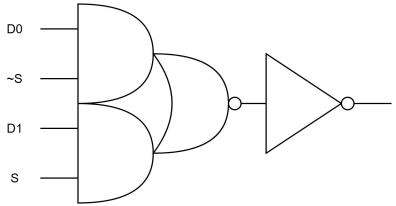
Column muxing

- Muxes constructed using AOI22 (And-Or-Inverter) gates



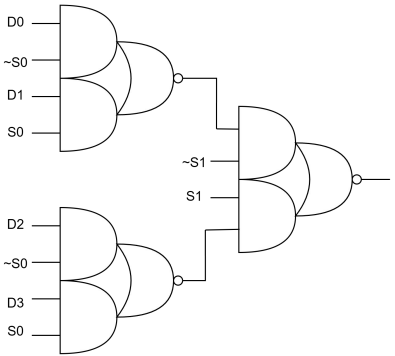
Standard AOI22 gate

- For 2:1 multiplexer used in ***column_mux_ratio = 2***
  - one AOI22 and one inverter in series



2:1 Multiplexer

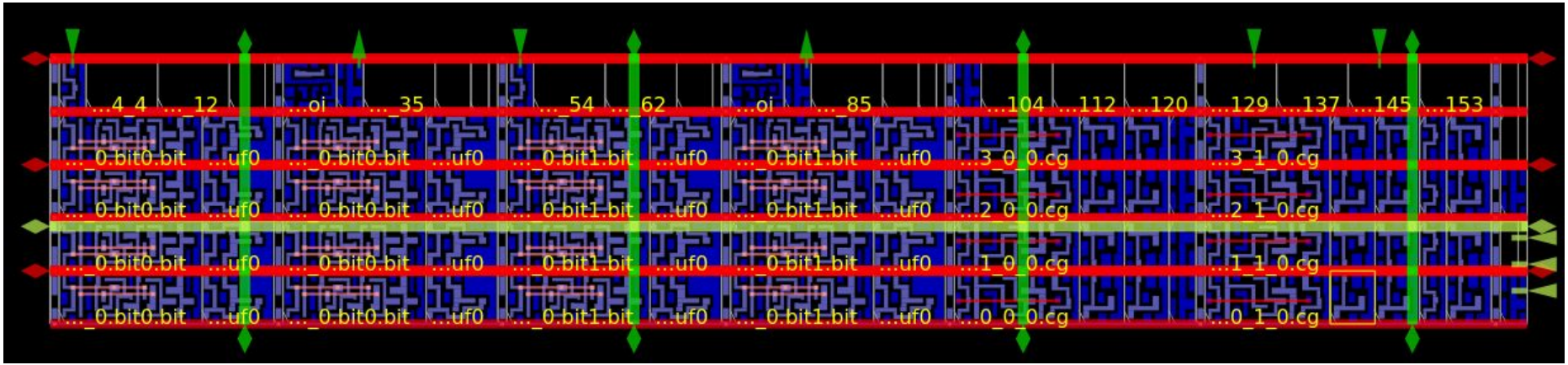
- For 4:1 multiplexer used ***column_mux_ratio = 4***:
  - two stage-one AOI22 into one stage-two AOI22 gate



4:1 Multiplexer

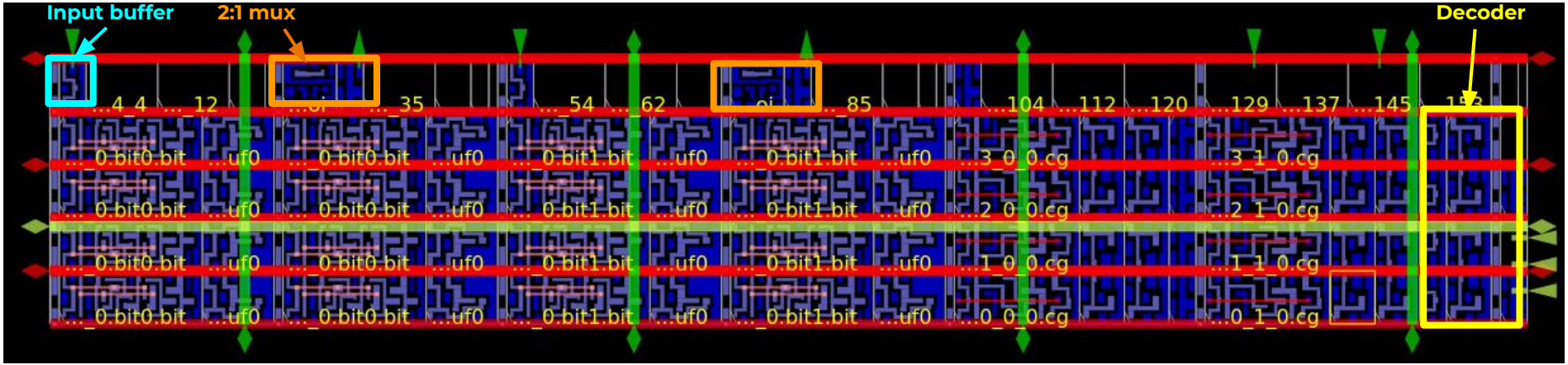
NYU

Column muxing

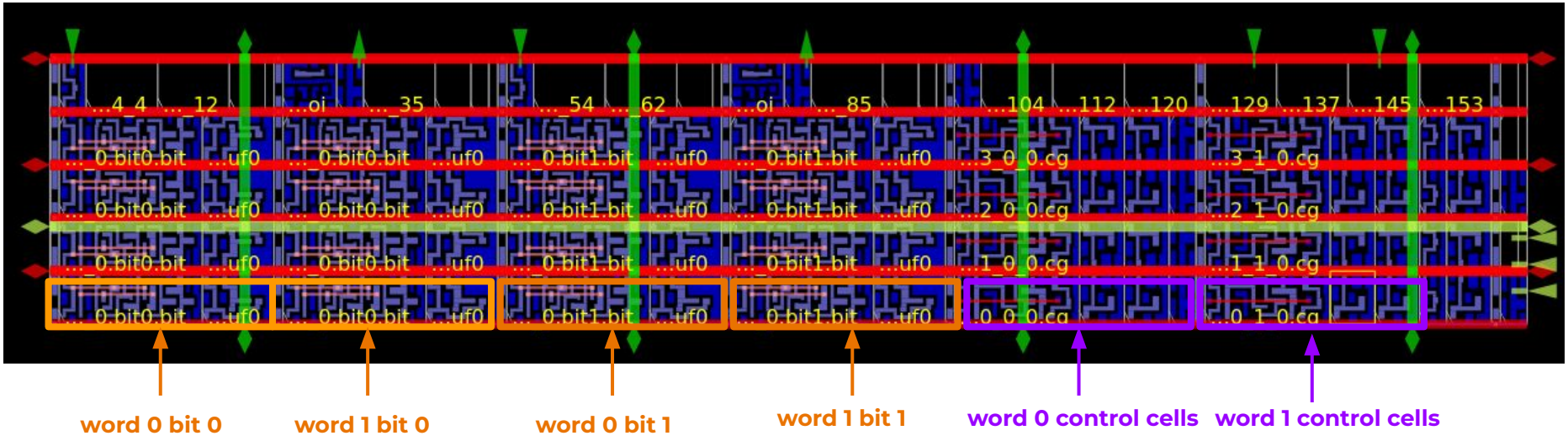


8x2 RAM Cell Layout (with ***column_mux_ratio*** =2)

Column muxing



Column muxing



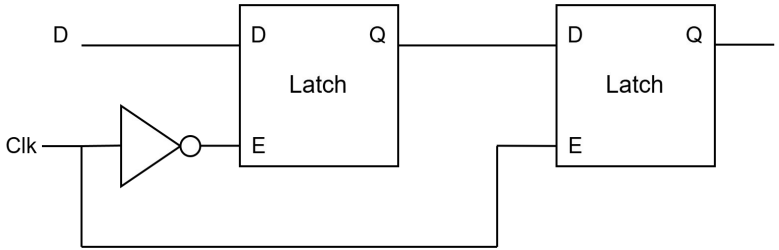
Bits are placed interleaved to reduce net lengths and allow more compact cell placement

NYU

Latch Memory

- Latches are compact, faster, power efficient vs. FF

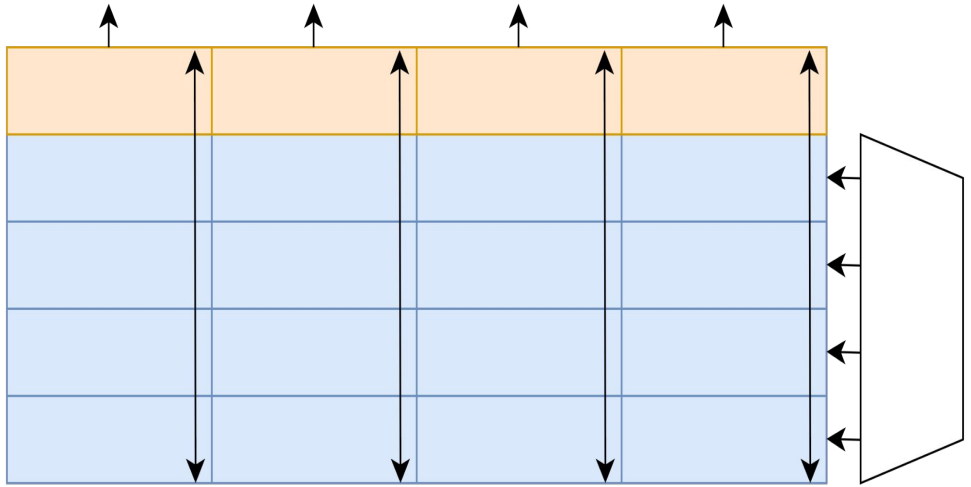
- Two-phase latch setup to capture data:
  - Split FFs into negative and positive latches
  - Share negative latch among all words
  - Same functionality as positive flip-flops



DFF using separate latches

Latch Memory

- Two-phase latch implementation:
  - Row of negative latches added
  - Storage cells are positive latches
  - Vertical net connects each column

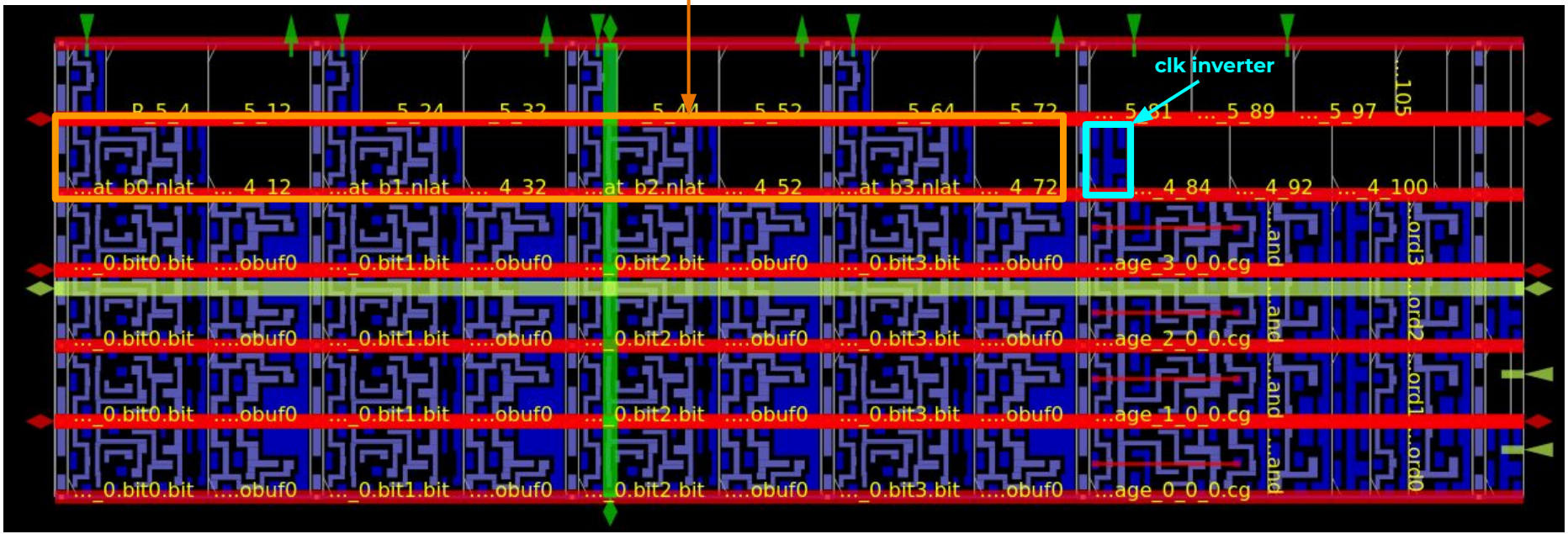


4x4 latch RAM

** Orange: negative latch; Blue: Positive latch*

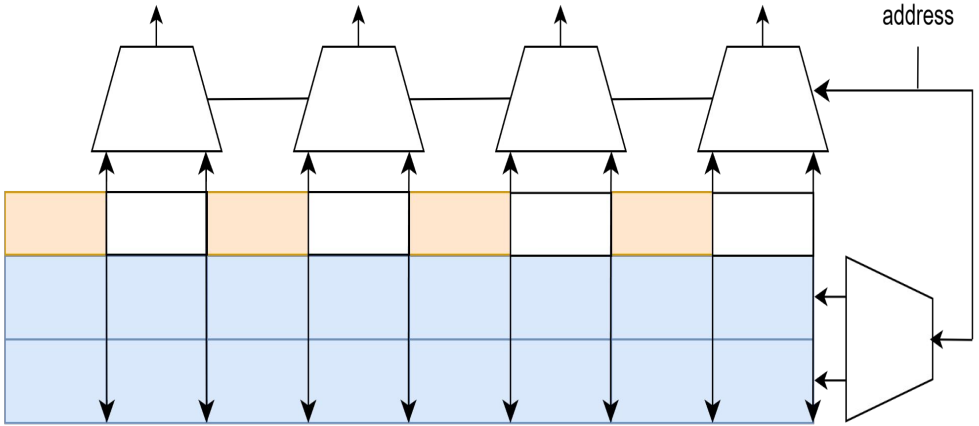
NYU

Latch Memory



Latch Memory

- Latch-based support column muxing
- Support all options available in FF version
- Identical functionality to FF version
  - asynchronous read
  - synchronous write

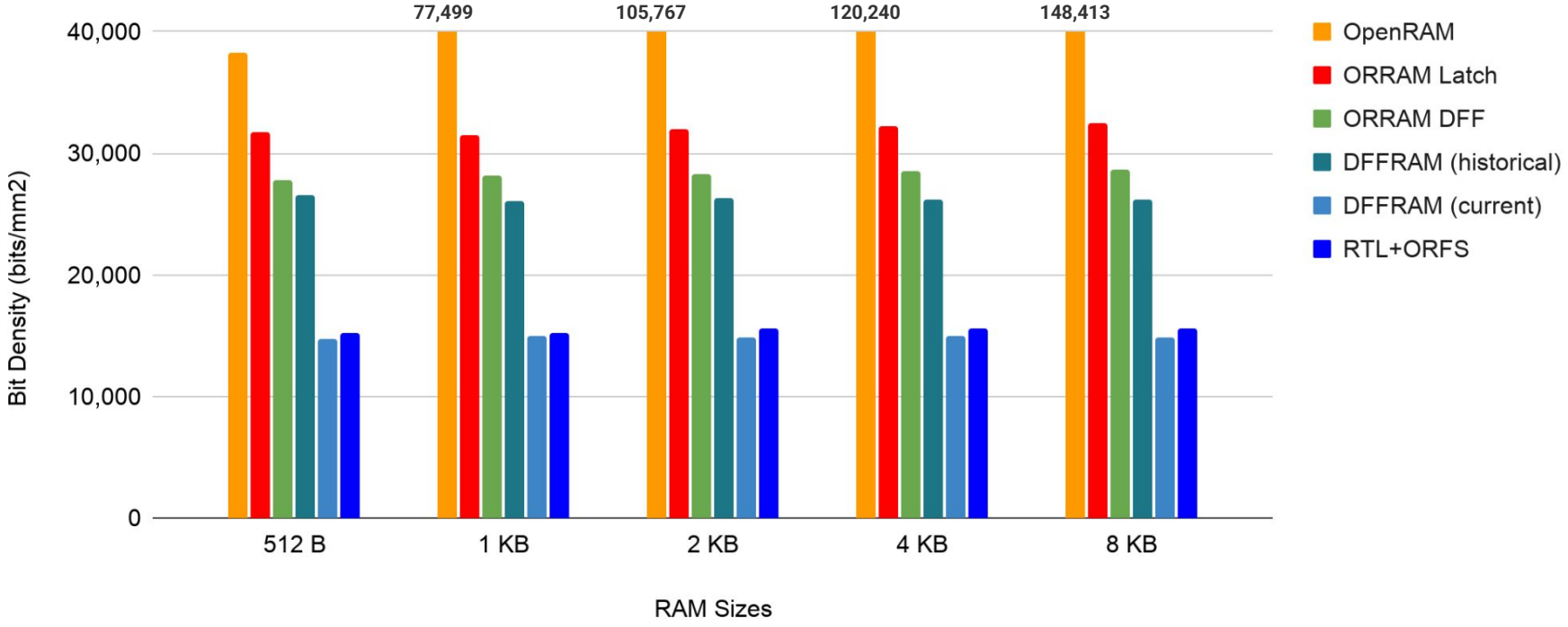


4x4 latch RAM (with ***column_mux_ratio*** =2)

** Orange: negative latch; Blue: Positive latch*

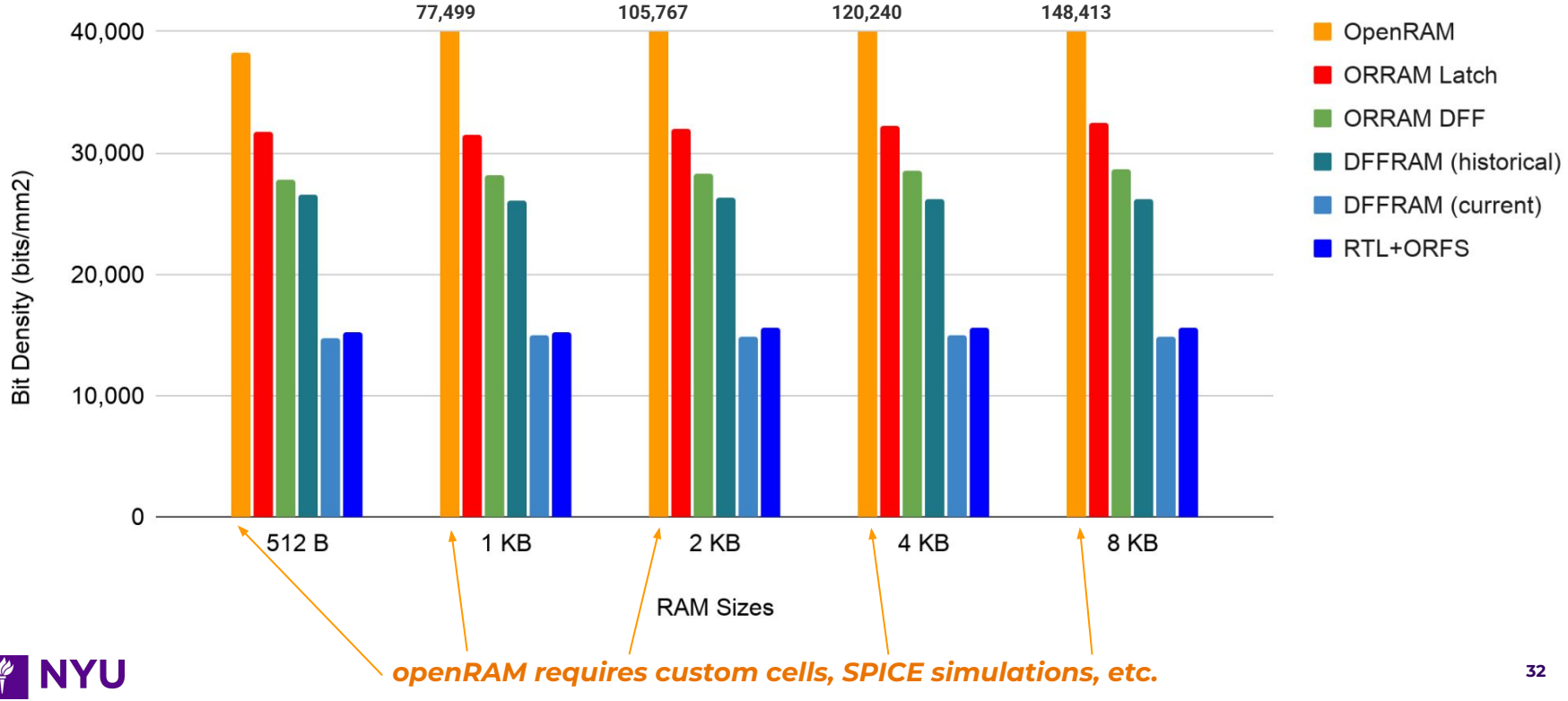
NYU

Results

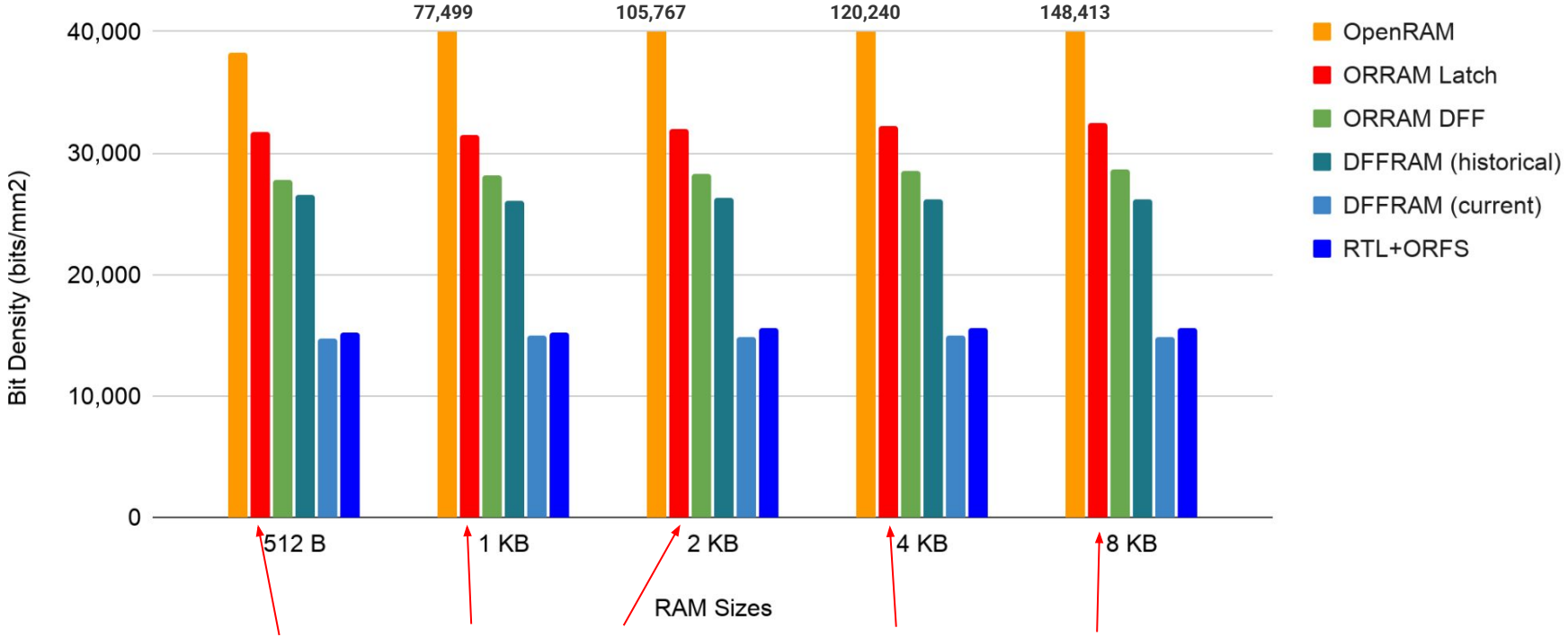

77,499
105,767
120,240
148,413
40,000
30,000
20,000
10,000
0
Bit Density (bits/mm2)
512 B
1 KB
2 KB
4 KB
8 KB
RAM Sizes
OpenRAM
ORRAM Latch
ORRAM DFF
DFFRAM (historical)
DFFRAM (current)
RTL+ORFS

NYU

Results


77,499
105,767
120,240
148,413
40,000
30,000
20,000
10,000
0
Bit Density (bits/mm2)
512 B
1 KB
2 KB
4 KB
8 KB
RAM Sizes
OpenRAM
ORRAM Latch
ORRAM DFF
DFFRAM (historical)
DFFRAM (current)
RTL+ORFS
openRAM requires custom cells, SPICE simulations, etc.

Results



ORRAM achieves best density among standard-cell based generators

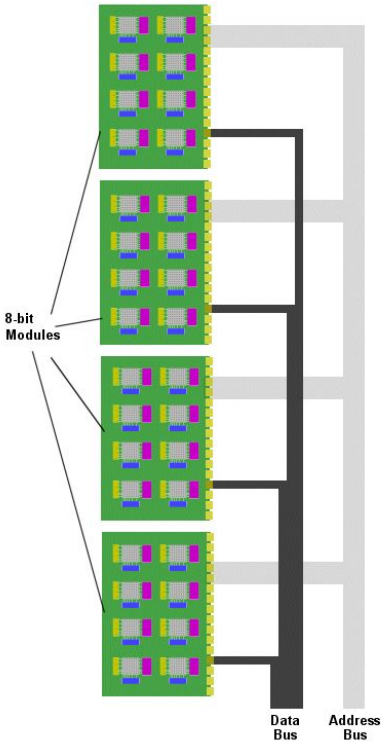
NYU

Takeaways

- Best in class for standard-cell based DFF/Latch memory (~28k/31k bits/mm^2)
- Requires only widely available PDK standard cells
- Native integration within OpenROAD
- Supports arbitrary porting, column muxing, and masking

Future Work

- Integrating timing analysis to perform timing and power optimization
- Banking support for large-scale RAMs
- Automatic RAM inference during synthesis



NYU

ORRAM: An OpenROAD-Integrated RAM Generator Using Standard Cells

Brayden Louie*, Thinh Nguyen*, Matt Liberty, Austin Rovinski*

*New York University, Precision Innovations, Inc.

ORRAM Repository

Results

Bit Density (bits/mm^2) across open-source memory generators

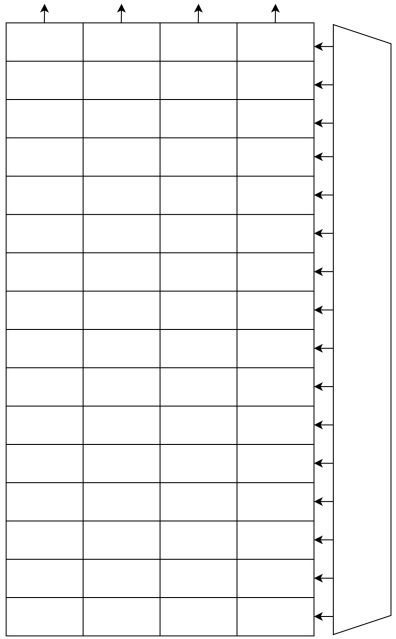
Size	**ORRAM Latch**	**ORRAM DFF**	DFFRAM (historical)	DFFRAM (current)	OpenRAM	RTL + ORFS
512 B	**31,795**	**27,734**	26,557	14,691	38,299	15,196
1 KB	**31,552**	**28,168**	26,027	14,951	77,499	15,250
2 KB	**31,977**	**28,339**	26,297	14,817	105,767	15,639
4 KB	**32,284**	**28,586**	26,196	14,948	120,240	15,566
8 KB	**32,508**	**28,677**	26,229	14,879	148,413	15,649

Example Script

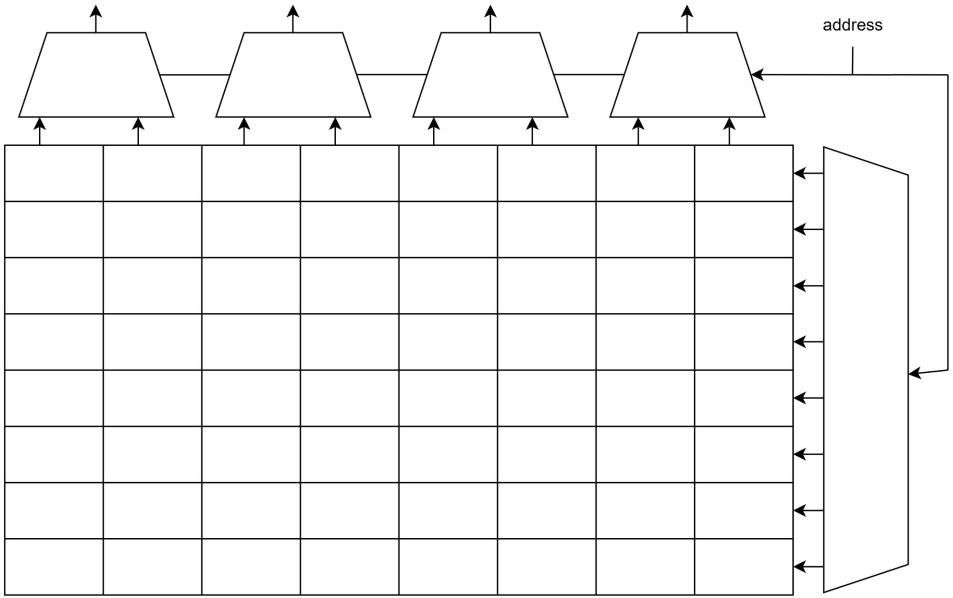
```
read_liberty
sky130hd/sky130_fd_sc_hd__tt_025C_1v80.lib

read_lef sky130hd/sky130hd.tlef

read_lef sky130hd/sky130_fd_sc_hd_merged.lef

set behavioral_file [make_result_file
make_8x8_behavioral.v]
```

```
generate_ram \ -mask_size 8 \ -word_size 8 \ -num_words 8 \
 -rw_ports 1 \ -storage_cell sky130_fd_sc_hd__dfxtp_1 \
 -routing_layer {met1 0.48} \ -ver_layer {met2 0.48 40} \
 -hor_layer {met3 0.48 20} \
 -filler_cells {sky130_fd_sc_hd__fill_1 sky130_fd_sc_hd__fill_2 \
  sky130_fd_sc_hd__fill_4 sky130_fd_sc_hd__fill_8} \
 -tapcell sky130_fd_sc_hd__tap_1 \
 -max_tap_dist 15 \
 -write_behavioral_verilog $behavioral_file
```

Column muxing



16x4 RAM (no column muxing)



16x4 RAM (with ***column_mux_ratio*** =2)