# A Digital Flow for Asynchronous VLSI Systems: Status Update

Udit Agarwal[1], Samira Ataei[2], Jiayuan He[1], Wenmian Hua[2], **Yi-Shan Lu[1]**, Sepideh Maleki[1], Yihang Yang[2], Keshav Pingali[1], Rajit Manohar[2]

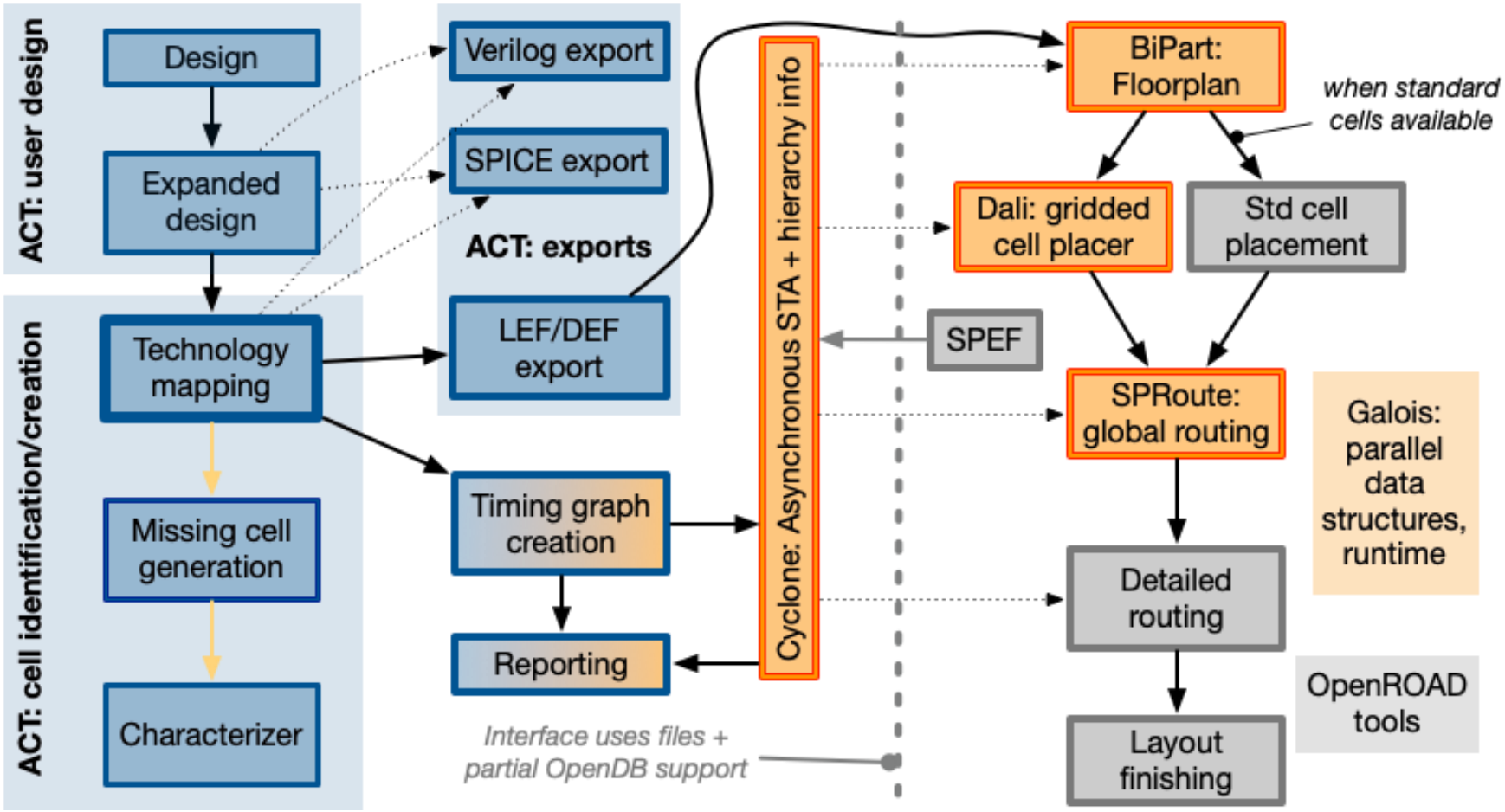
[1]University of Texas at Austin, [2]Yale University

November 5, 2020 at WOSET 2020


# A little bit about me…

- Yi-Shan Lu
  - PhD student at CS, UT Austin
  - Advisor: Prof. Keshav Pingali
- Research interests
  - Parallelization & language design for domain-specific computation
  - Current focus: EDA algorithms, timing analysis & simulation
- Selected honors
  - Graph Challenge Champion, HPEC 2017
  - Third Place Award, TAU Contest 2019
  - Second Place, CADathlon at ICCAD 2019
  - Participation Award, TAU Contest 2020

# Asynchronous design flow at a glance





[1] W. Hua, Y.-S. Lu, K. Pingali, R. Manohar. Cyclone: A static timing and power analysis engine for asynchronous circuits. In *ASYNC* 2020.
[2] Y. Yang, J. He, R. Manohar. Dali: A gridded cell placement flow. In *ICCAD* 2020.
[3] J. He, M. Burtscher, R. Manohar, K. Pingali. SPRoute: A scalable parallel negotiation-based global router. In *ICCAD* 2019.

# Updates

- Cyclone [1]
  - Asynchronous timer & power analyzer
- BiPart
  - Deterministic parallel hypergraph partitioner
- Dali [2]
  - A gridded cell placer
- AMC [4]
  - Asynchronous memory compiler

[1] W. Hua, Y.-S. Lu, K. Pingali, R. Manohar. Cyclone: A static timing and power analysis engine for asynchronous circuits. In *ASYNC* 2020.
[2] Y. Yang, J. He, R. Manohar. Dali: A gridded cell placement flow. In *ICCAD* 2020.
[4] S. Ataei, R. Manohar. AMC: An asynchronous memory compiler. In *ASYNC* 2019.

# Cyclone [1]: Comprehensive async timing & power analyzer

- Need to analyze cycles explicitly for asynchronous circuits
- Functionality enhancement
  - Supports QDI circuits with data & bundled-data logic timing constraints
  - Power analysis integrated with timer
- Performance improvement
  - Faster timing graph creation by exploiting module hierarchy
  - Effectively parallelized using Galois
    - Steady slew & delay computation
    - Longest-path forest construction in critical cycle ratio algorithm
    - Timing propagation
    - Timing constraint checking

[1] W. Hua, Y.-S. Lu, K. Pingali, R. Manohar. Cyclone: A static timing and power analysis engine for asynchronous circuits. In *ASYNC* 2020.

# Cyclone [1]: Performance on large circuits

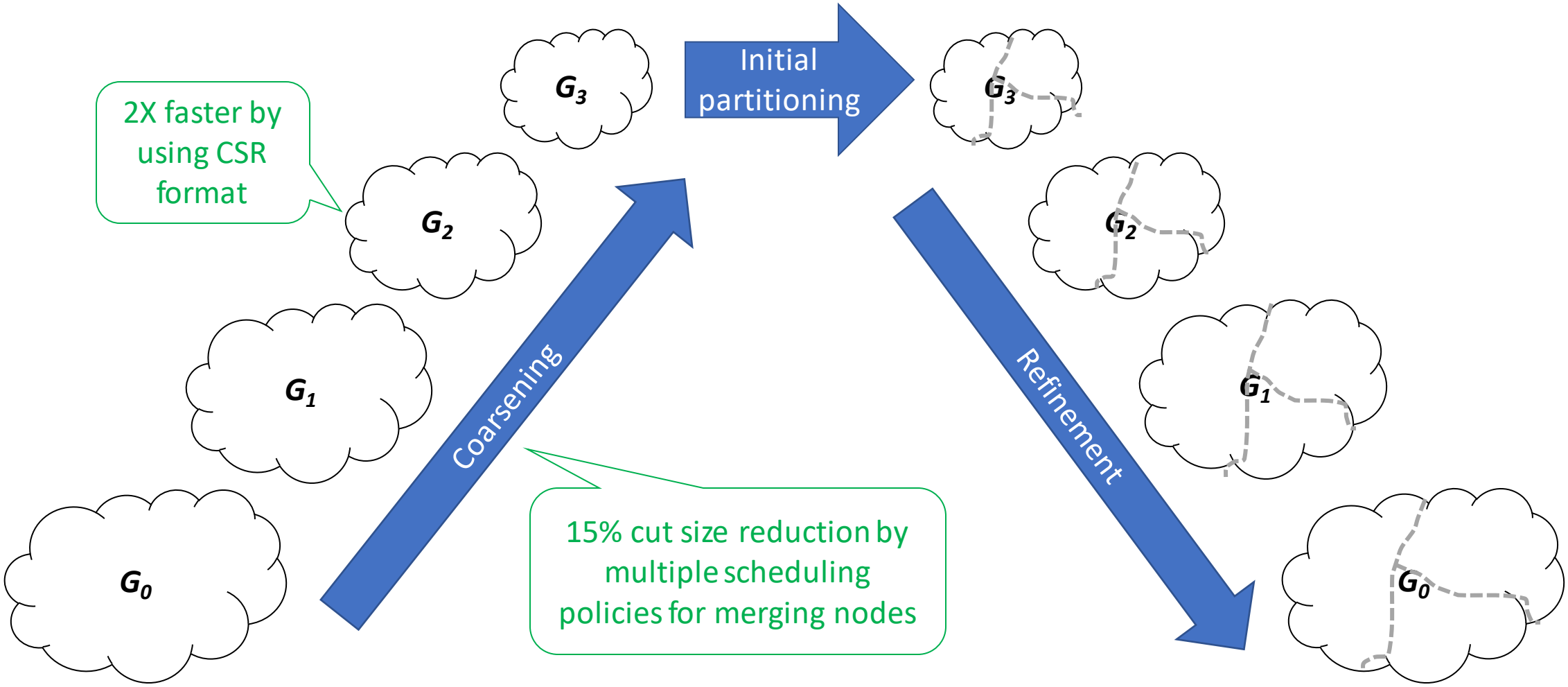
| Selected circuits from TAU 2015 benchmark suites | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Circuit properties | | | | | Max. cycle ratio | | Full performance analysis | | |
| Name | # pins | p* (ns) | M | Power (mW) | YTO (s) (#t) | CPLEX (s) | Seq (s) | Best (s) (#t) | X |
| bd203 | 495 | 0.443 | 1 | 0.521 | 0.010 (01) | 0.010 | 0.017 | 0.017 (01) | 1.00 |
| s5387 | 88,292 | 4.388 | 3 | 22.602 | 0.969 (14) | 3.090 | 9.039 | 1.937 (28) | 4.67 |
| ac97_ctrl | 650,709 | 3.785 | 3 | 190.356 | 8.486 (21) | 60.390 | 102.820 | 16.594 (28) | 6.20 |
| vga_lcd | 5,689,435 | 7.046 | 1 | 911.437 | 100.112 (49) | 2,267.920 | 2,889.255 | 145.180 (56) | 19.90 |

Faster performance characterization by better cycle ratio algorithm

6-20X self speedup through parallelization for large designs

[1] W. Hua, Y.-S. Lu, K. Pingali, R. Manohar. Cyclone: A static timing and power analysis engine for asynchronous circuits. In *ASYNC* 2020.

# BiPart:
# Deterministic parallel hypergraph partitioner

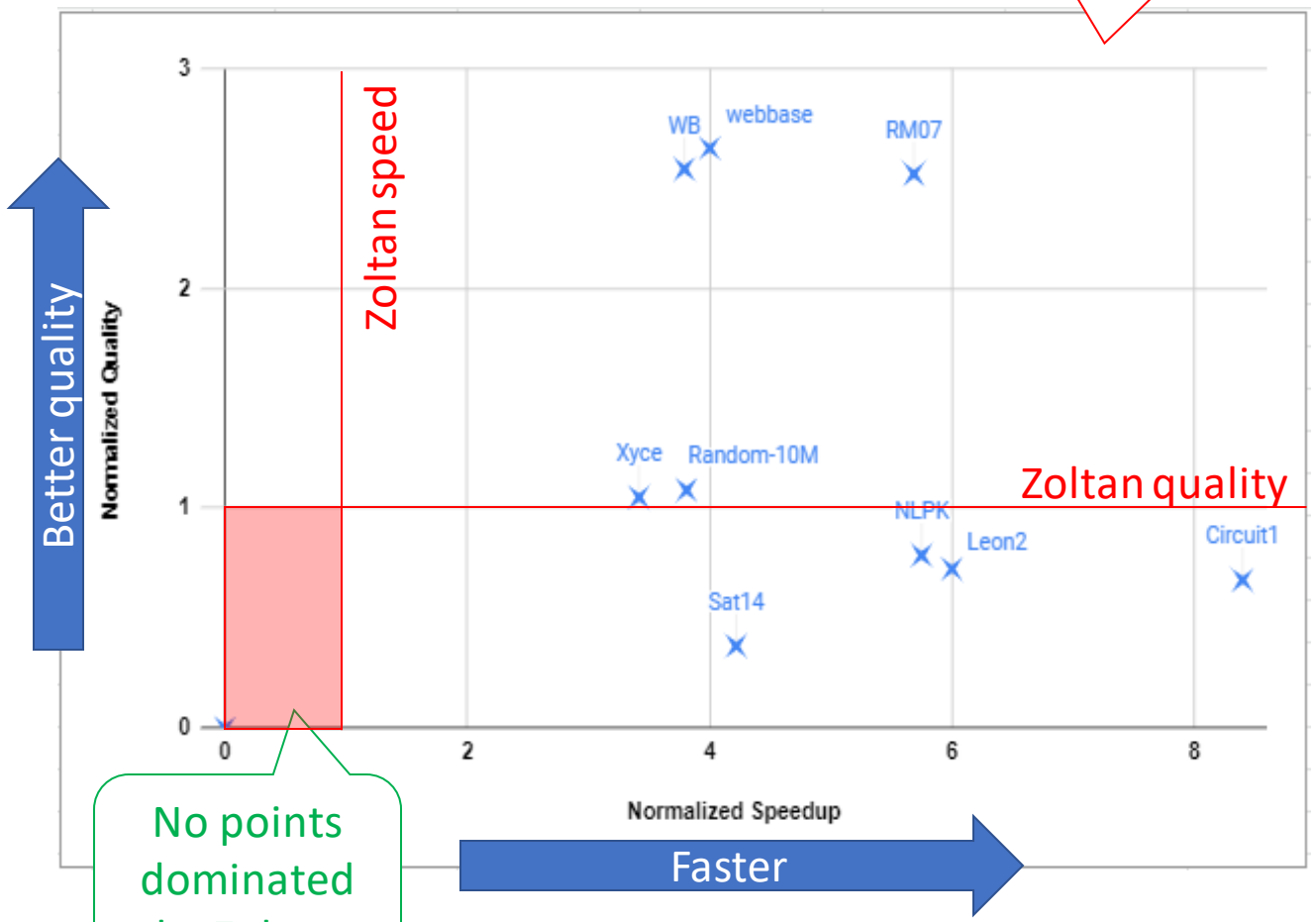


# BiPart: Comparison w/ Zoltan

| Graph | # Nodes | # Hedges | # Edges | BiPart (1) (sec) | BiPart (14) (sec) |
|---|---|---|---|---|---|
| Random-15M | 15.0M | 17.0M | 280.6M | 431.9 | 64.85 |
| Random-10M | 10.0M | 15.0M | 115.0M | 198.6 | 35.07 |
| WB | 9.8M | 6.9M | 57.2M | 20.89 | 7.32 |
| NLPK | 3.5M | 3.5M | 96.8M | 17.49 | 5.88 |
| Xyce | 1.9M | 1.9M | 9.5M | 3.20 | 0.94 |
| Circuit1 | 1.9M | 1.9M | 8.9M | 2.90 | 0.98 |
| Leon2 | 62.7K | 1.7M | 6.8M | 1.89 | 1.14 |
| webbase | 1.1M | 800.8K | 2.4M | 0.67 | 0.46 |
| Sat14 | 1.0M | 1.0M | 3.1M | 58.76 | 9.90 |
| RM07 | 381.7K | 381.7K | 37.5M | 3.08 | 0.92 |

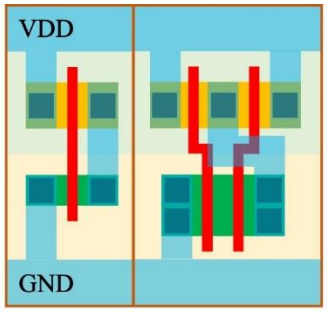
6-7X self speedup

Zoltan cannot finish Random-15M (OoM)

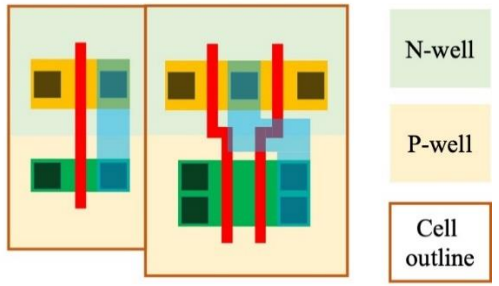



No points dominated by Zoltan

# Dali [2]: A gridded cell placer




Standard cell




Gridded cell

**Dali: A gridded cell placement flow**

Core problem: weighted wire-length optimization

Design → Global placement → Forward-backward legalization → Well legalization → Power grid design → Layout

Existing techniques
- Model wire-length as a quadratic function
- Obtain a rough placement

- Removes cell overlaps
- Avoid large cell displacement

- Create mini-rows
- Clean design rule violations related to N/P-wells
- Create N/P-wells & place well tap cells

- Connect VDD/GND pins to power supply

[2] Y. Yang, J. He, R. Manohar. Dali: A gridded cell placement flow. In *ICCAD* 2020.

# Dali [2]: Comparison to standard-cell methodology

**Placement region treatment**

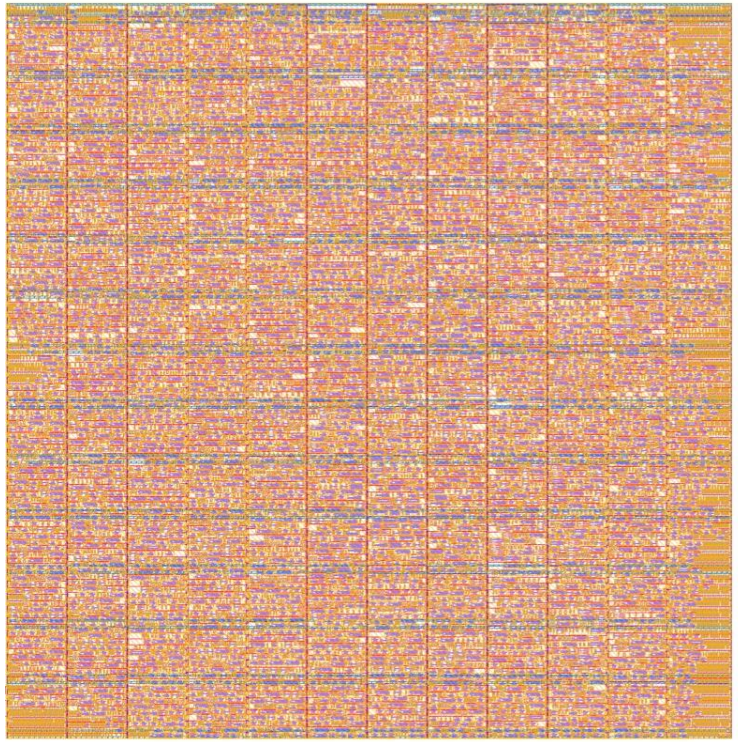
| Properties | Standard cell | Gridded cell |
|---|---|---|
| Rows in the placement region | Predefined/Static | Dynamic |
| N/P-well | Preplaced | Mini-row-based |
| Well-tap cells | Preplaced | Mini-row-based |

**Placement flow**

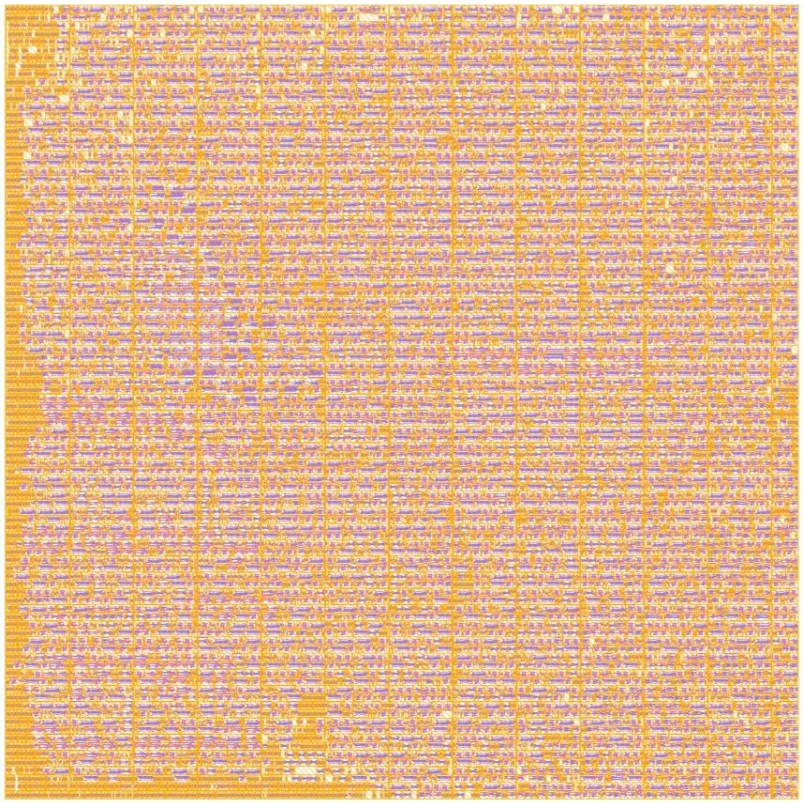
| Placement stage | Standard cell placer | Dali |
|---|---|---|
| Global placement | Yes | Yes |
| Detailed placement | Yes | Within mini-row after WL |
| Legalization | Align to rows | Align to routing grid |
| Well-legalization | Implicit by abutment/filler cells | Mini-row construction |
| Power routing | Implicit by abutment/filler cells | Placement-based |

[2] Y. Yang, J. He, R. Manohar. Dali: A gridded cell placement flow. In *ICCAD* 2020.

# Dali [2]: Used to tape out chips in 65nm process



**Gridded cell methodology**



**Standard cell methodology**

[2] Y. Yang, J. He, R. Manohar. Dali: A gridded cell placement flow. In *ICCAD* 2020.

# AMC [4]: Asynchronous memory compiler

- Memory compiler
  - Enables automatic generation of memory layouts to minimize the development costs of ASIC & processor designs
- Asynchronous memory compiler (AMC)
  - The first open-source memory compiler that generates Asynchronous pipelined SRAMs with high throughput and best-case latency
  - Provides GDSII layout, SPICE netlist, LEF and LIB files, and Verilog model of SRAM for variable size and configurations
  - v1.0 available on GitHub w/ a reference implementation for SCMOS: https://github.com/asyncvlsi/AMC

[4] S. Ataei, R. Manohar. AMC: An asynchronous memory compiler. In *ASYNC* 2019.

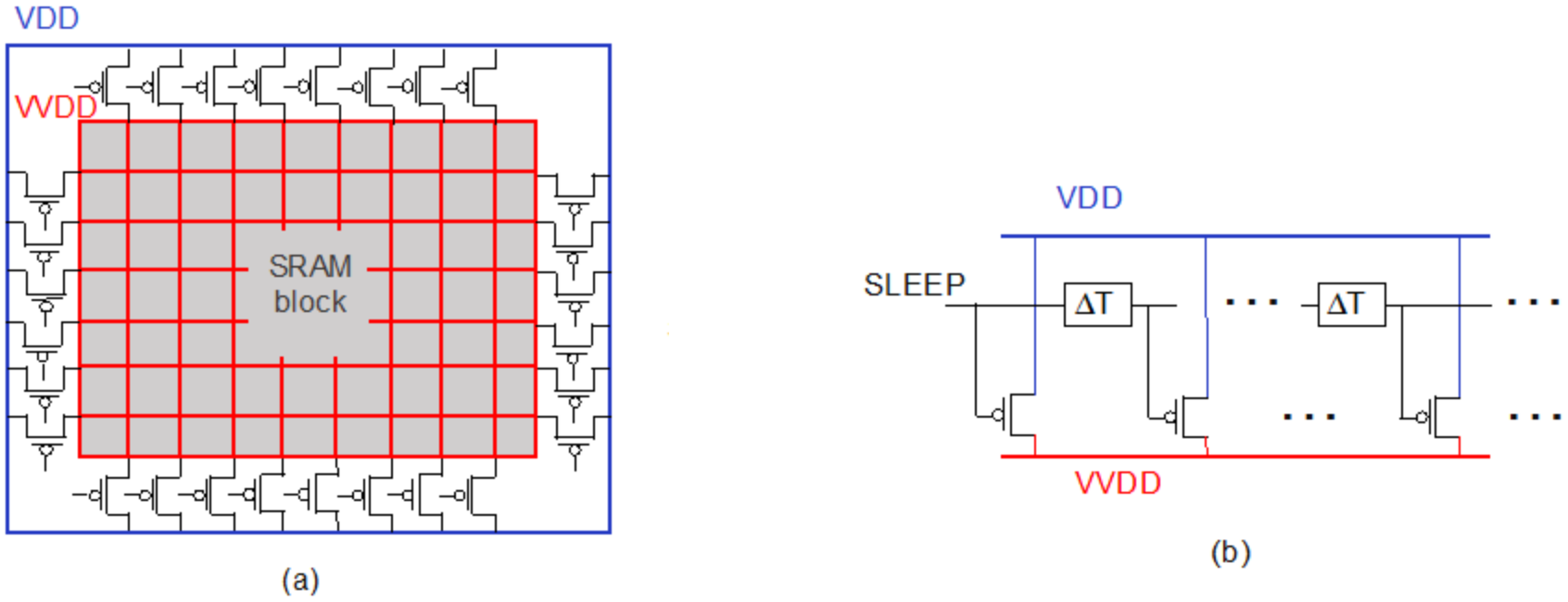
# AMC [4]: Features

- Supported memory functions
  - Three types of operations: read, write & read-modify-write
  - Synchronous interface
  - SRAM BIST (built-in self test) based on March C- algorithm
  - Power-gating option
  - Write-masking option
- Ease of use
  - Support for different memory cell layouts & different bank orientations
  - Portable to new technology nodes;
    successful on 0.5um, 65nm, 28nm & 12nm FINFET technologies

[4] S. Ataei, R. Manohar. AMC: An asynchronous memory compiler. In *ASYNC* 2019.

# AMC [4]: Power-gating option

- Power-gating provides two operation modes for asynchronous SRAMs:
  - SLEEP or low power mode
  - WAKE-UP or active mode

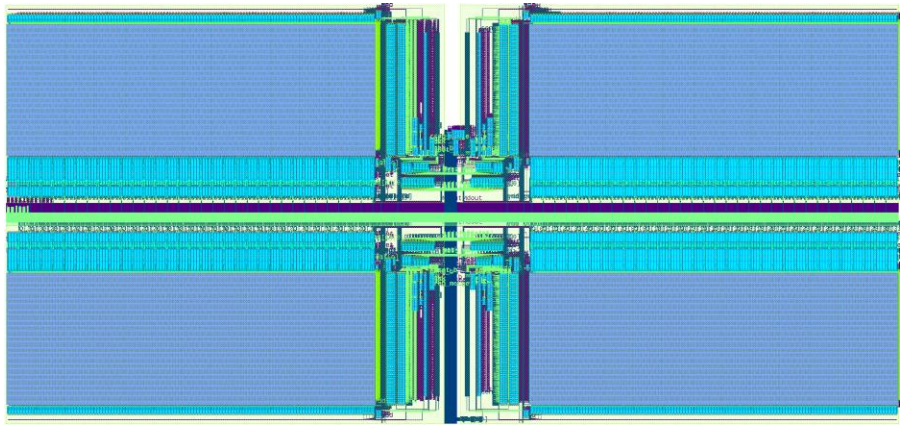



(a) Ring-style power-gating and (b) Daisy-chain SLEEP signal distribution

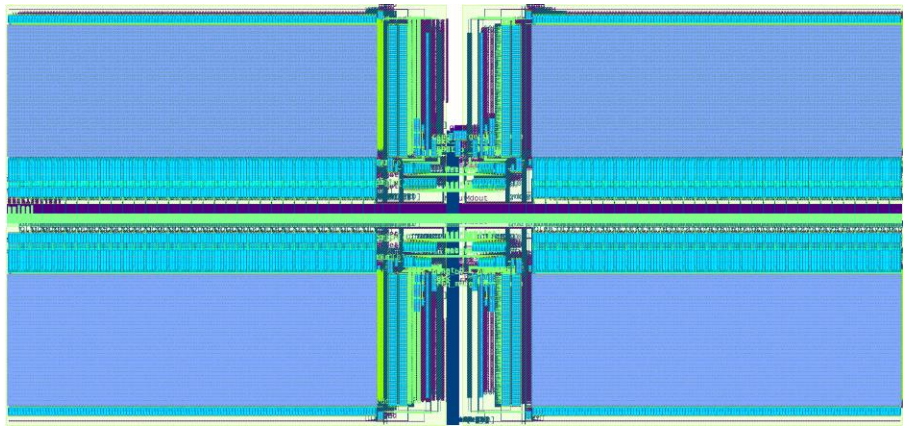
[4] S. Ataei, R. Manohar. AMC: An asynchronous memory compiler. In *ASYNC* 2019.

# AMC [4]: Write-masking option

- Write-masking determines the data bits to write during the memory write mode
  - When the write mask pin *k* is high (WM[k] =1), the corresponding data bit (DIN[k]) is selected, and its data is written to the memory
  - When the write mask pin is low, no data is written for that bit and memory cell retains its previous value



(a)



(b)

32KB SRAM in 12nm FinFET technology (a) without write-masking **1x** and (b) with write-masking **1.04x**

[4] S. Ataei, R. Manohar. AMC: An asynchronous memory compiler. In *ASYNC* 2019.

# Conclusions

- Updates in our async design tool chain
  - Cyclone, asynchronous timer & power analyzer
  - BiPart, deterministic parallel hypergraph partitioner
  - Dali, gridded cell placer
  - AMC, asynchronous memory compiler
- Future works
  - Make the flow timing driven
  - Improve the flow in terms of QoR & runtime
  - Support for more async logic families

# Thanks!

Visit our GitHub repository at http://github.com/asyncvlsi/