# Efficient Hypergraph Modeling of VLSI Circuits for MFS-Based Emulation and Simulation Acceleration

**Jiahao Xu, Chunyan Pei, Shengbo Tong, and Wenjian Yu**

**Dept. Computer Science & Tech., BNRist, Tsinghua University, Beijing, China**

# Outline

- Background&Motivation
- Problem Formulation
- Adaptive Flattening
- Clock Modeling
- Evaluations

- **Background&Motivation**
- Problem Formulation
- Adaptive Flattening
- Clock Modeling
- Evaluations

# Introduction

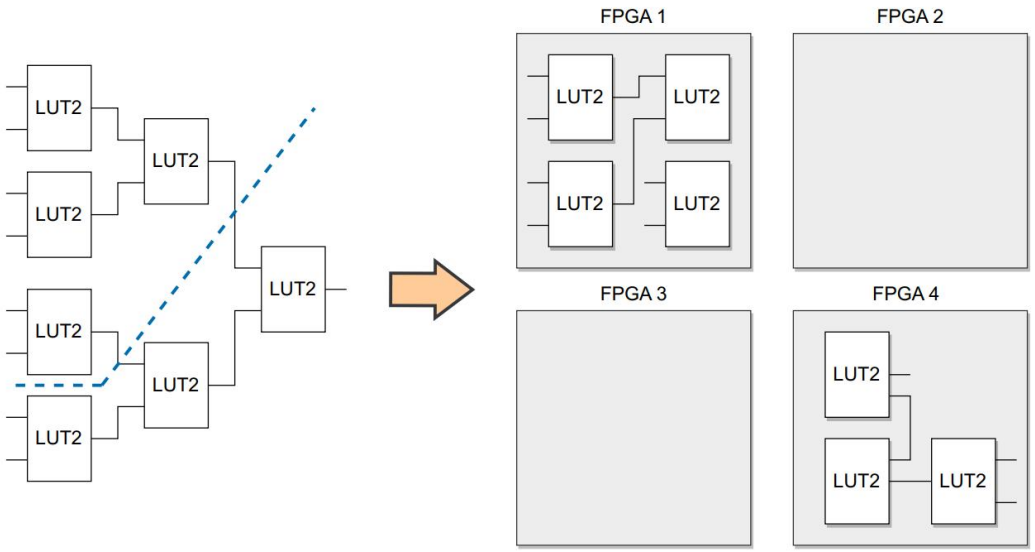
- **Multi-FPGA system (MFS) is widely employed for logic emulation and simulation acceleration.**

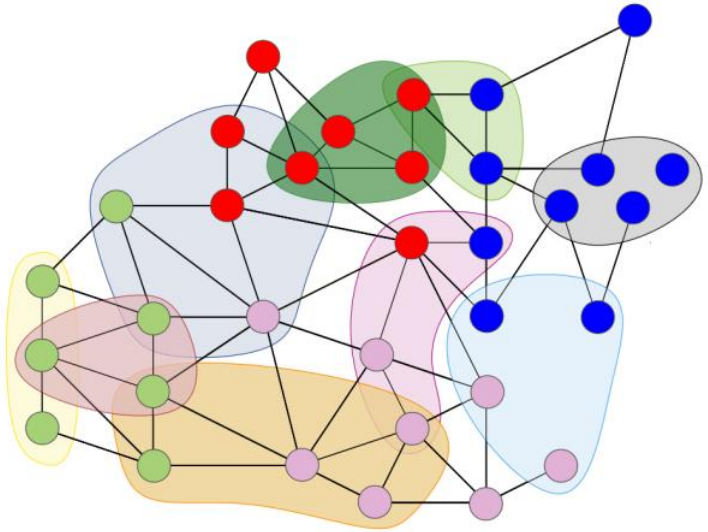
- **The capacity of a single FPGA is relatively limited.**

- **How to effectively partition and map the circuit netlist into MFS is of concern.**

# Hypergraph Partitioning

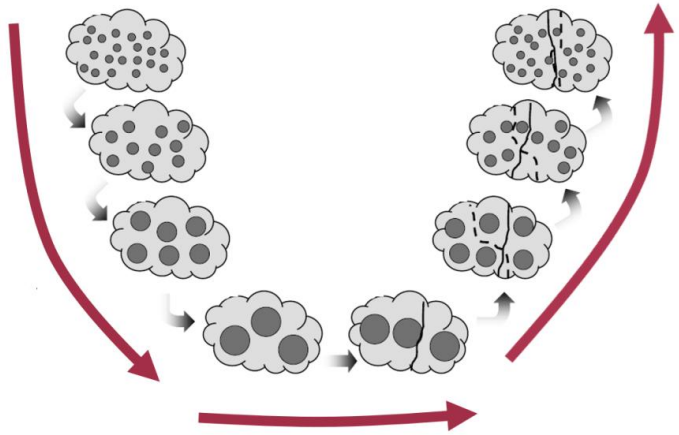
- **The netlist is converted to abstract hypergraph first.**
- **Hyperedges in hypergraph can have more than two vertices.**
- **Hypergraph partitioning is NP-hard.**

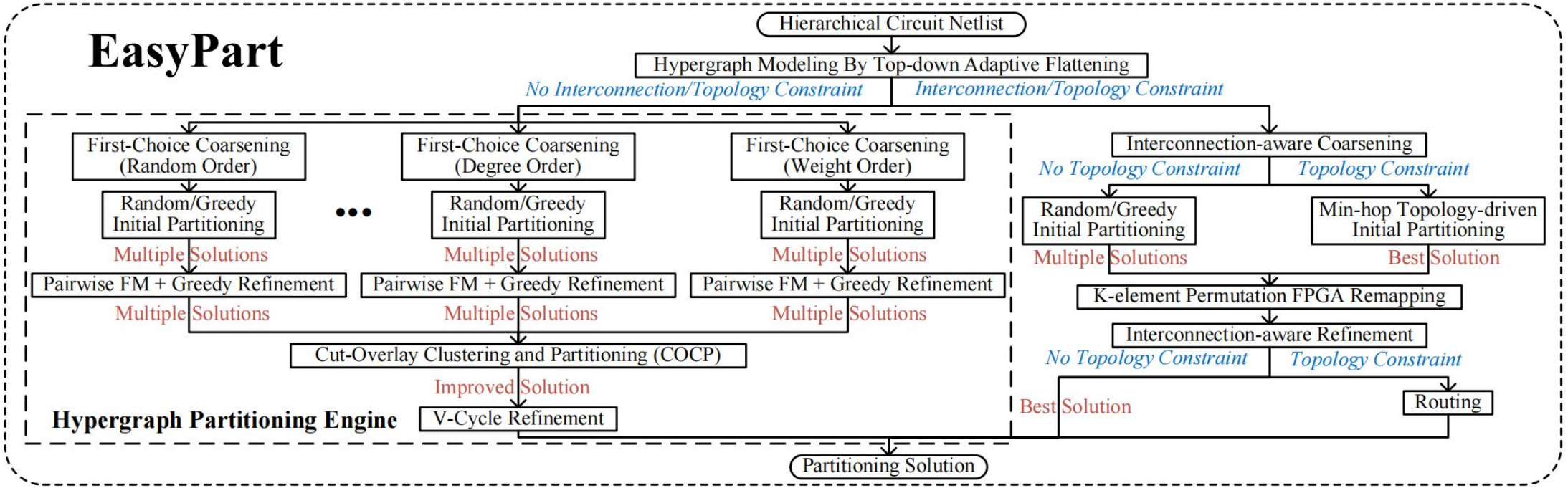


# Multilevel Partitioning Engine

- **The multilevel framework consist of three main phases: Coarsening, Initial Partitioning, Refinement.**





- Background&Motivation
- **Problem Formulation**
- Adaptive Flattening
- Clock Modeling
- Evaluations

# Problem Formulation

- **Multiple Constraints**
  - Mainly, the total **resources** occupied by vertices assigned to a single block cannot exceed a threshold (better if balanced).

$$(\frac{1}{K}-\epsilon)\sum_{v\in V}w(v)\leq\sum_{v\in V_i}w(v)\leq(\frac{1}{K}+\epsilon)\sum_{v\in V}w(v)\ ,\forall\ 1\leq i\leq K$$

  - $w(v)$ **can be a vector** to represent different resources.

  - Other constraints: **fixed vertex** constraint, **grouping** constraint, **interconnection** constraint, **topology** constraint.....

# Problem Formulation

- **Cutsize**
  - A most important **optimization objective**.

$$c(e) = w(e) \cdot |D(e) \backslash S(e)|$$

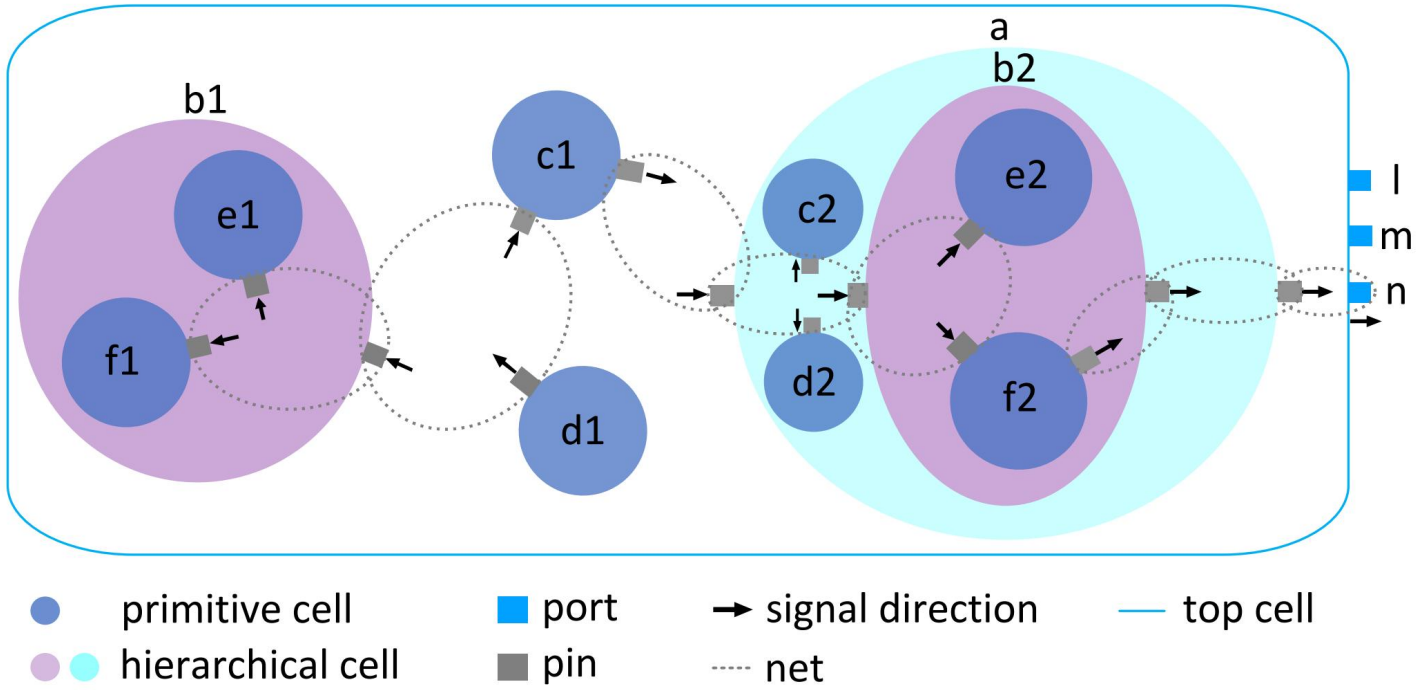
$D(e)$ : Drain nodes
$S(e)$ : Source nodes
$w(e)$ : Edge weight

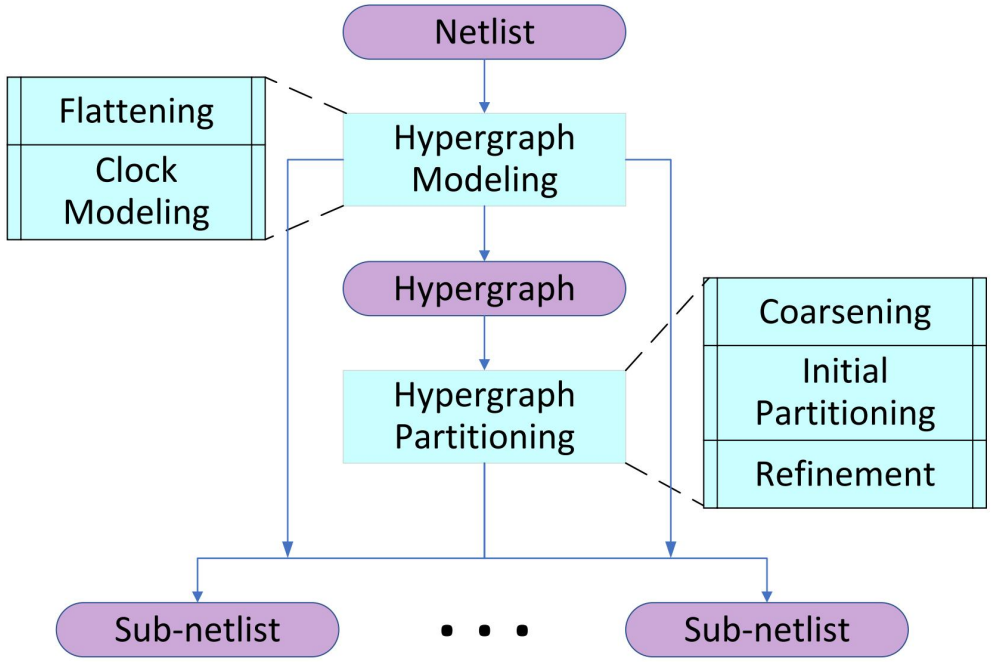
$$cutsize = \sum_{e \in E} c(e)$$

  - **Edge weight * Number of blocks** $e$ being assigned to.

# Hypergraph Modeling

- **VLSI circuits are usually designed in a hierarchical manner. Flattening the hierarchical netlist is the first task of the hypergraph modeling.**



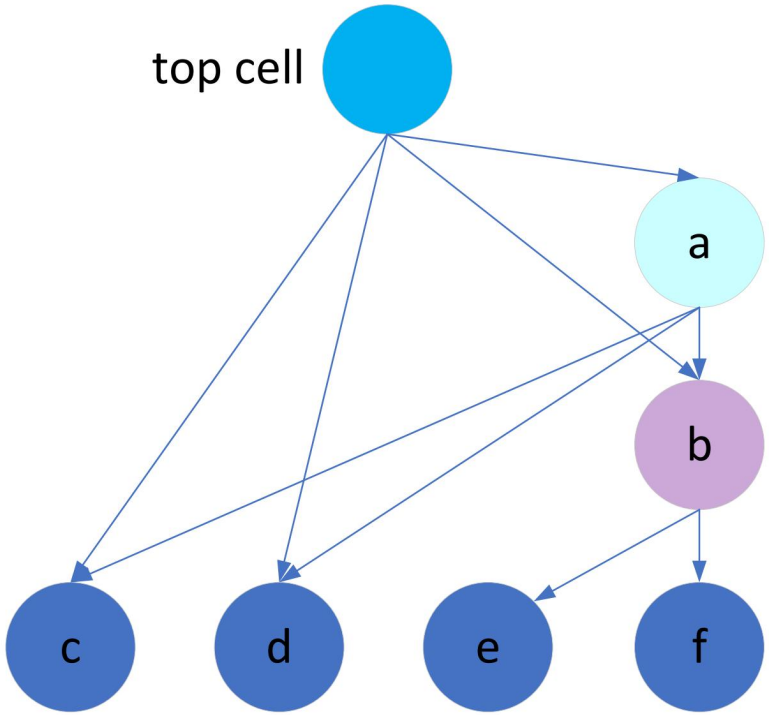



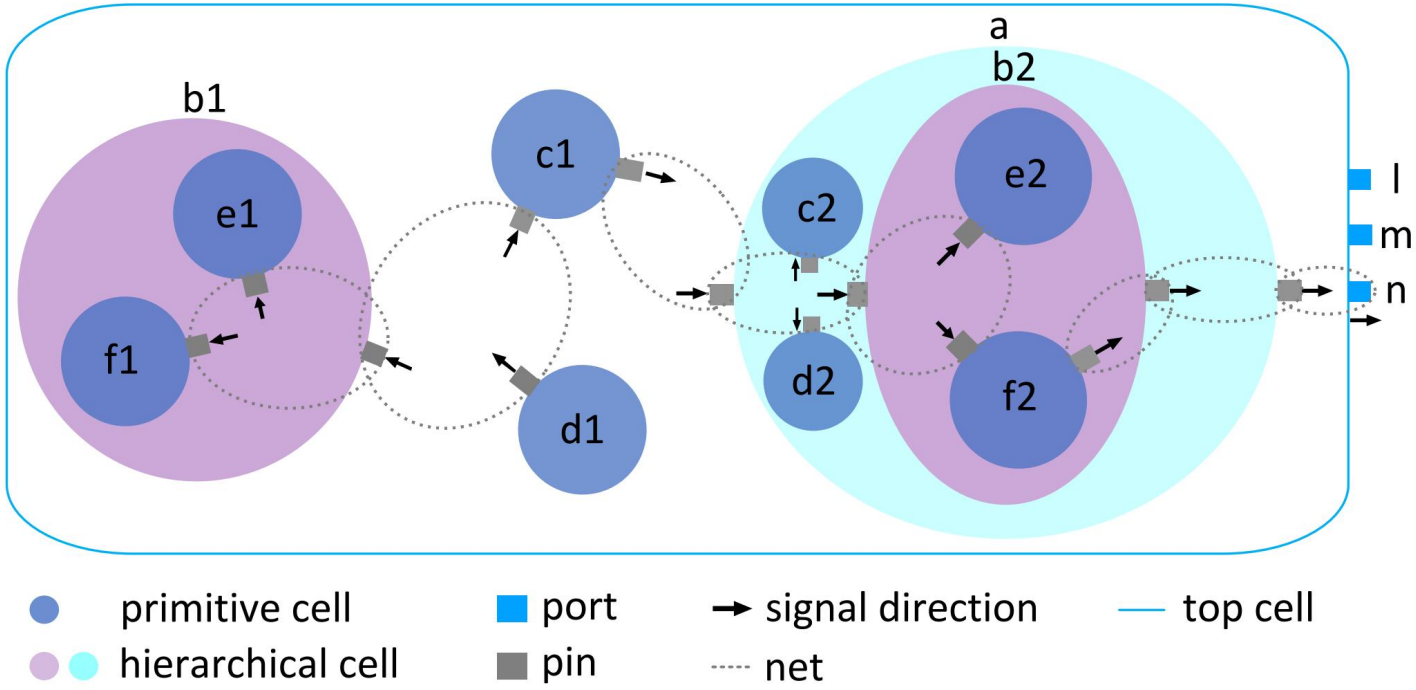

- **Observation: Leverage the hierarchical information.**
  - **Keep some hierarchical cells unflattened.**

- Background&Motivation
- Problem Formulation
- **Adaptive Flattening**
- Clock Modeling
- Evaluations

# Adaptive Flattening

- **Flattening the hierarchical netlist is the first task of the hypergraph modeling.**
  - **A trivial method is to flatten all the hierarchical cells.**
  - **Leading to huge hypergraph.**

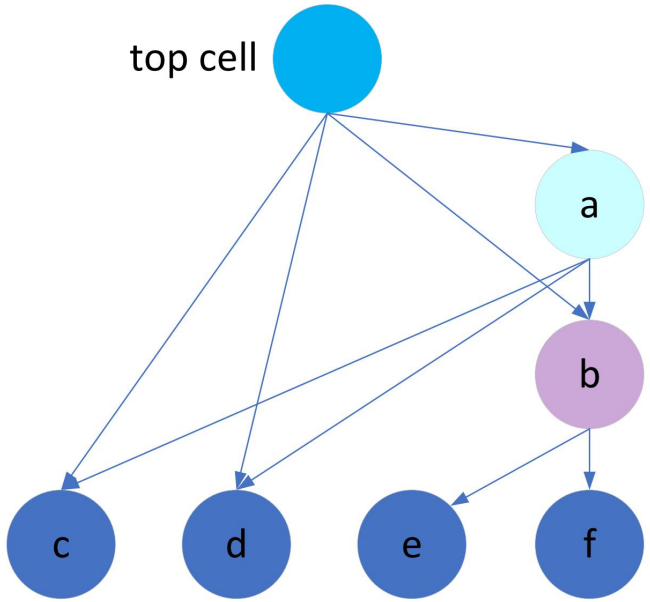


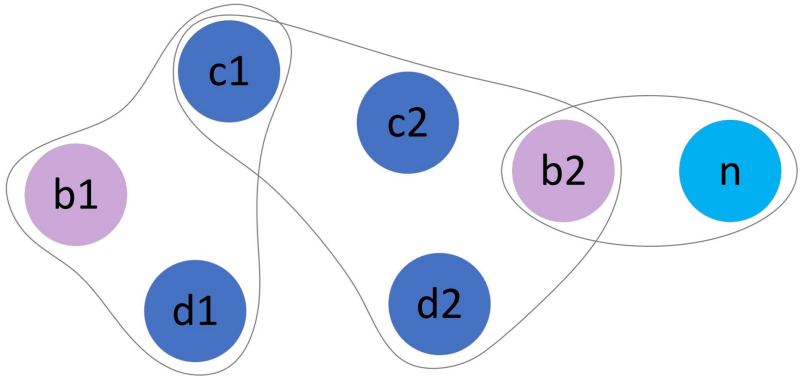



# Adaptive Flattening

- **Try: Keep some hierachycal cells as hypernodes.**

- **To reduce cutsize and satisfy resource balance:**
  - Each hypernode should not be too huge.
  - Dense internal connections, sparse external connections -> Tend to keep.
  - Otherwise -> Tend to flatten.



# Adaptive Flattening

- **Calculate resources based on Dynamical Programming on the netlist DAG.**
- **Decide whether to flatten based on resourcces and density.**

- **Density:**

$$r(u,v) = \sum_{u,v \in e} \frac{w_e}{|e| - 1}$$

$$dens(cell) = \frac{1}{N(cell)} \sum_{v_1 \in cell} \sum_{v_2 \in e} \frac{w_e}{|e| - 1},\ v_1 \in e,\ e \subseteq cell$$

$$= \frac{1}{N(cell)} \sum_{e \subseteq cell} w_e \, |e|$$

**Algorithm 3** Adaptive flattening based on dynamic programming

**Input:** Netlist file F in EDIF format, resource vector $R_{\text{fpga}}$ for each FPGA
**Output:** Hypergraph $G(V, E)$

1: Get top cell $C_t$ from F
2: DfsResourceCal($C_t$)
3: Calculate the total resources $R_{tot} = R[C_t]$ of netlist F
4: Identify the index of critical resource $k$, according to $R_{tot}$ and $R_{\text{fpga}}$
5: $Q \leftarrow$ top cell $C_t$
6: **while** $|Q| \neq 0$ **do**
7: Pop cell $C$ from $Q$
8: **if** $C$ is not primitive **then**
9: **if** $R[C](k) \geq \epsilon_1 R_{\text{fpga}}(k) \vee R[C](i) \geq \epsilon_2 R_{\text{fpga}}(i), \forall i \neq k$ **then**
10: Flatten $C$
11: Append $Q$ with child cells in $C$
12: **end if**
13: **end if**
14: **end while**
15: Merge nets associating cells in $Q$ and use them to form $G(V, E)$

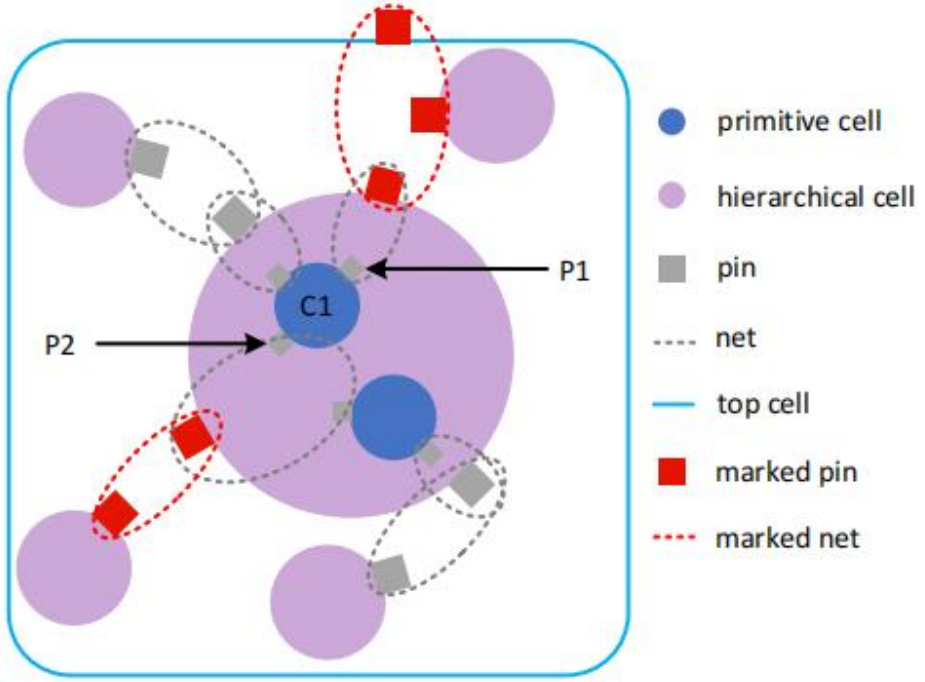
- Background&Motivation
- Problem Formulation
- Adaptive Flattening
- **Clock Modeling**
- Evaluations

# Clock Modeling

- **Another nontrivial task in the hypergrpah modeling is the clock modeling which preserves the information of clock nets needed for deploying sub-netlists later.**

- **Ports, nets connected to a clock signal port should be labeled.**

- **Some primitive cells transmit clock signal.**
  - Hierarchical cells exist after adaptive flattening.
  - The propagation of clock signal within them needs to be considered.



# Clock Modeling

- **Search the hypergraph by BFS.**
- **When encounter hierarchical cells, temp-flatten it.**
  - Check whether the primitive cells inside transmit clock signal.
  - Execute recursively if encounter hierarchical cells inside.
- **BFS starts from different top-level clock signal port.**
  - **Can be parallelized.**

**Algorithm 5** The parallel clock modeling algorithm

**Input:** Hypergraph $G(V, E)$
**Output:** Clock net array $Clk$

```
1:  Create queue Q ← top-level cells
2:  Create threads T_i, i = 1, 2, ..., T_num
3:  for each thread T_i do
4:      Pop cell C from Q
5:      Temp-flatten C
6:      Assign nbrs of ports on C
7:  end for
8:  Construct graph with top-level ports included
9:  Create Q ← top-level clock ports
10: while |Q| ≠ 0 do
11:     Pop port P from Q
12:     for each neighbor port nbr in nbrs[P] do
13:         if nbr not visited then
14:             Assign Clk[nbr]
15:             Assign Clk[net]    ▷ for each net where P, nbr ∈ net ∈ E
16:             Assign Clk[C]      ▷ P and nbr are ports on cell C
17:             Q ← nbr
18:         end if
19:     end for
20: end while
```

- Background&Motivation
- Problem Formulation
- Adaptive Flattening
- Clock Modeling
- **Evaluations**

# Evaluations

**Table 1: Characteristics of benchmarks.**

| Benchmark | #Cells | #Nets | Benchmark | #Cells | #Nets |
|---|---|---|---|---|---|
| industry1 | 1,920,989 | 4,402,785 | industry3 | 31,779,122 | 32,030,744 |
| industry2 | 6,266,965 | 15,340,974 | industry4 | 63,536,570 | 64,013,296 |

**Table 2: The computational results with the approach of completely flattening and the proposed adaptive flattening approach implemented by the basic algorithm and the DP-based algorithm.**

| Benchmark | Complete flattening | | | | Basic algorithm of adaptive flattening (Alg. 1) | | | | | | DP-based algorithm of daptive flattening (Alg. 3) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $T_1$ (s) | Mem(GB) | $T_2$ (s) | *cutsize* | $\|V\|$ | $\|E\|$ | $T_1$ (s) | Mem(GB) | $T_2$ (s) | *cutsize* | $\|V\|$ | $\|E\|$ | $T_1$ (s) | Sp1 | Mem(GB) | $T_2$ (s) | Sp2 | *cutsize* |
| industry1 | 63 | 13.4 | 475 | 4773 | 506,308 | 773,551 | 62 | 6.6 | 100 | 4107 | 506,623 | 773,856 | 16.2 | **3.8X** | 6.6 | 105 | **4.5X** | 4054 |
| industry2 | 287 | 54.7 | 2330 | 7894 | 337,150 | 1,110,690 | 201 | 23.8 | 138 | 8236 | 337,150 | 1,110,690 | 30 | **6.7X** | 23.8 | 149 | **15.6X** | 8251 |
| industry3 | 807 | 122.5 | 6801 | 30 | 56,295 | 59,343 | 2320 | 60 | 16 | 34 | 56,295 | 59,343 | 56 | **41.4X** | 60 | 16 | **425X** | 34 |
| industry4 | 1906 | 245.1 | >2h | - | - | - | >2h | - | - | - | 112,575 | 115,615 | 111 | - | 119.8 | 50 | - | 97 |

$T_1$ denotes the time for netlist flattening; $T_2$ denotes the time for hypergraph partitioning; Mem denotes the peak memory consumption during the flattening. Sp1 means the speedup ratio of Alg. 3 to the basic adaptive flattening (Alg. 1), while Sp2 means the speedup ratio of the hypergraph paratition to that based on complete flattening.

# Evaluations

**Table 3: The results of parallel clock modeling algorithm.**

| Benchmark | Alg. 5 (single-thread) | | Alg. 5 (8 threads) | | |
|---|---|---|---|---|---|
| | Time (s) | Mem (GB) | Time (s) | Sp. | Mem (GB) |
| industry1 | 83 | 6.1 | 19 | **4.4X** | 6.6 |
| industry2 | 300.3 | 22.8 | 39.7 | **7.6X** | 23.8 |
| industry3 | 220.2 | 59.4 | 32.3 | **6.8X** | 60 |
| industry4 | 466 | 119.8 | 72.3 | **6.4X** | 120.5 |

# Conclusion

- The proposed **adaptive flattening and clock modeling method** efficiently construct the hypergraph for partitioning.

- The hypergraph is of an appropriate scale, which greatly **cuts down the time of hypergraph partitioning** while **preserving the solution quality.**

# Efficient Hypergraph Modeling of VLSI Circuits for MFS-Based Emulation and Simulation Acceleration

## Thank you!

## Q&A

**Presenter: Xu Jiahao**

**Email: jhxu24@cse.cuhk.edu.hk**