Synthesis and DSE techniques for matrix-based ambipolar logic architectures

N. Yakymets, I. O'Connor, K. Jabeur, S. Le Beux

Prof. Ian O'Connor
ian.oconnor@ec-lyon.fr
Lyon Institute of Nanotechnology
Université de Lyon – Ecole Centrale de Lyon

Main drivers and levers

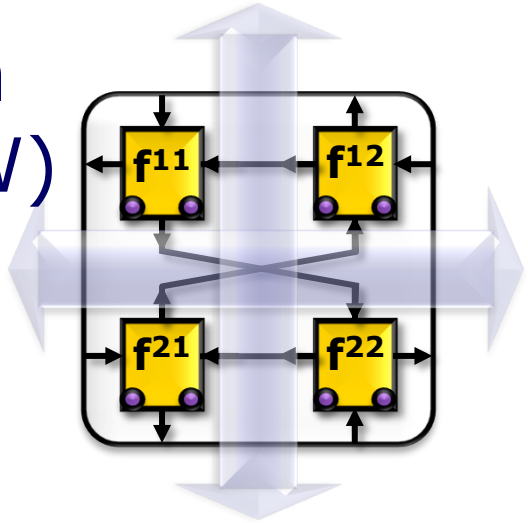
- Reducing energy consumption in computing is key to **cost management**, **sustainability**
  - For scientific computing & HPC
  - For data management
  - For pervasive computing & IoT
- This means computing as fast as CMOS, but with **better energy efficiency**
- Where are the levers?
  - Improving devices : **energy / bit**
  - Improving architectures : **adapting** hardware to application requirements
  - Improving applications : **just enough** (accuracy, speed …)

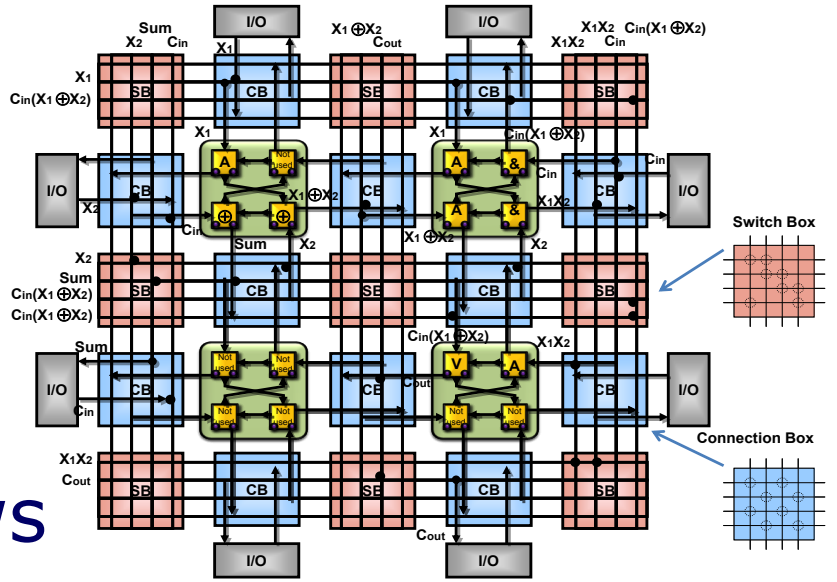
Ambipolar nanofabrics

- At nanoscale, only **regular fabrics** are viable for systematic variability (e.g. OPC) and for self-assembly
- Still random variability – need **device-level tuning** and **system-level reconfigurability**
- **Multi-gate devices** can offer this level of flexibility
- Ambipolar double gate devices in regular nanofabrics could offer a means to solve many problems at the same time

Reconfigurable nanofabrics

- **Dense nano-matrices** of **reconfigurable cells** based on ambipolar DG-FETs (CNT, Si-NW)
- **Enriched functionality**
- **Multiple data directivity** and operator co-implementation
- **Fault-detection** and operator propagation through island matrix
- **Hierarchical control**
- **Packing** tools and flows





Agenda

- Key ideas
- Matrix-based nanocomputer architecture
  - Logic cell
  - Matrix of cells
  - Cluster of matrices
- O-cycle design approach
  - Design flow
  - Mapping the problem
  - Optimization
- Results and discussion
- Conclusion and future work

Research protocol

- Problem: **mapping large applications** onto **hierarchical architectures** based on **novel nanodevices**
- Hypothesis: combining both IP reuse and multi-level mapping concepts will enable to cope with application complexity and reduce circuit design time
- Contribution: **"O-cycle" design flow** that exploits the **IP reuse** concept for the development and further reuse of hardware component libraries through **recursive multi-level mapping**

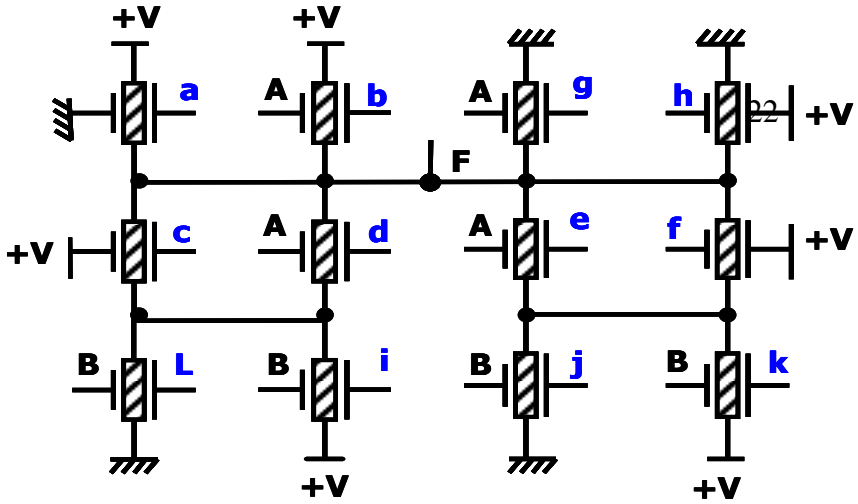
O-cycle design flow

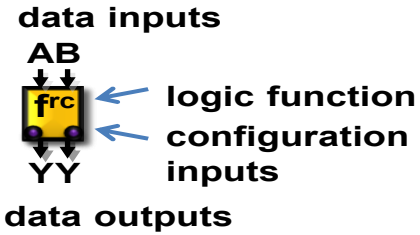
- Observation:
  - **application mapping** executes **top down**
  - **implementation** is carried out **bottom up** using pre-existing libraries of IP cores
- Target flow
  - takes into account the **variable performance** of reconfigurable logic cells associated with their internal functions
  - optimizes a design through **multi-objective search**

N. Yakymets, I. O'Connor, K. Jabeur, S. Le Beux, "Multi-Level Mapping of Nanocomputer Architectures Based on Hardware Reuse," *IEEE J. Emerging and Selected Topics in Circuits and Systems (JETCAS)*, Special Issue on Computing in Emerging Technologies, vol. 5, no. 1, pp. 88-97, Mar. 2015

Logic cell

- Lowest level represented by a **fine-grain logic cell** with low device count based on ambipolar FETs that can be **configured** to any 1 of L ($\leq 2^{2^{N\_inputs}}$) logic functions e.g. RSL_12T





K. Jabeur, et al. "Ambipolar double-gate FET binary-decision-diagram (Am-BDD) for reconfigurable logic cells," *Int. Symp. on Nanoscale Architectures (NANOARCH)*, San Diego, June, 2011.

Function-dependent performance

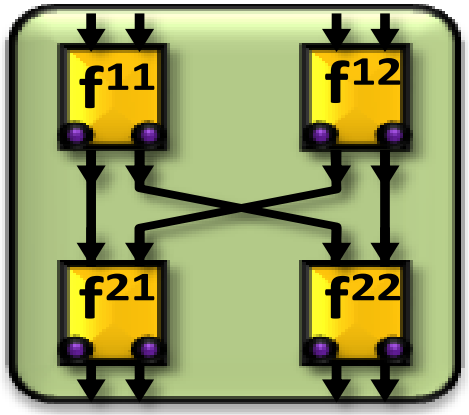
Function	Av. power / uW	Av. delay / ps	Function	Av. power / uW	Av. delay / ps
/(A+B)	20	10.5	B	10	20
1	0	0	0	0	0
A+B	10	15.15	A./B	16	11.5
/A	15.3	7.5	/A+B	12	21
A	10	11	A.B	10	16
B./A	12	21.1	/(A.B)	20	11
A+/B	16.3	11.55	/(A XOR B)	26	16.27
/B	16	13.45	A XOR B	25.4	16

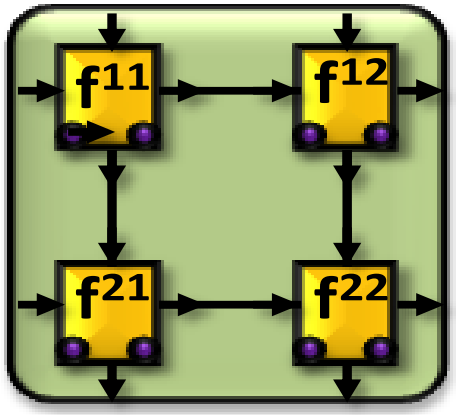
For f=1GHz ; t_r=t_f=20ps ; C_L=150aF

Matrix of cells

- Logic cells organized in a **matrix** through **scalable** and **fixed** interconnect topologies

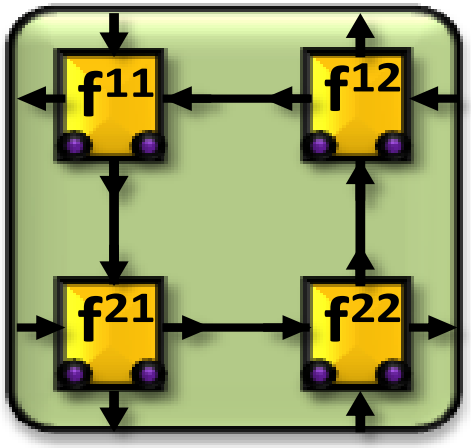
Butterfly (N->S)

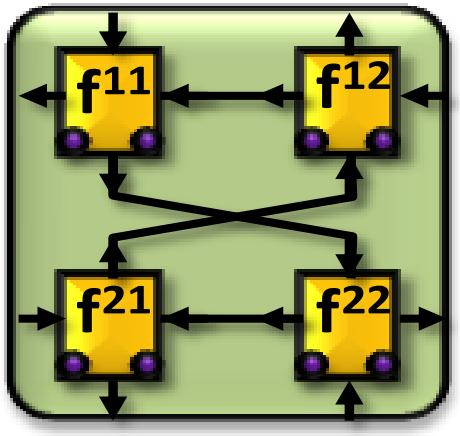




Systolic Array (N->S, W->E)

Cell matrix (N<->S, W<->E)



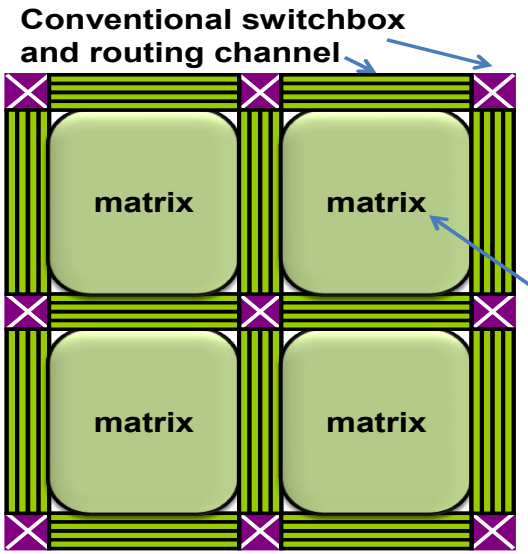


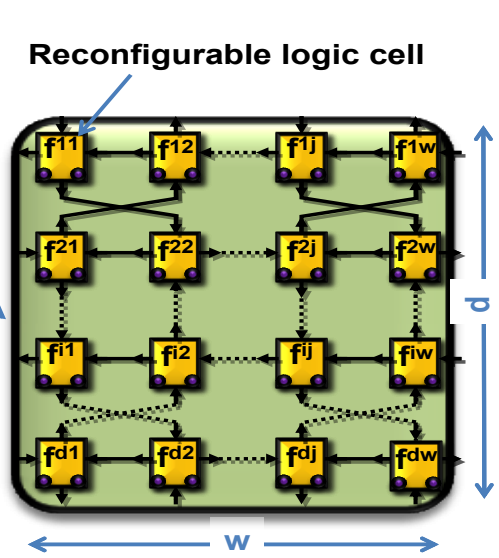
Cross cap (N<->S, W<->E)

Cluster of matrices

- At top level: matrices are grouped into a **cluster**
  - In an **island-style** FPGA architecture (where cell matrices replace configurable logic blocks), or
  - In a **homogeneous cell matrix** (where several matrices are connected through a fixed interconnect topology)

Island style

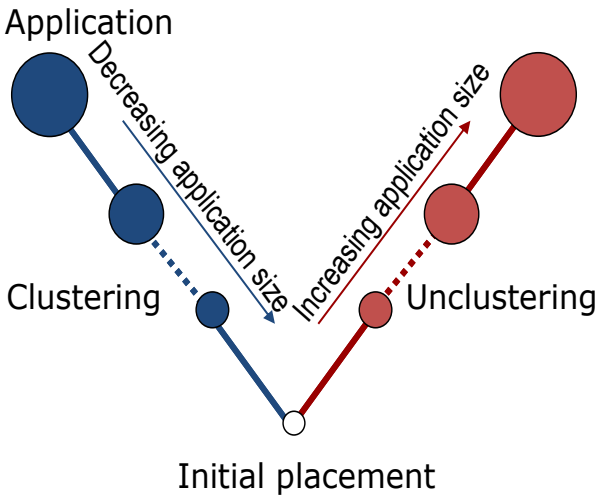




Homogeneous matrix

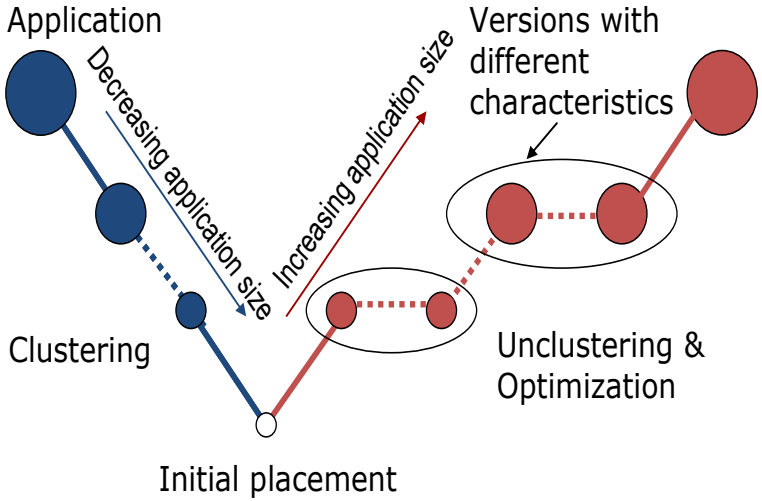
Conventional design flows

- **multi-level V-cycle** uses early clustering algorithms which rely upon 2 phases (clustering, unclustering) :



Dai H, Zhou Q, Bian JN. Multilevel optimization for large-scale hierarchical FPGA placement. *Journal of computer science and technology,* 25(5): 1083 – 1091, 2010.

- **multiple V-cycle** builds on multi-level V-cycle by reclustering an application during the unclustering phase:

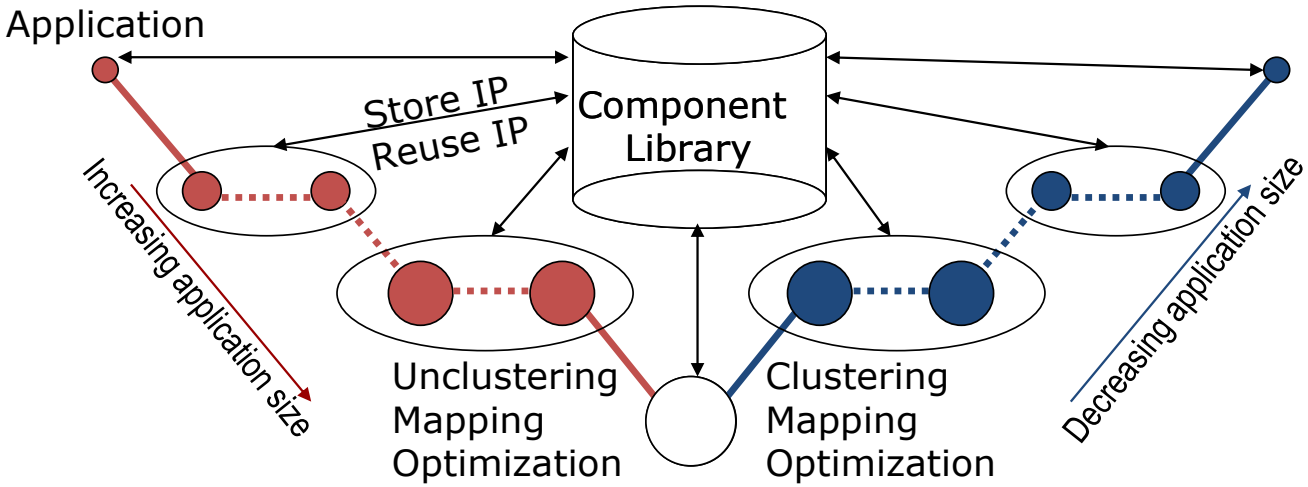


J. Cong, S.K. Lim, "Edge separability-based circuit clustering with application to multilevel circuit partitioning," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 23(3), pp. 246–357, 2004.

V-cycle issues – meet in the middle?

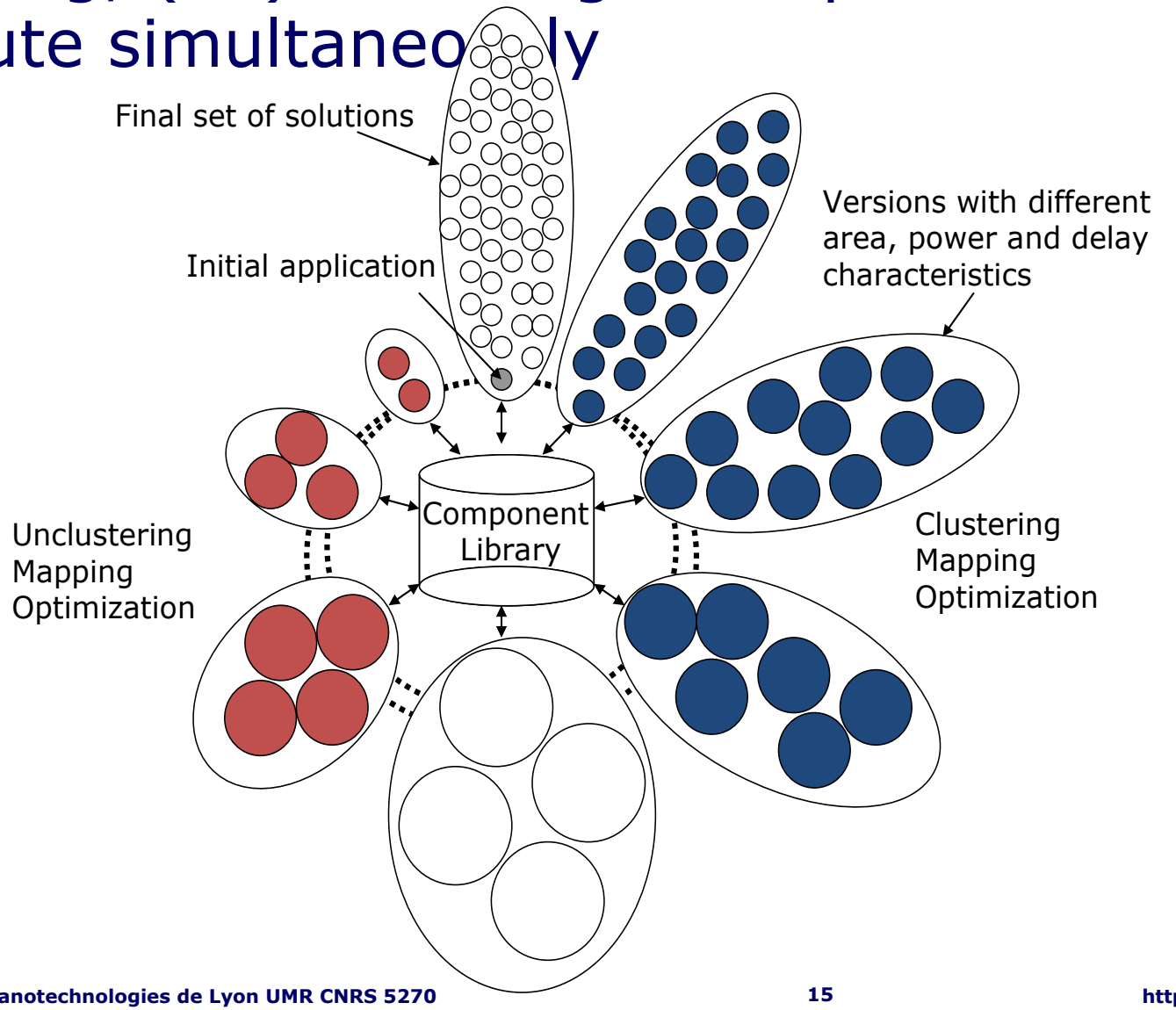
O-cycle approach

- extends the V-cycle design flow to use **pre-existing** libraries of IP cores
- merges application **mapping, placement and routing** tasks of conventional FPGA-based design flow into a **single** mapping process using multi-objective genetic algorithm (GA)



Time-abstracted O-cycle

- Mapping, (un)clustering and optimization execute simultaneously



Comparison of mapping approaches

Mapping approach	Optimization	IP reuse	Design approach
Multi-level V-cycle	Single objective	Not implemented	Bottom-up
Multiple V-cycle	Multiple objective	Not implemented	Bottom-up
O-Cycle (this work)	Multiple objective	Implemented	Bottom-up and Top-down

Problem formulation

- **Application:**
  - hierarchical HDL circuit description represented as a hypergraph *C(V,E)*
    - V = set of hypervertices. Every hypervertex $v_i \in V$ is a module of a top-level circuit representation
    - E = set of hyperedges. Every hyperedge $e_{ij} \in E$ is an interface between two modules represented by hypervertices v_i and v_j
- **Architecture:**
  - also represented as a hypergraph *G(N,T)*
    - N = set of hypervertices. Each hypervertex $n_i \in N$ is a part of the target architecture, suitable for implementation of the largest module $v_l \in V$ of a top-level circuit representation
    - T = set of hyperedges

Problem formulation (continued)

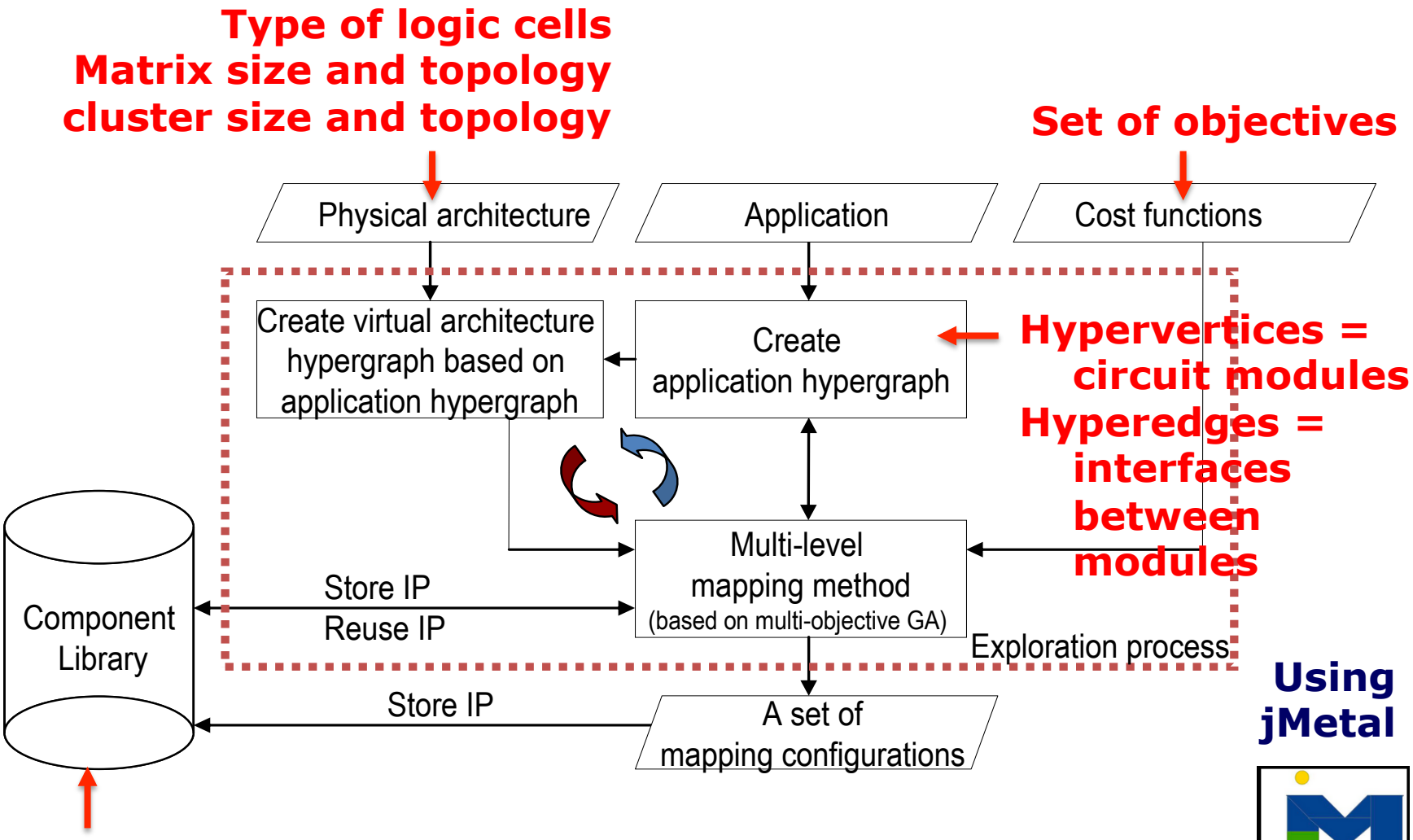
- **Component Library:**
  - represented as a set of already mapped IP cores $L = \{l_1, \ldots, l_q\}$
  - O-Cycle Mapping generates set of mappings $M=\{m_1 \ \ldots, m_k\} \in L$, where $m_i \in M$ is a hypergraph $S_i(V_{si},E_{si})$ describing a single implementation of circuit hypergraph $C(V,E)$ onto the target architecture hypergraph $G(N,T)$
  - IP cores chosen from component library according to main objective(s) to be optimized (area, power, delay)
- **Cost function (fitness + objectives)**
  - Fitness = $\dfrac{\text{paths found in } S_i(V_{si},E_{si})}{\text{hyperedges in } C(V,E)}$

Exploration process

F: G,C, Component Library L:->Mapping Solutions

```
ForEach Hypervertex v of Hypergraph C
  if L contains pre-mapped module corresponding to
  Hypervertex v
    Use pre-mapped module from L
  else
    Get SubHyperGraph C' from v
    Define cluster size, xxy, for G
    If (finest architecture level for C' is reached) OR
    (all Hypervertices of C' are mapped and stored to L)
      Run GA to map C' onto G -> Mapping Solutions
      Update Component Library L
      Return Mapping Solutions
    F(G, C', L)
End ForEach
return Mapping Solutions
```

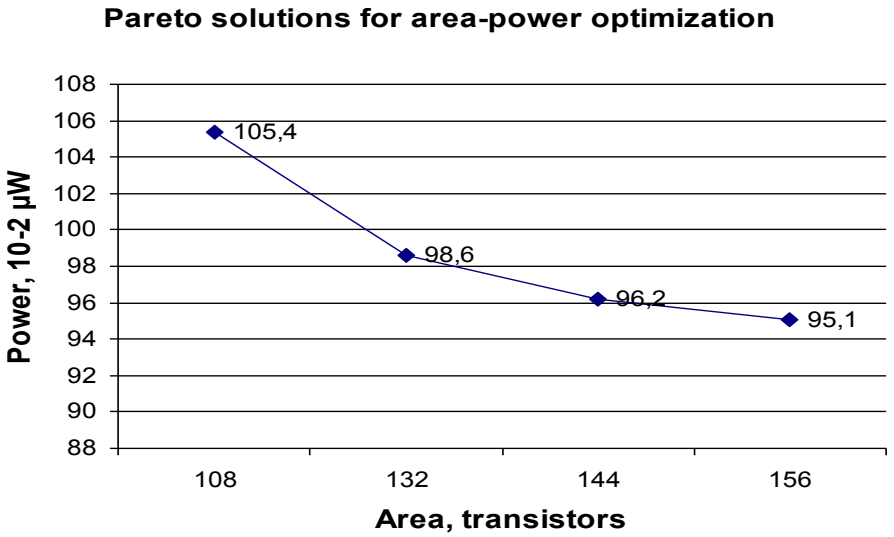
Flow implementation

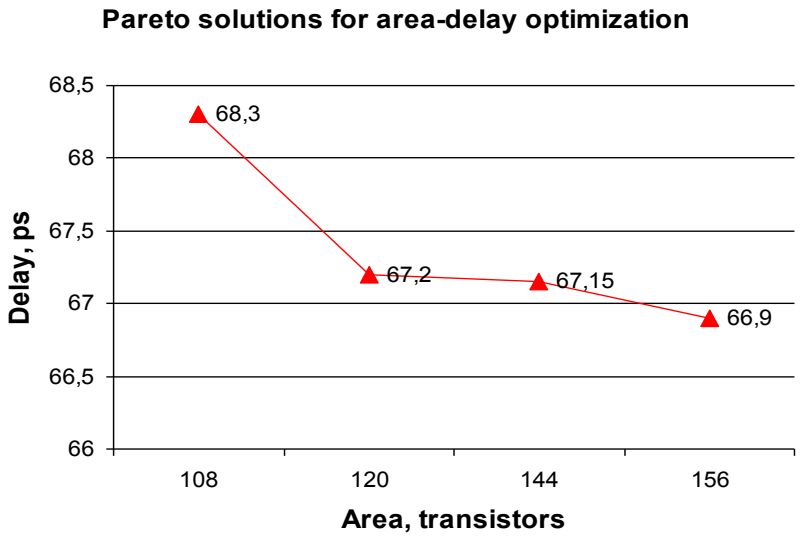


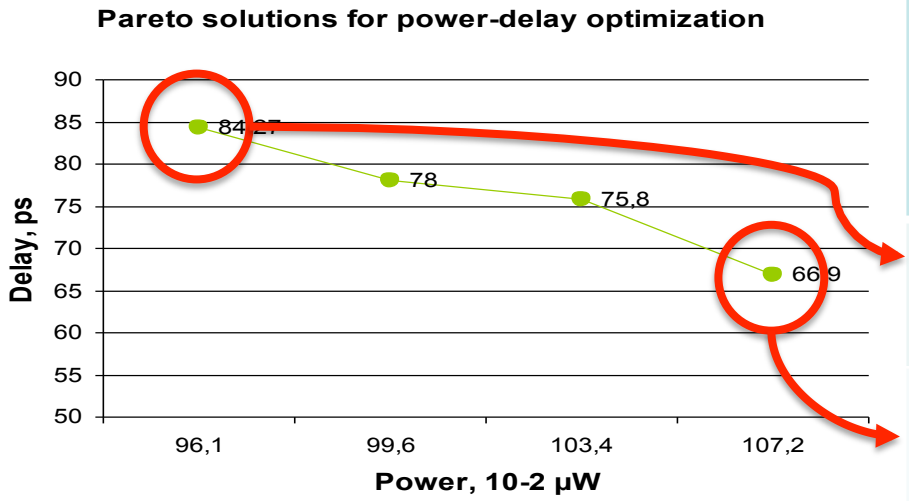
Experimental setup

- Application:
  - MCNC + regular structure benchmarks
- Optimization according to three objectives:
  - design area, power consumption, critical path delay
  - NSGA-II algorithm for mapping (population size: 400 – maximum number of iterations: 10000 - maximum fitness rate - crossover rate: 80% - mutation rate: 20%)
- Physical architecture:
  - RSL_12T reconfigurable cells
  - homogeneous matrix using butterfly, systolic array, cell matrix or cross-cap topologies
- Machine: Intel Core 2 duo 1.83GHz, 1GB RAM
- **5-40s mapping process time** for each hierarchical level

Pareto-fronts for 1-bit adder







FIR-16	Area / devices	Power / mW	Critical path delay / ns
Power-optimized	59652	40.1	27.0
Delay-optimized	48012	44.3	21.4

Cross-cap topology example results

Benchmark	Area / devices	Power / uW	Cr. path delay / ps	Cluster size / vertices
1-bit adder	168	150.8	116.15	4×4 cells
8-bit multiplier	2664	2349.6	1578.2	8×6
12-bit adder	2676	2359.6	1998.8	7×8
16-FIR filter	85440	75347.2	57232	10×9
alu2	984	1132.5	1167.5	6×6
alu4	2448	2890.3	1573.9	5×5
mult32	43008	45032.0	34927.4	6×6
xor5	60	111.6	84.0	3×2 cells

Comparison to V-cycle

Bench-mark	Area / transistors			Critical path delay / ps		
	V-cycle	O-cycle	Impr. %	V-cycle	O-cycle	Impr. %
1-bit adder	543	108	80	84.3	68.3	19
8-bit mult	3487	2412	31	1279.6	1195.4	7
12-bit adder	4007	1956	51	1675.8	1424.6	15
16-FIR	96954	69696	28	45698.8	41920.0	8
alu2	2349	948	60	632.8	586.1	7
alu4	4480	2352	48	1040.0	987.2	5
mult32	50370	40896	19	44737.3	28537.0	36
xor5	489	48	90	87.6	68.8	21
Average	-	-	49	-	-	15

Using systolic array topology

Conclusion

- O-level design flow:
  - chooses among different power and delay characteristics of reconfigurable logic cells that vary according to cell internal function
  - allows the whole system to be optimized to lower power consumption, critical path delay and area
- Experimental results demonstrate, on average, for the systolic array topology and with respect to conventional flows
  - 49% transistor count reduction
  - 31% power reduction
  - 15% critical path delay reduction