Customized and Reconfigurable FPGAs until the Variation is **Coming**

Terrence Mak

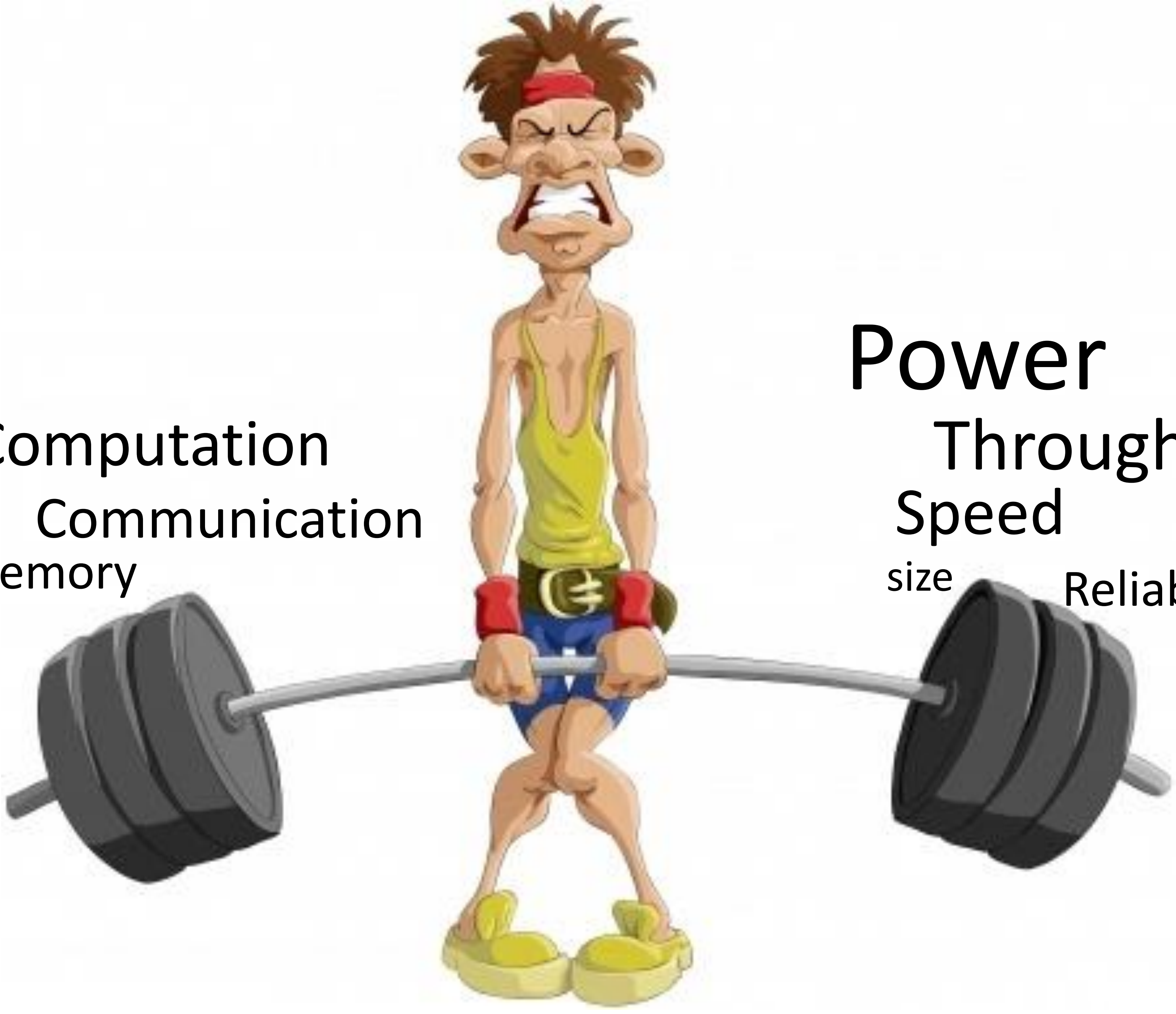
tmak@ecs.soton.ac.uk

Department of Electronics and Computer Science

University of Southampton, UK

Content

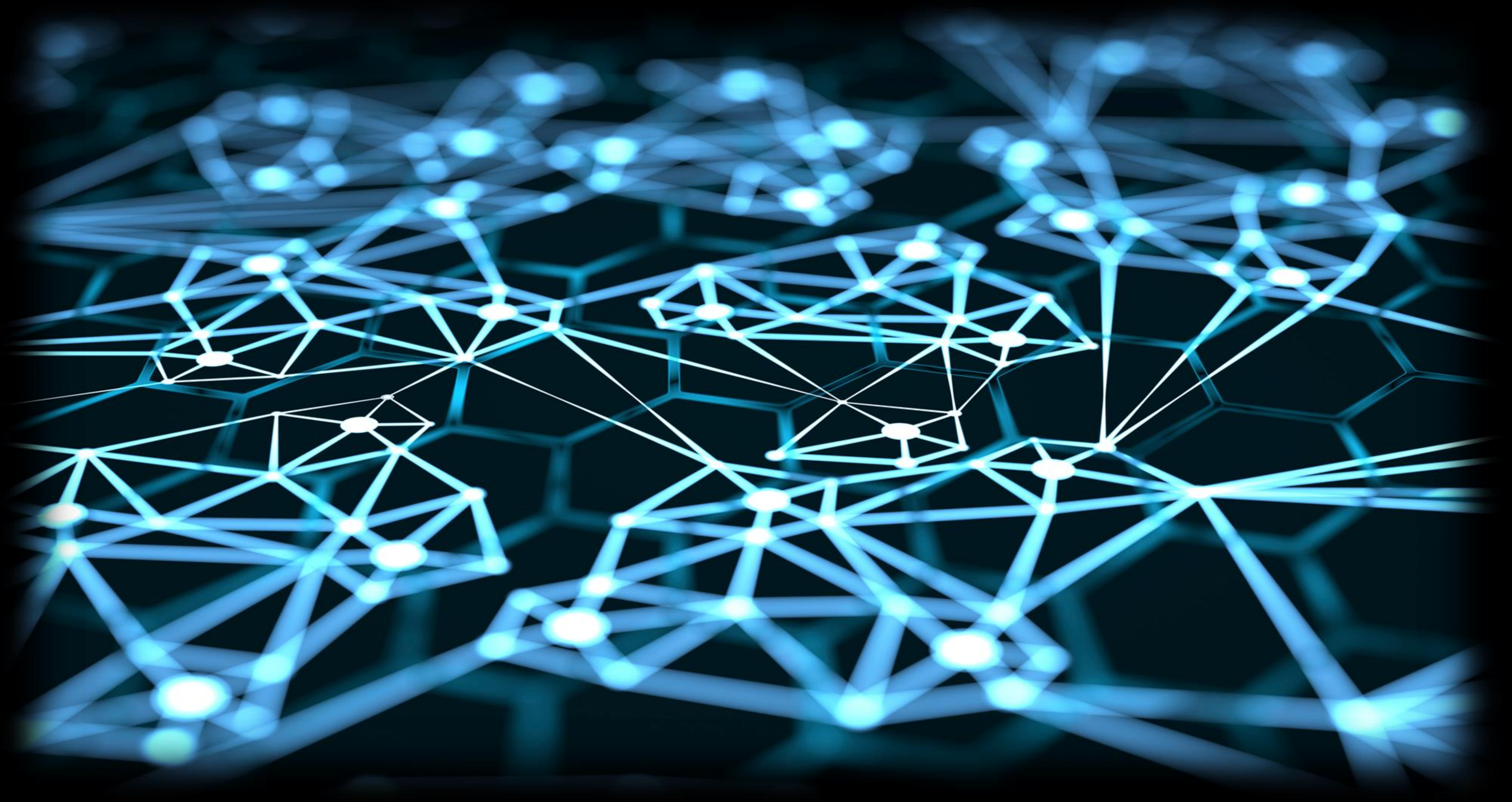
- Previous work
- New challenge and motivation
- The Market-on-Chip architecture and operations
- Potentials and discussions


Power
Computation
Throughput
Communication
Speed
Memory
size
Reliability

How to manage on-chip networks?

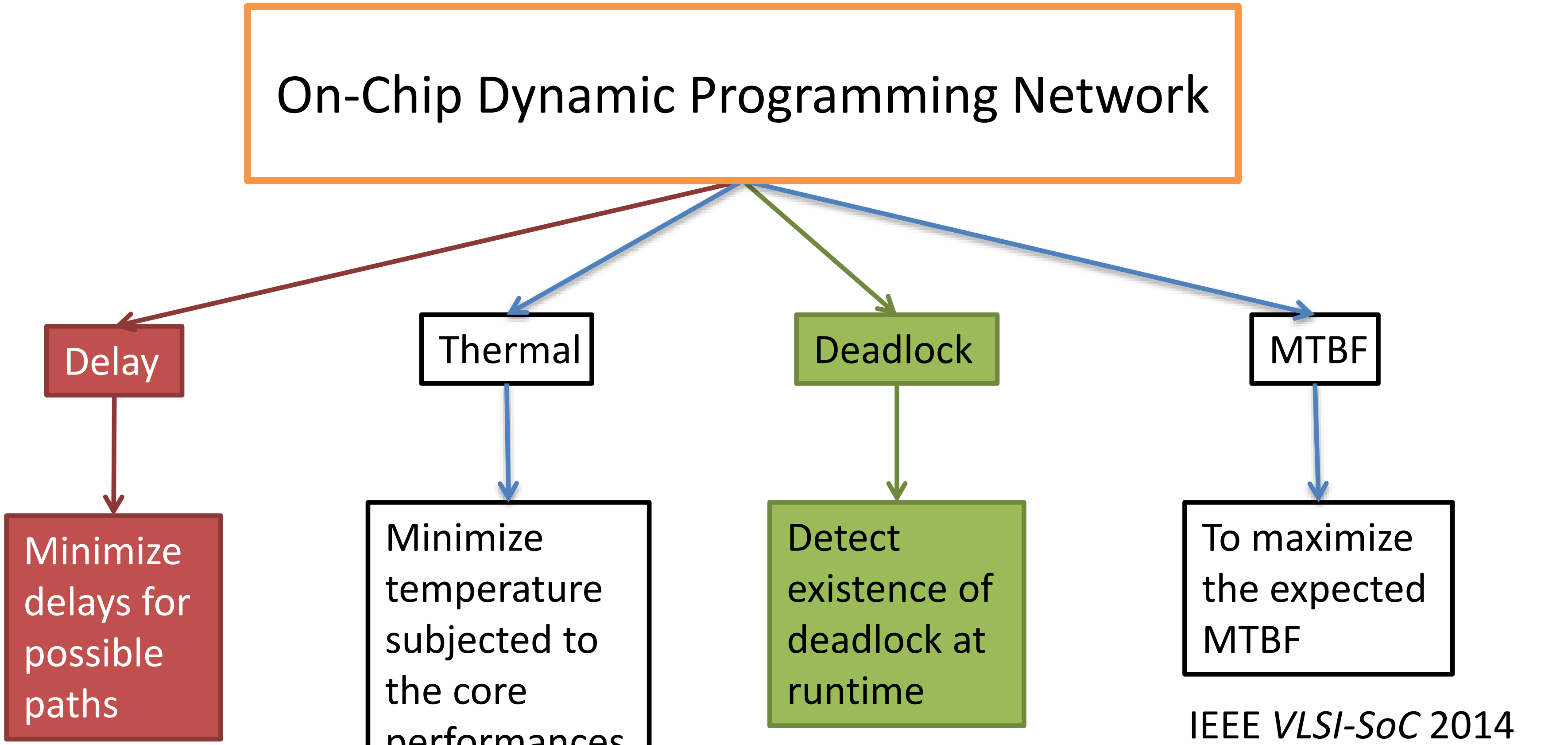
- Power
- Delay
- Thermal
- Reliability

Using networks to managemnt networks



How to Manage On-chip Networks?

- To manage networks, we can use networks


On-Chip Dynamic Programming Network
Delay
Thermal
Deadlock
MTBF
Minimize delays for possible paths
Minimize temperature subjected to the core performances
Detect existence of deadlock at runtime
To maximize the expected MTBF

IEEE Trans. Ind. Electronics 2011

ACM ECS 2014

*DATE'*11 2011
(Best paper award)

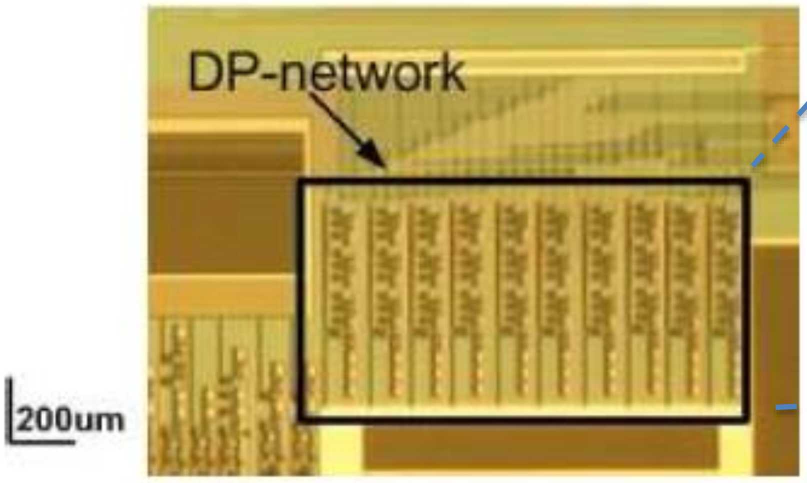
IEEE *VLSI-SoC* 2014
(Best paper award)

The Magic Behind the Scene

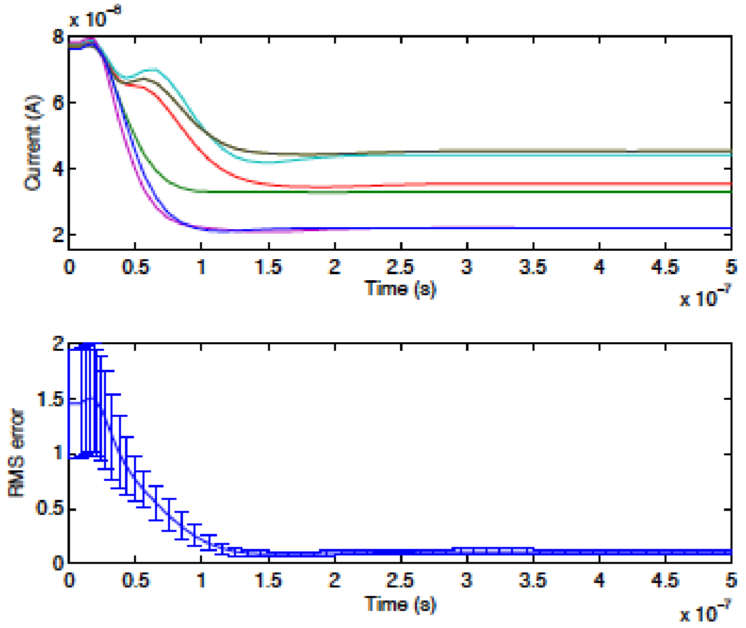
- Re-formulate the Bellman equation as energy function

$$\frac{dV_i(t)}{dt} + \frac{1}{\lambda_i} V_i(t) = \frac{1}{\lambda_i} \min_{\forall k} \left\{ \frac{1}{a_{ik}} \left[C_{i,k} + V_k(t) \right] \right\}$$

- Fabricated chip



Current-mode design



T. Mak *et al.*, *IEEE TCAS-I*, 2010

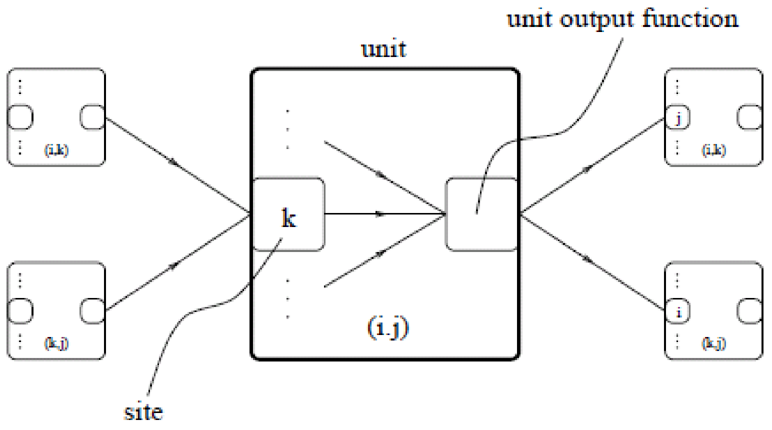
On-Chip Dynamic Programming Network for Optimization

- The Bellman optimality equation

$$V_t^{(k)}(v) = \min_{\forall u \in V} \left\{ V_t^{(k-1)}(u) + C_{i,j} \right\}$$

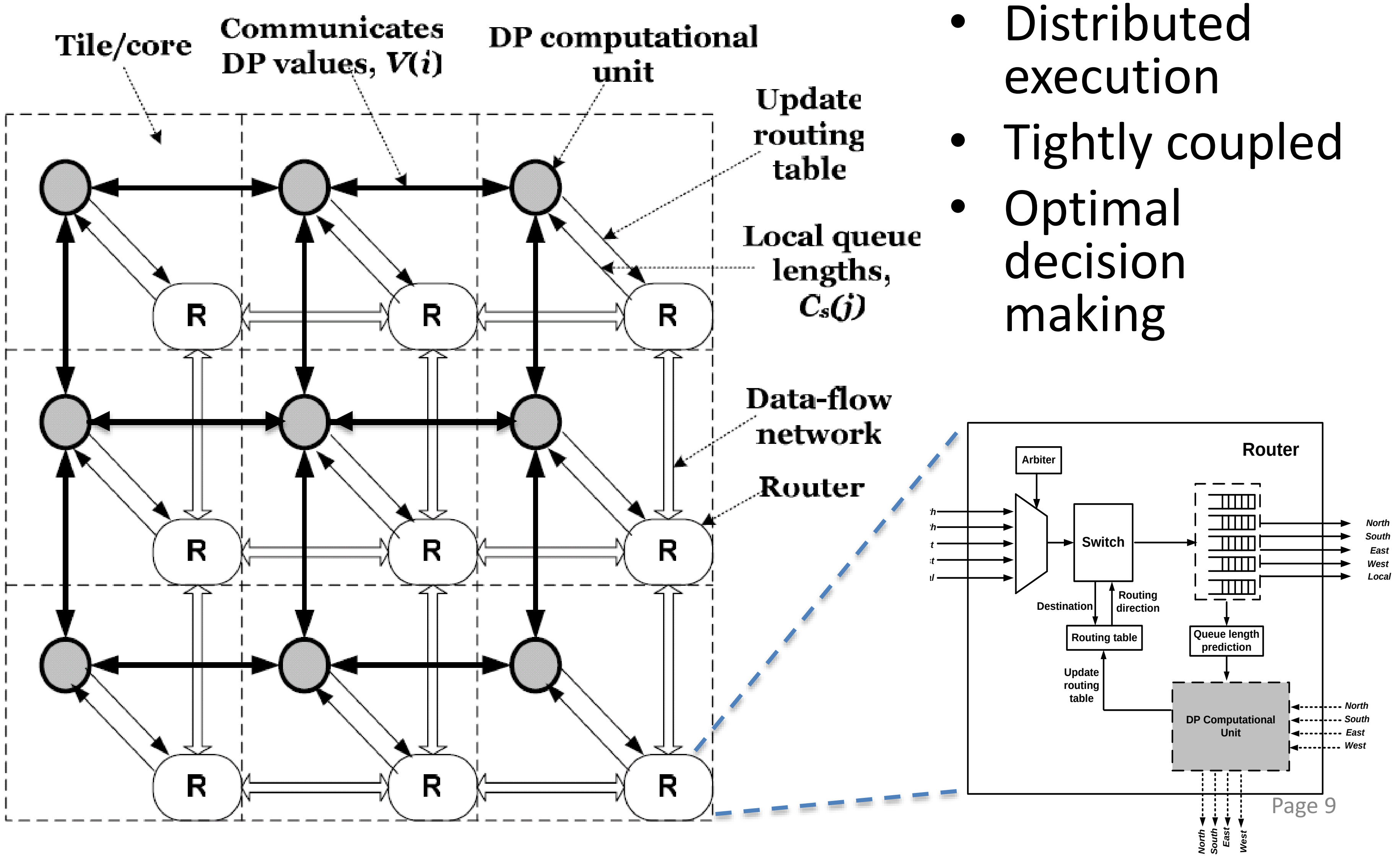
- Dynamic programming network transformation

$$\boxed{\begin{aligned} S_k(i,j) &= \frac{1}{a_{ik}} \left[C_{i,k} + g(k,j) \right], \forall k \\ g(i,j) &= \min_{\forall k} \{ S_k(i,j) \} \end{aligned}} \longrightarrow$$



- On-chip dynamic programming network provides an ***embedded*** and ***distributed*** resolution

On-Chip Dynamic Programming Network for Routing

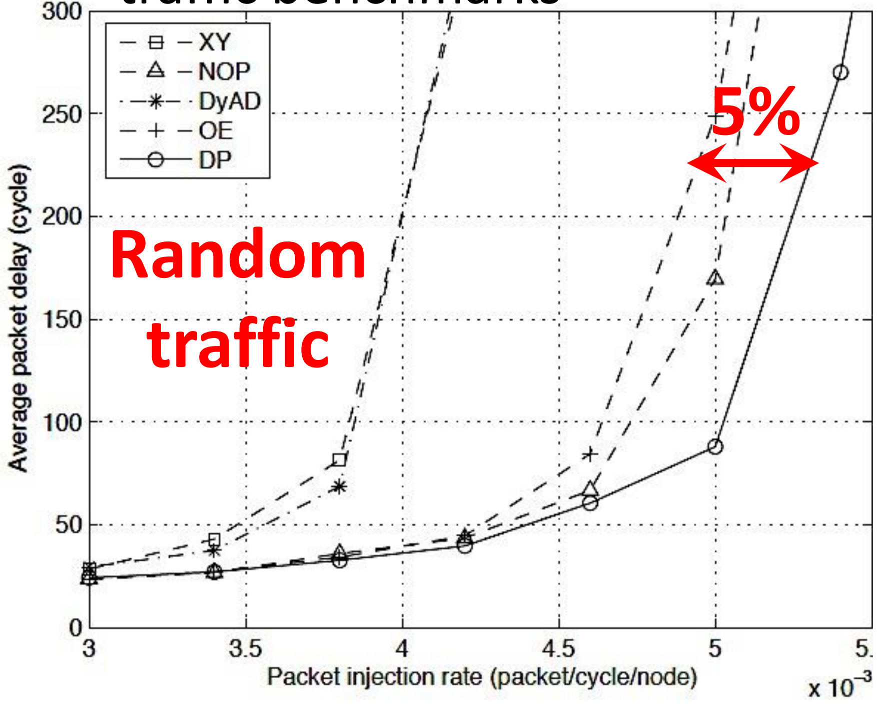


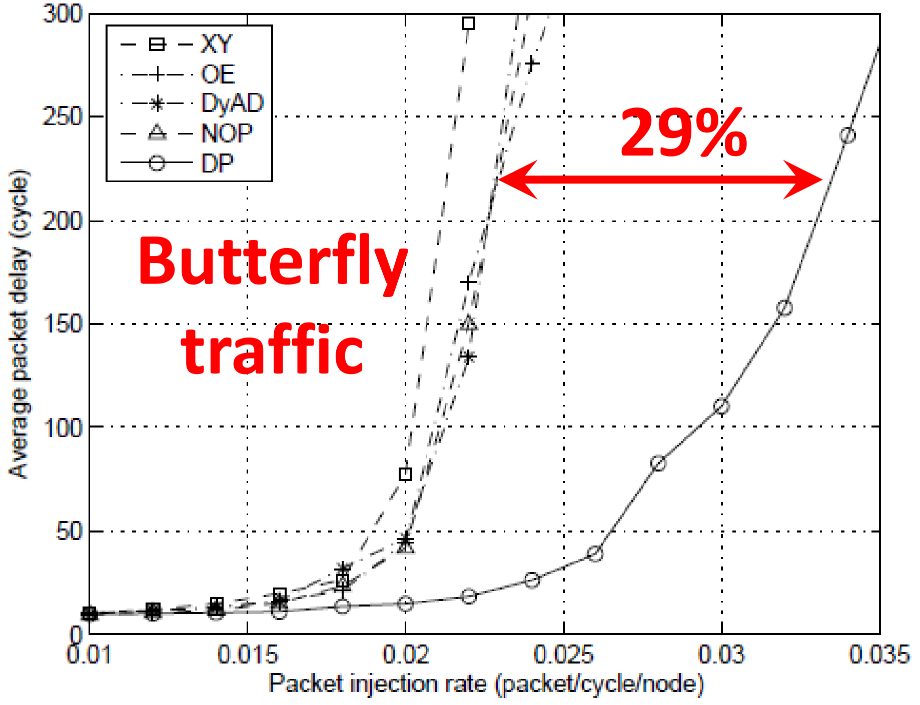
- Distributed execution
- Tightly coupled
- Optimal decision making

On-Chip Dynamic Programming Network for Routing (II)

- SystemC cycle accurate simulator
- Realizing different routing models and traffic benchmarks

Traffics	Packet injection rate (Packet/cycle/node$\times 10^{-3}$) XY	DyAD	Odd-Even	NoP	DP
Random-1	5.06	5.05	4.92	5.24	5.45
Random-2	3.07	3.12	4.52	4.54	5.08
Transpose	10.8	11.1	13.0	14.7	17.3
Butterfly	20.1	20.9	21.0	21.1	29.2
DP Improvement	28.9%	27.5%	18.4%	14.3%	





[T. Mak, *et al.*, *IEEE T-IE*, 2011]

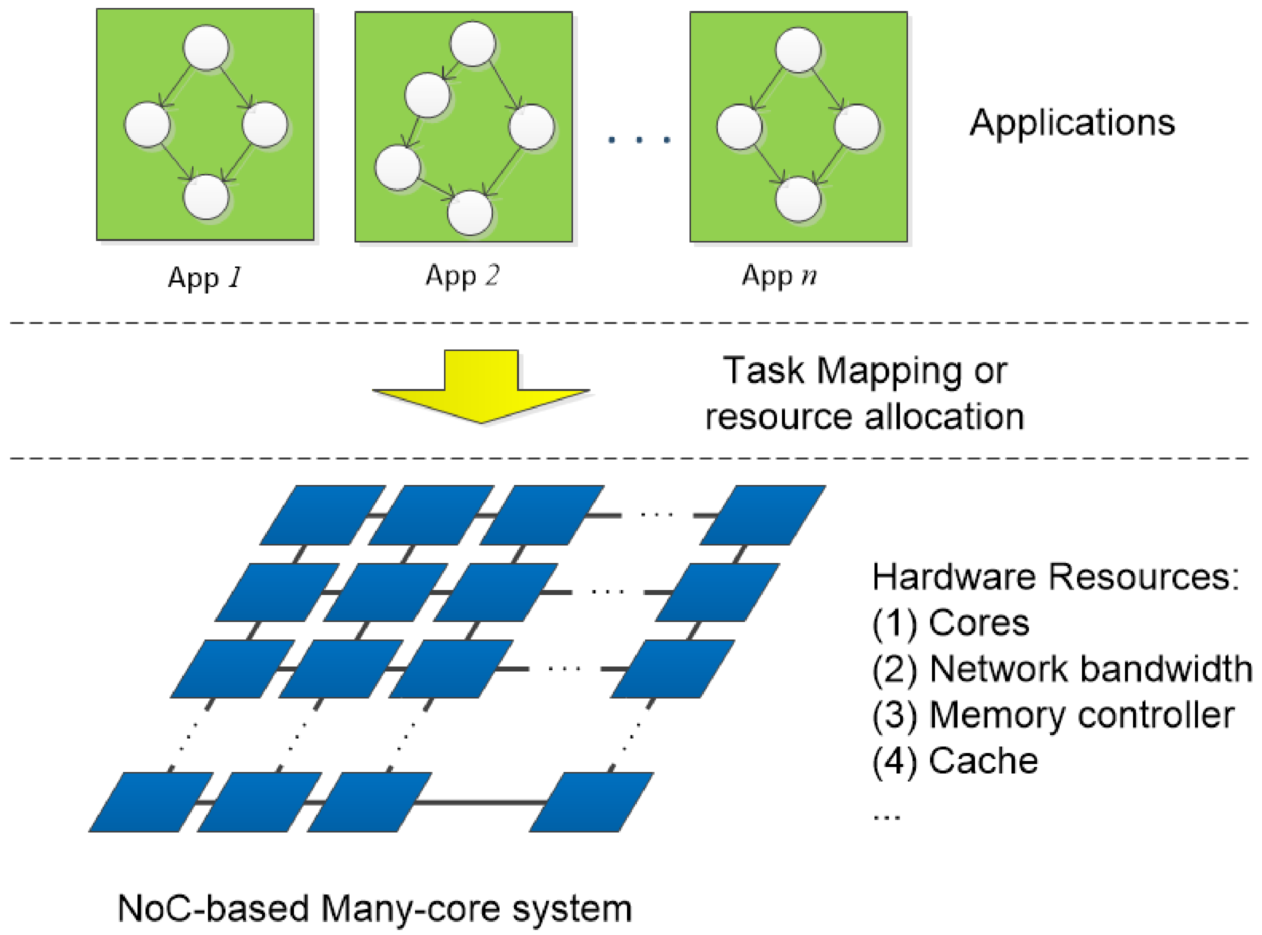
Content

- Previous work
- New challenge and motivation
- The Market-on-Chip architecture and operations
- Potentials and discussions

New Challenge

- Number of applications are increasing rapidly AND the **VARIATION** of cores in the application is huge
  - Mobile phone (from a phone to a multimedia platform)
  - Vehicle and airplane
  - Security systems (banking or exhibition)

Traditional Runtime Task Mapping

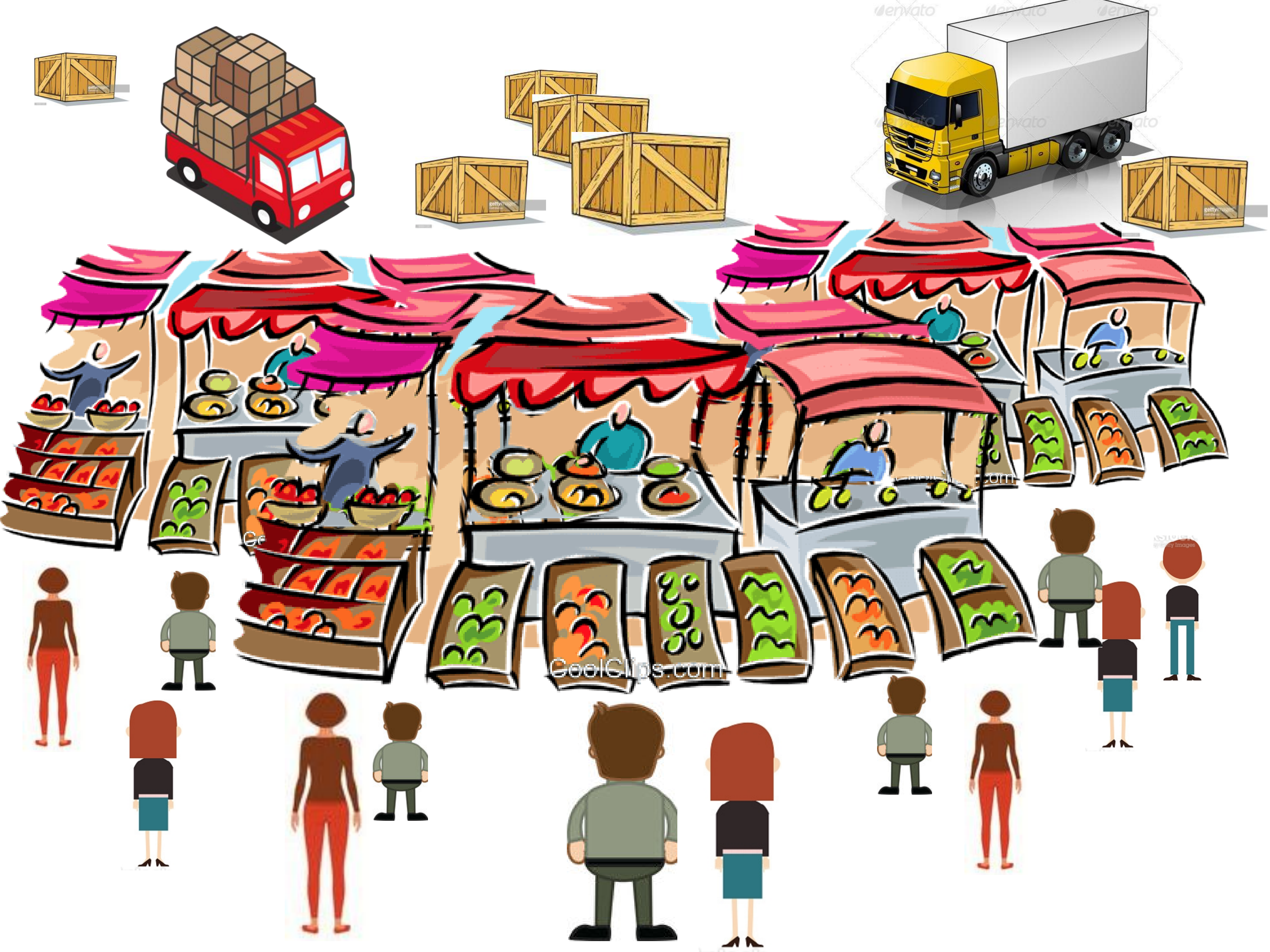


Traditional Runtime Task Mapping

- ***Offline***: Task mapping is fixed at compile-time
  - Complex heuristics algorithm
  - Not flexible
- ***Online***: Task mapping is determined when tasks arrive
  - *Less scalability*: Increased complexity with core count
  - *Sub-optimal*: Simple heuristics thus solution is far less optimal
  - *Not fair*: when hardware resource is not enough, some applications have to wait
  - *Less adaptive* to vary according to applications

Motivations

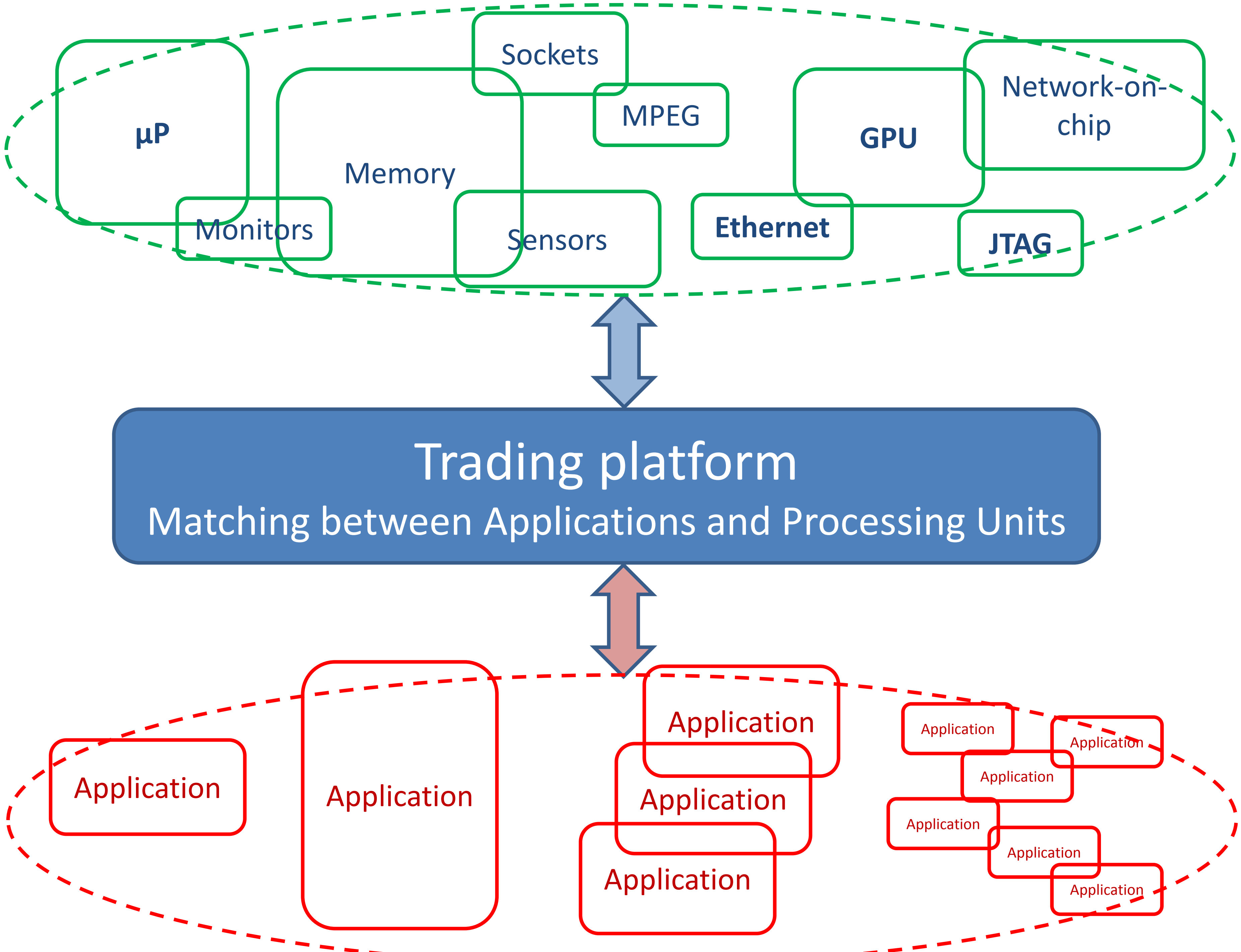
- How to handle ***variations?***
- ***We might need a bit of Heuristic***
- ***Super-Adaptive?***
- Quick and to avoid deadlock
- **Scalability (from hundred to thousands)**
  - Number of components and applicaitons
  - Variations



Supply

MARKET

Demand


Sockets
MPEG
μP
Memory
Monitors
Sensors
GPU
Network-on-chip
Ethernet
JTAG
Trading platform
Matching between Applications and Processing Units
Application
Application
Application
Application
Application
Application
Application
Application
Application
Application
Application
Application

Content

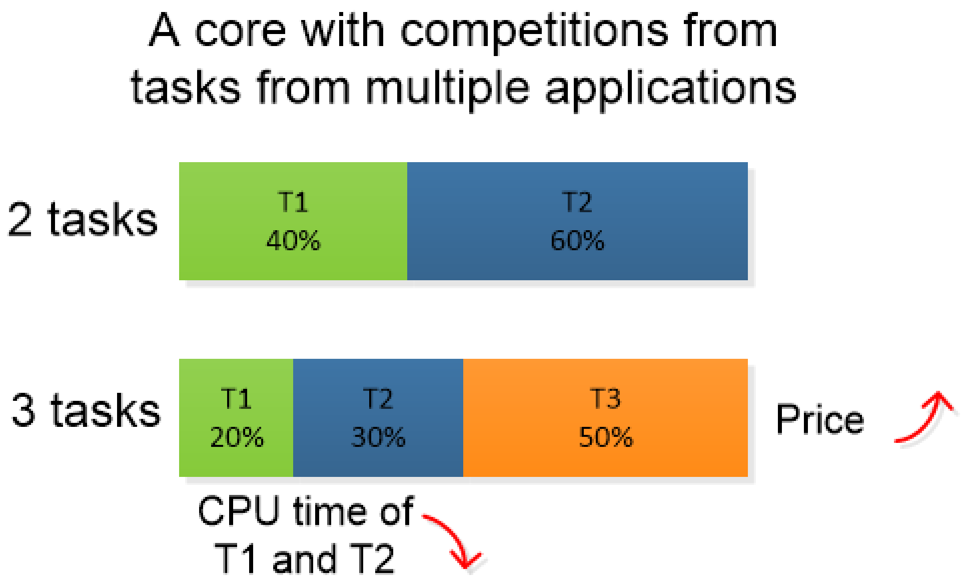
- Previous work
- New challenge and motivation
- The Market-on-Chip architecture and operations
- Potentials and discussions

Proposed: Market-on-Chip

- A ***market*** is to manage and control
  - how to efficiently allocate hardware resources to applications
  - Define rules
- ***Supply*** (Sell): multiple hardware resources
  - (Heterogeneous) cores, network bandwidth, memory controller, cache, etc.
- ***Demand*** (Buy): each application is an agent, bidding for the resources according to the resource requirement
  - Applications have different characteristics, i.e. require different amounts of resources
  - The requirement can vary with time

Market-on-Chip
Price and ***Waiting time***

- Competing a resource based on ***Price***



Task 1 and Task 2 bid some money for a core. When a task is running, the money is consumed until the money is run out. Then the task is stalled to save money and the core is switched to task 2. In the example, the CPU time of task 1 is 40%.

When task 3 arrives, the core receives more biddings. The price of the core increases. The money is run out faster. The CPU time of tasks decrease to 20%.

Proposed: Market-on-Chip

- **Hierarchical arrangement**
  - A centralized agent (government) determines the prices based on resource demand
  - Distributed decision making: each agent requests for the resources and makes decisions in distributed manner (more scalable)

More to Consider

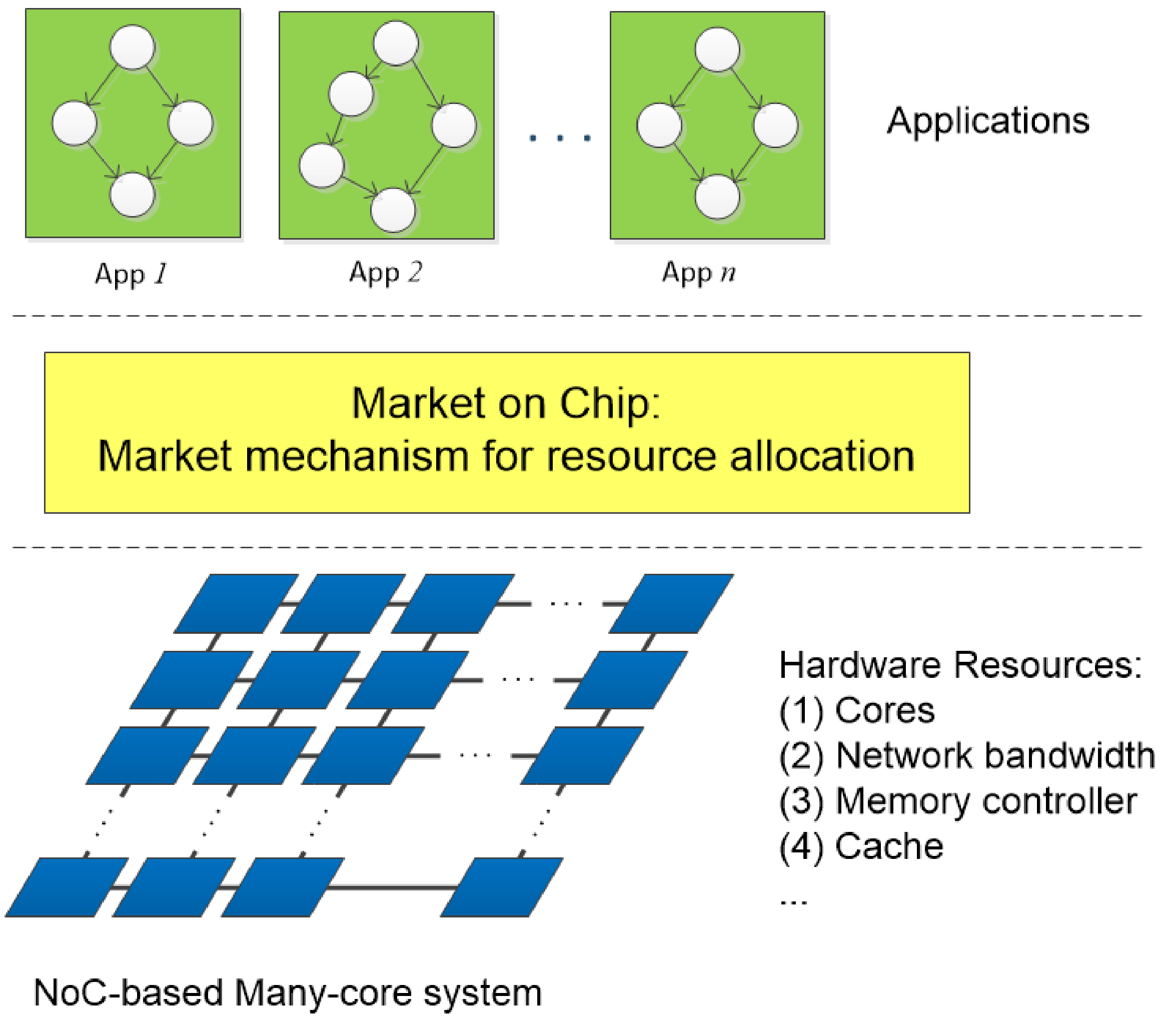
- How to design the market to make it simple and efficient
- Other constraints such as thermal design power (TDP), reliability, throughput, etc.
- How to define the utility function of each agent

Conclusion

- ***Variation*** keeps going up, we need to think about how to handle it.
- Analogue of market – supply and demand
- Inside an FPGA – ***components*** and ***applications***
- Make use of ***Price (updated)*** for exchange and ***Waiting time***

END

Proposed: Market-on-Chip



App 1
App 2
App n
Applications
Market on Chip:
Market mechanism for resource allocation
Hardware Resources:
(1) Cores
(2) Network bandwidth
(3) Memory controller
(4) Cache
...
NoC-based Many-core system