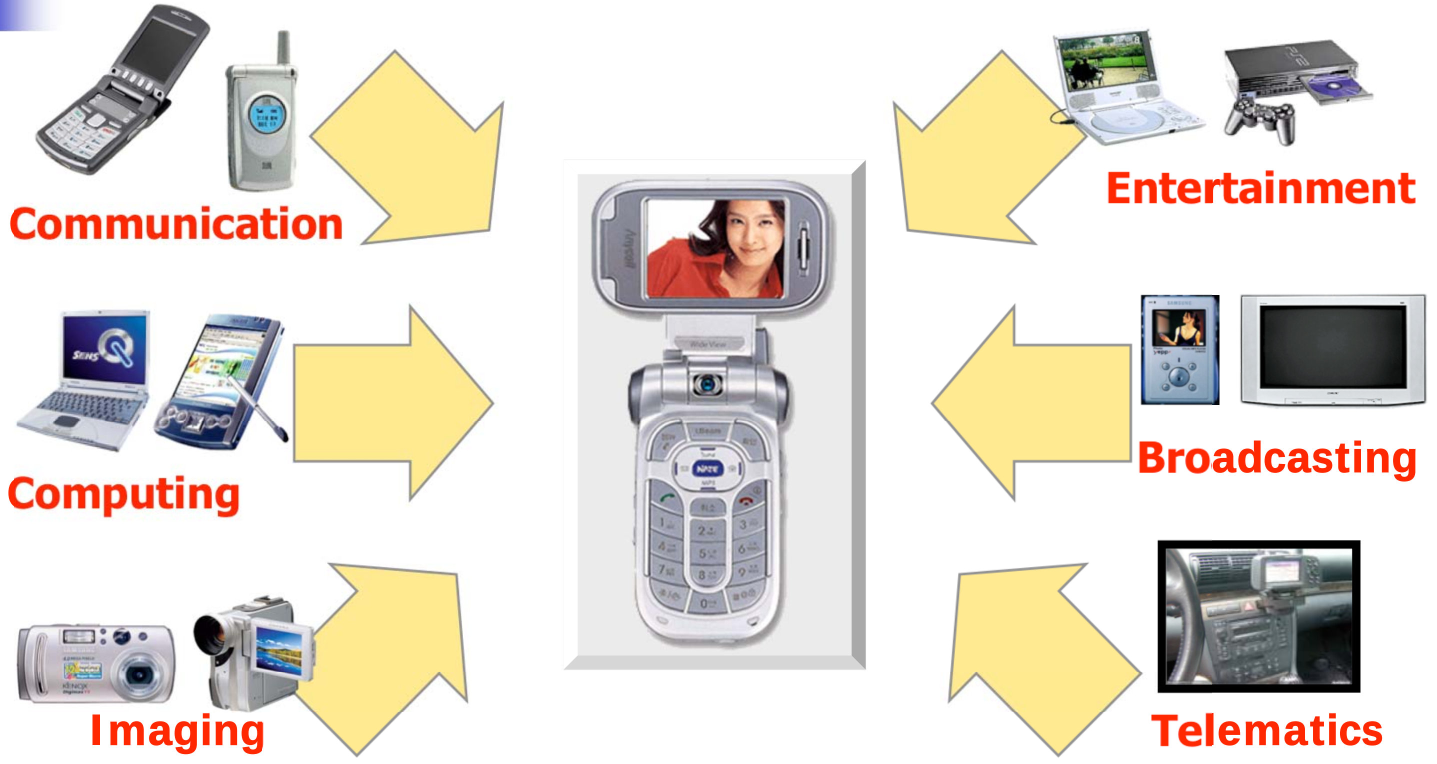
Tecniche di ottimizzazione per lo sviluppo di applicazioni embedded su piattatforme multiprocessore su singolo chip

Michela Milano

mmilano@deis.unibo.it

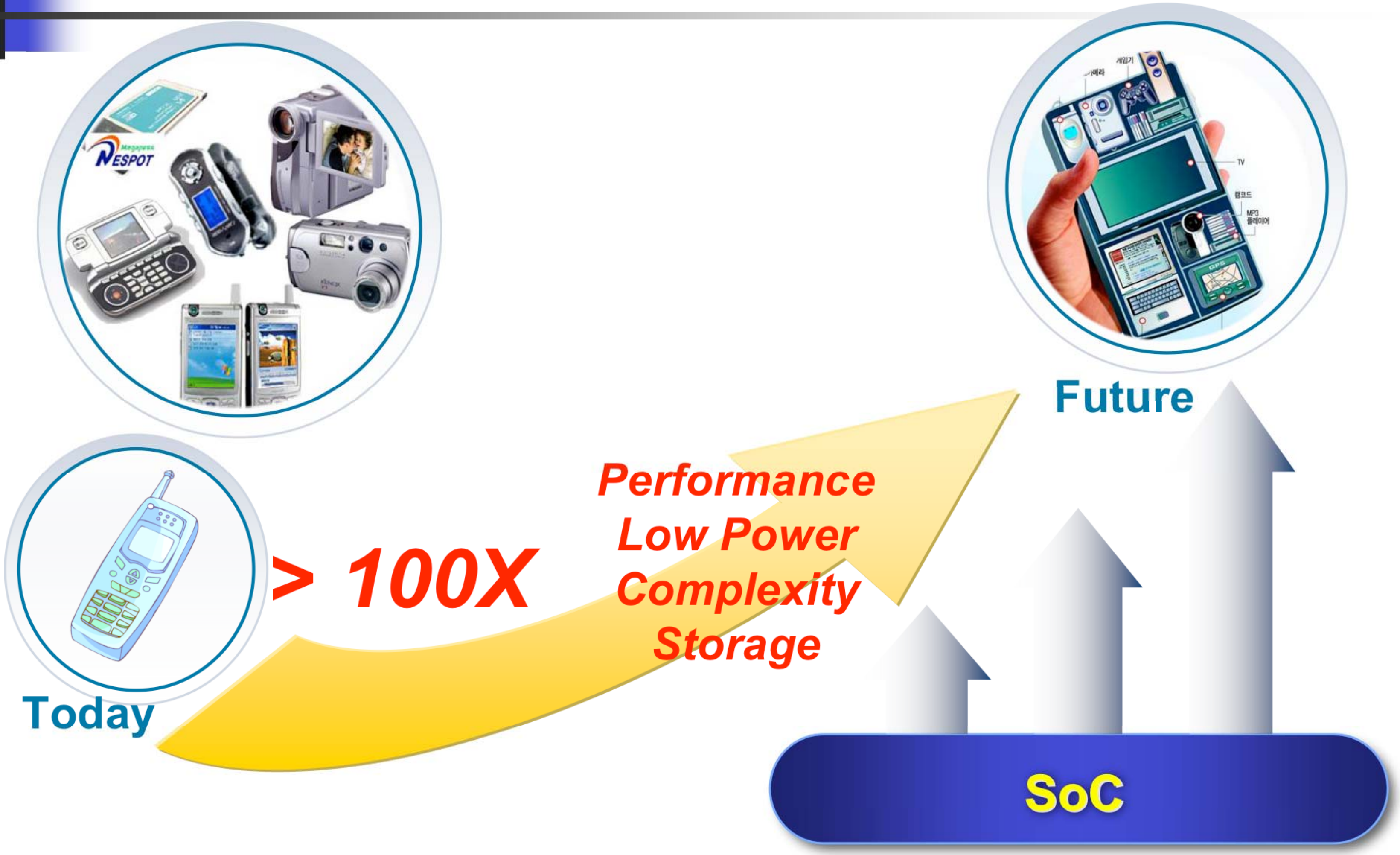
DEIS Università di Bologna

Digital Convergence – Mobile Example



- One device, multiple functions
- Center of ubiquitous media network
- Smart mobile device: next drive for semicon. Industry

SoC: Enabler for Digital Convergence



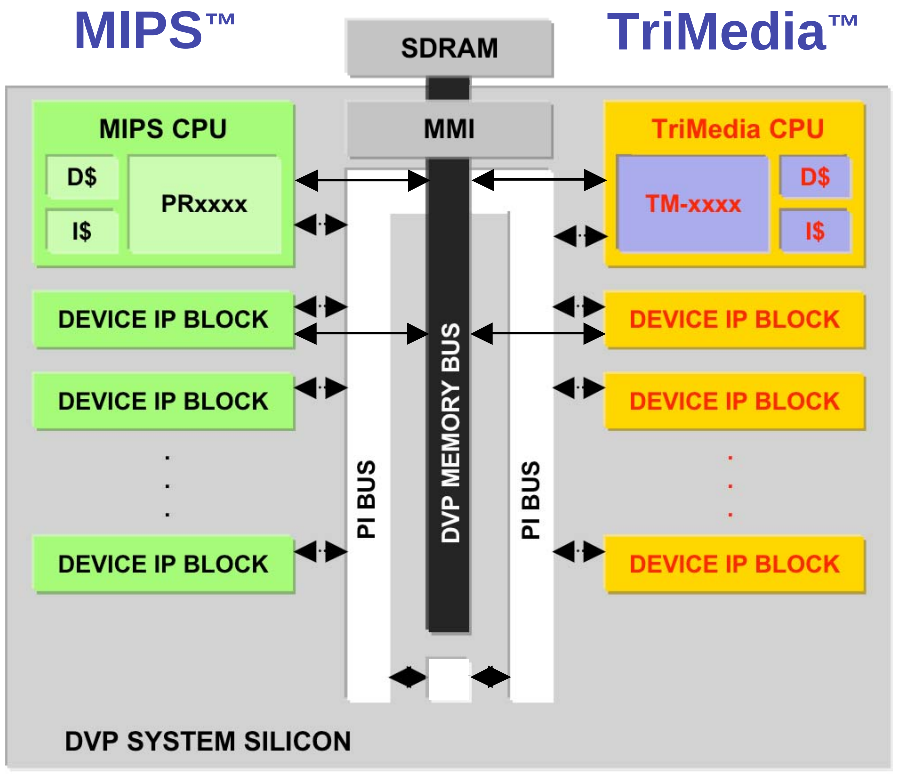
Systems on chip

- Moore’s law provides exponential growth of resources
- But design does not become easier
  - ➔ Deep submicron problems (DSM)
    - Wire vs. transistor speed, power, signal integrity
  - ➔ Design productivity gap
    - IP re-use, platforms, NoCs
    - Verification technologies

The evolution of SoC platforms

General-purpose
Scalable RISC
Processor
• 50 to 300+ MHz
• 32-bit or 64-bit

Library of Device
IP Blocks
• Image coprocessors
• DSPs
• UART
• 1394
• USB
…



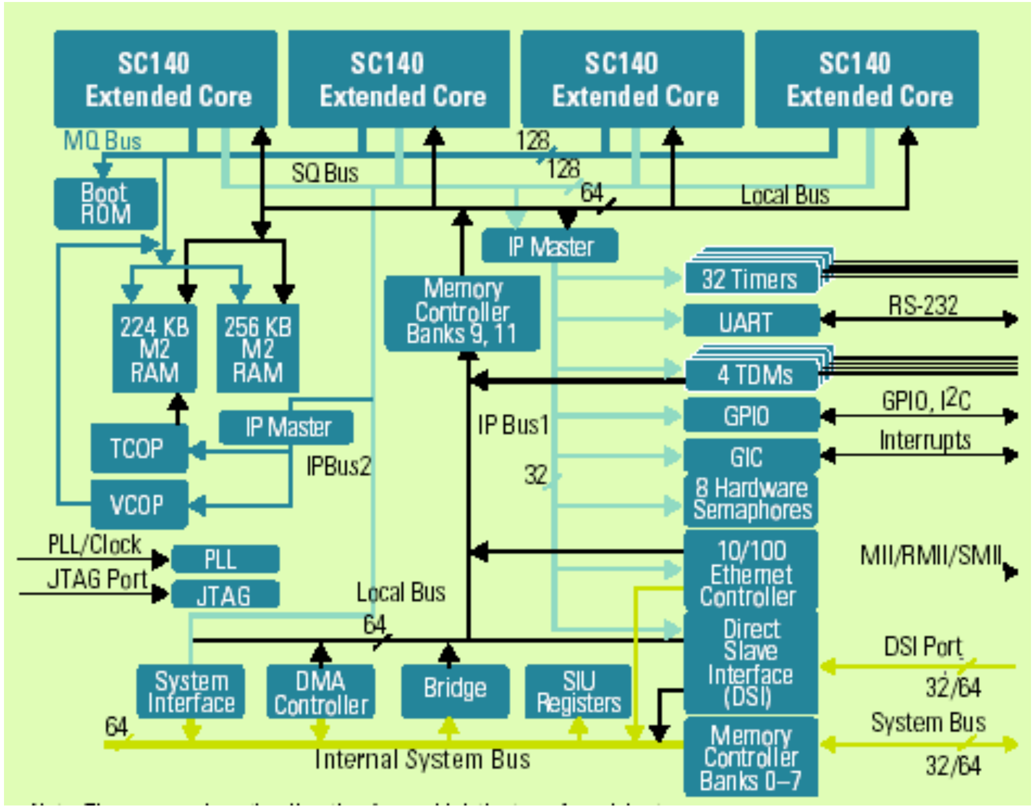
Scalable VLIW
Media Processor:
• 100 to 300+ MHz
• 32-bit or 64-bit

Nexperia™
System Buses
• 32-128 bit

- ***2 Cores***: Philips' Nexperia PNX8850 SoC platform for High-end digital video (2001)

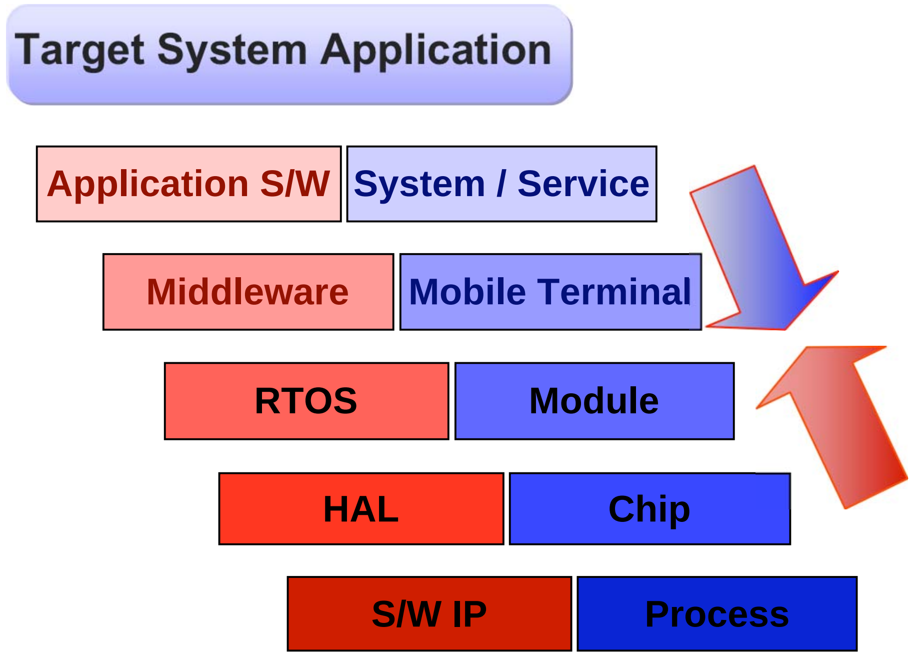
Running forward...

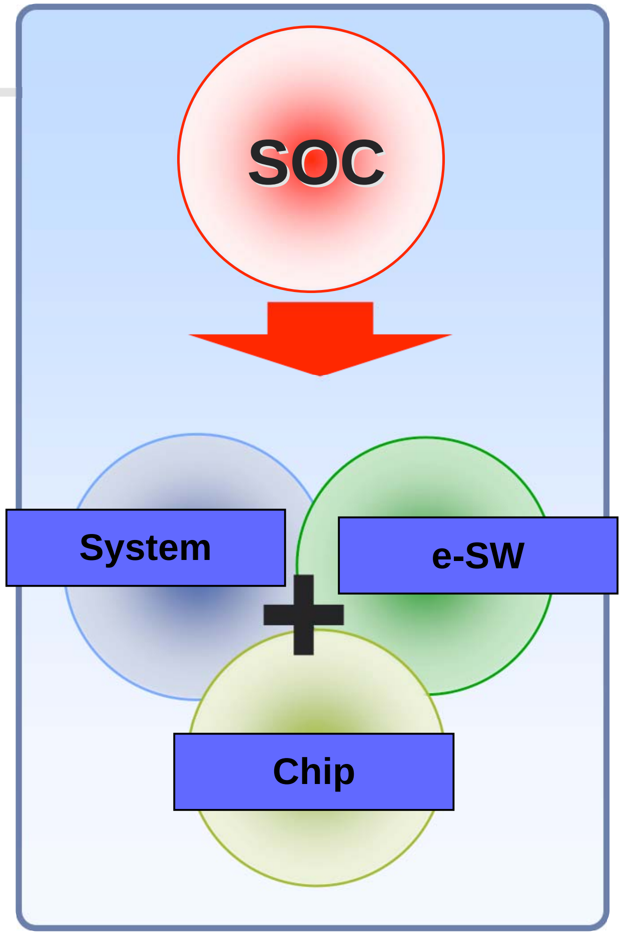
- Four 350/400 MHz StarCore SC140 DSP extended cores
- 16 ALUs: 5600/6400 MMACS
- 1436 KB of internal SRAM & multi-level memory hierarchy
- Internal DMA controller supports 16 TDM unidirectional channels,
- Two internal coprocesssors (TCOP and VCOP) to provide special-purpose processing capability in parallel with the core processors



- ***6 Cores***: Motorola's MSC8126 SoC platform for 3G base stations (late 2003)

SoC → Solution-on-a-Chip





Requires design of Hardware AND software

Design as optimization

- Design space

  The set of “all” possible design choices

- Constraints

  Solutions that we are not willing to accept

- Cost function

  A property we are interested in (execution time, power, reliability…)

Optimization techniques

- We will consider two techniques:
  - **Constraint Programming**
  - **Integer Programming**
- Two aspects of a problem :
  - **Feasibility**
  - **Optimality**
- Merging the two, one could obtain better results

Pros and Cons

- ☺ Declarative programming: the user states the constraint the solver takes into account propagation and search
- ☺ Strong on the feasibility side
- ☺ Constraints are symbolic and mathematic: expressivity
- ☺ Adding a constraint helps the solution process: flexibility

- ☹ Weak optimality pruning if the link between problem decision variables and the objective function is loose
- ☹ No use of relaxations

Example

Weak optimality pruning if the link between problem decision variables and the objective function is loose

- Scheduling problems: minimizing makespan
  - Problem variables are starting times of activities, the last activity provides the makespan
- Scheduling problems: minimizing allocation cost
  - Problem variables are starting times of activities and resources, the sum of resource assignment cost is the objective function

    OF = C1+ C2+C3+C4 and all cost range from [1..50] suppose we have a upper bound on the problem of 70. Nothing can be pruned
- The situation is even worse if the OF depends on couples of assignments

Integer Programming

- Standard form of Combinatorial Optimization Problem (IP)
  - min $z = \sum_{j=1}^{n} c_j x_j$
  - subject to

    $\sum_{j=1}^{n} a_{ij} x_j = b_i \qquad i = 1..m$

    $x_j \geq 0 \qquad j = 1..n$

    x_j integer $\longleftarrow$ May make the problem NP complete

- Inequality $y \geq 0$ recasted in $y - s = 0$
- Maximization expressed by negating the objective function

0-1 Integer Programming

- *Many Combinatorial Optimization Problem can be expressed in terms of 0-1 variables (IP)*

  - min $z = \sum_{j=1}^{n} c_j x_j$
  - subject to

    $\sum_{j=1}^{n} a_{ij} x_j = b_i \quad i = 1..m$

    $x_j : [0,1]$ $\Leftarrow$ May make the problem NP complete

Linear Relaxation

- $\min \; z = \sum_{j=1}^{n} c_j x_j$
- subject to

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \qquad i = 1..m$$

$$x_j \geq 0 \qquad j = 1..n$$

Linear Relaxation

xj integer Removed

- The linear relaxation is solvable in POLYNOMIAL TIME
- The SIMPLEX ALGORITHM is the technique of choice even if it is exponential in the worst case

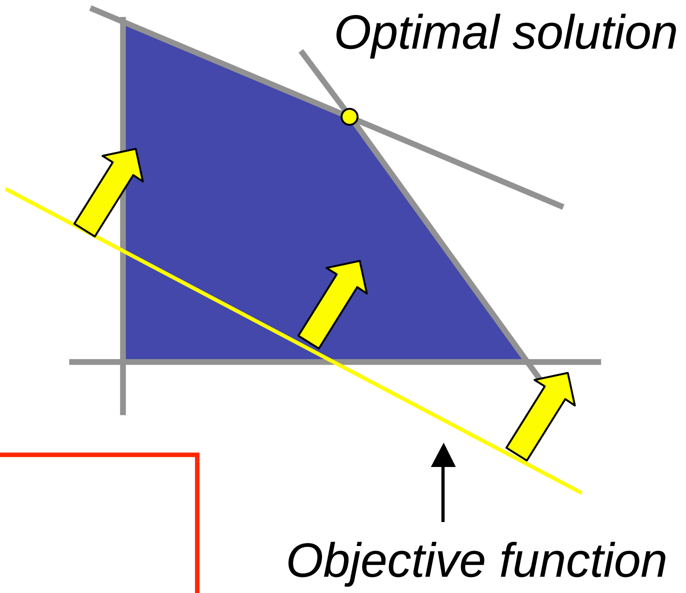
Geometric Properties

- The set of constraints defines a polytope
- The optimal solution is located on one of its vertices

  - $\min \; z = \sum_{j=1}^{n} c_j x_j$
  - subject to

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \qquad i = 1..m$$

$$x_j \geq 0 \qquad j = 1..n$$



The simplex algorithm starts from one vertex and moves to an adjacent one with a better value of the objective function

IP solving process

- The optimal LP solution is in general fractional: violates the integrality constraint but provides a bound on the solution of the overall problem

- The solution process interleaves branch and bound:
  - relaxation
  - search

Pros and Cons

- ☺ Declarative programming: the user states the constraint the solver takes into account relaxation and search
- ☺ Strong on the optimality side
- ☺ Many real problem structure has been deeply studied

- ☹ Only linear constraints should be used
- ☹ If sophisticated techniques are used, we lose flexibility
- ☹ No pruning on feasibility (only some preprocessing)

Resource-Efficient Application mapping for MPSoCs

Given a platform

- **Achieve a specified throughput**
- **Minimize usage of shared resources**

Allocation and scheduling

- Given:
  - An hardware platform with processors, (local and remote) storage devices, a communication channel
  - A pre characterized task graph representing a functional abstraction of the application we should run
- Find:
  - An allocation and a scheduling of tasks to resources respecting
    - Real time constraints
    - Task deadlines
    - Precedences among tasks
    - Capacity of all resources
- Such that
  - the communication cost is minimized

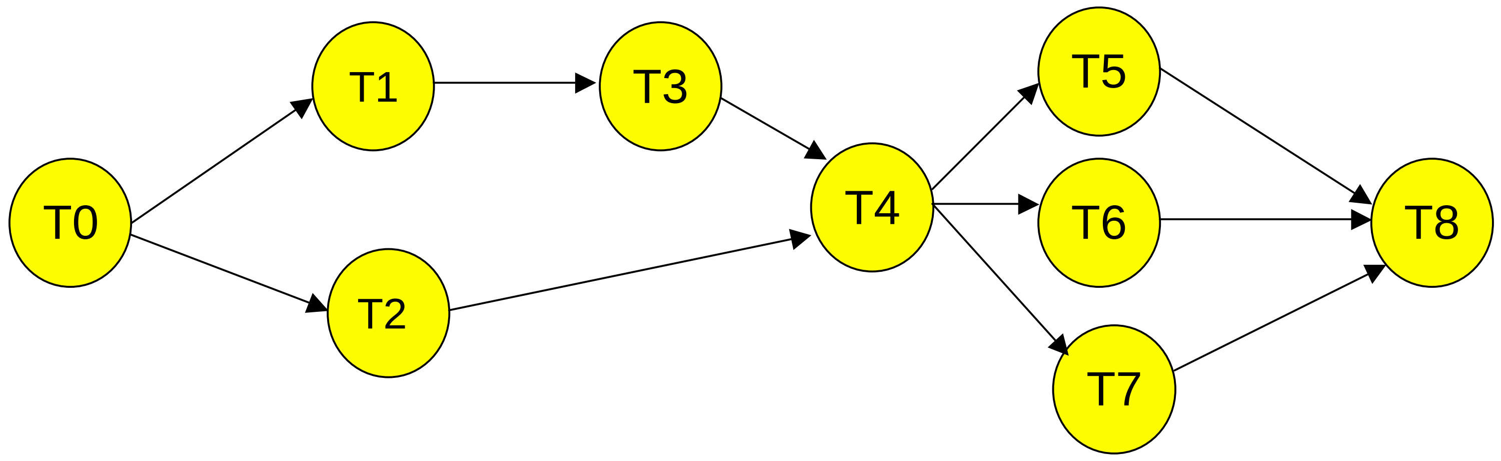
Allocation and scheduling

- The platform is a multi-processor system with N nodes
  - Each node includes a processor and a schretchpad memory
  - The bus is a shared communication channel
  - In addition we have a remote memory of unlimited capacity (realistic assomption for our application, but easily generalizable)

- The task graph can be of any kind. In this case it has a pipeline workload
  - Real time video graphic processing pixel of a digital image
  - Task dependencies, i.e., arcs between tasks
  - Computation, communication, storage requirements on the graph

The application

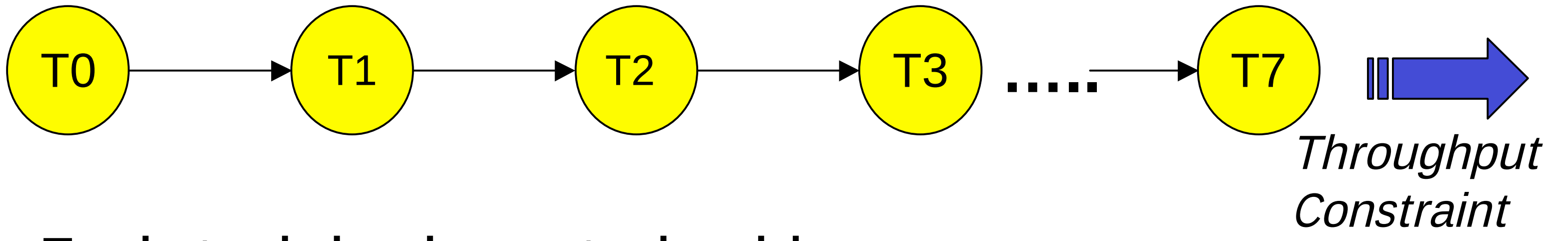
Generic Taks Graph: nodes are functional asbstractions, arcs represent communications


$$\begin{tikzcd}
 & T1 \arrow[r] & T3 \arrow[dr] & & T5 \arrow[dr] & \\
T0 \arrow[ur] \arrow[dr] & & & T4 \arrow[ur] \arrow[r] \arrow[dr] & T6 \arrow[r] & T8 \\
 & T2 \arrow[urr] & & & T7 \arrow[ur] &
\end{tikzcd}$$

Conditional Taks Graph: not all nodes are executed. They model if then else behaviour

The application

Signal Processing Pipeline

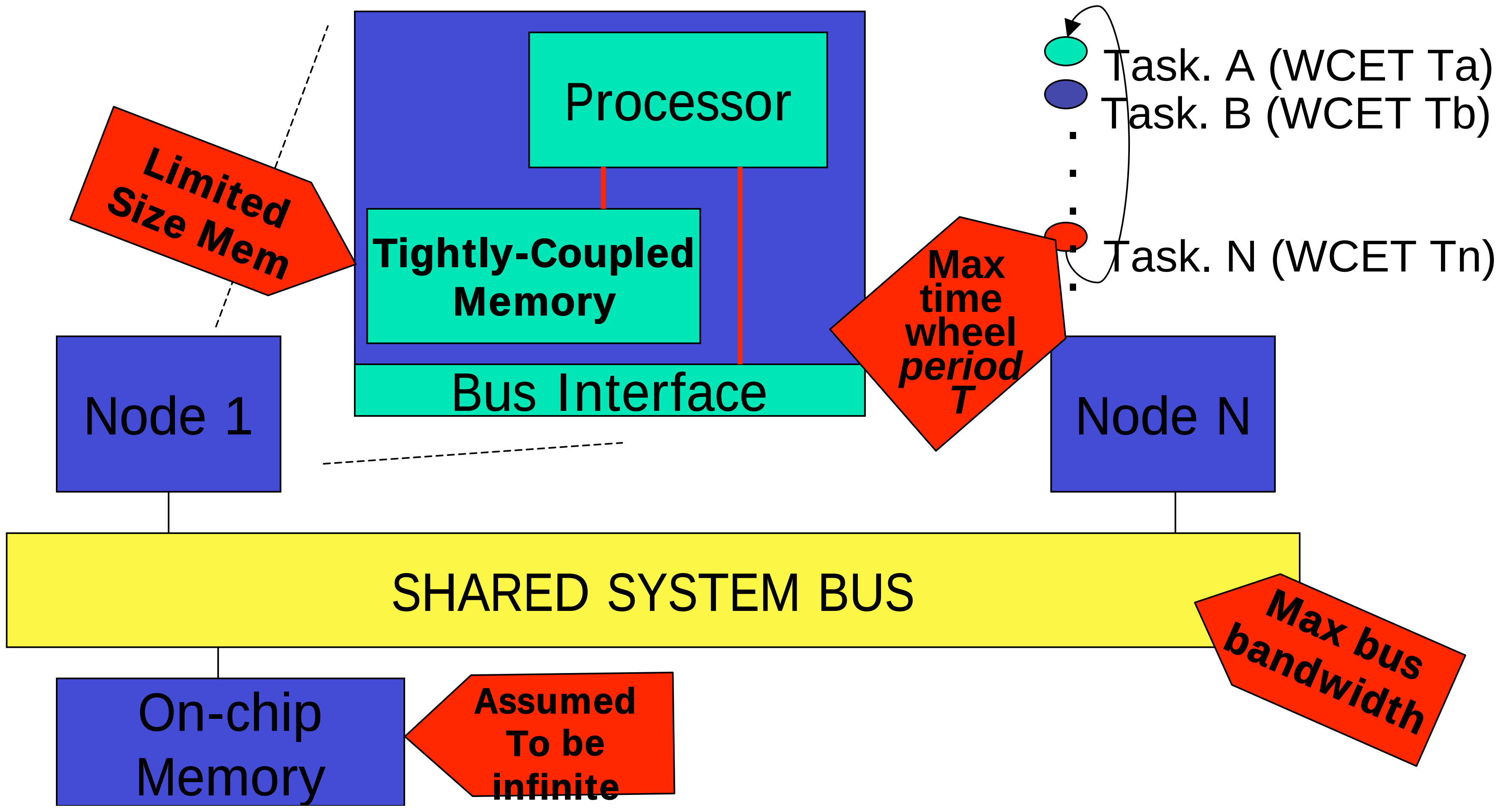


Each task is characterized by:

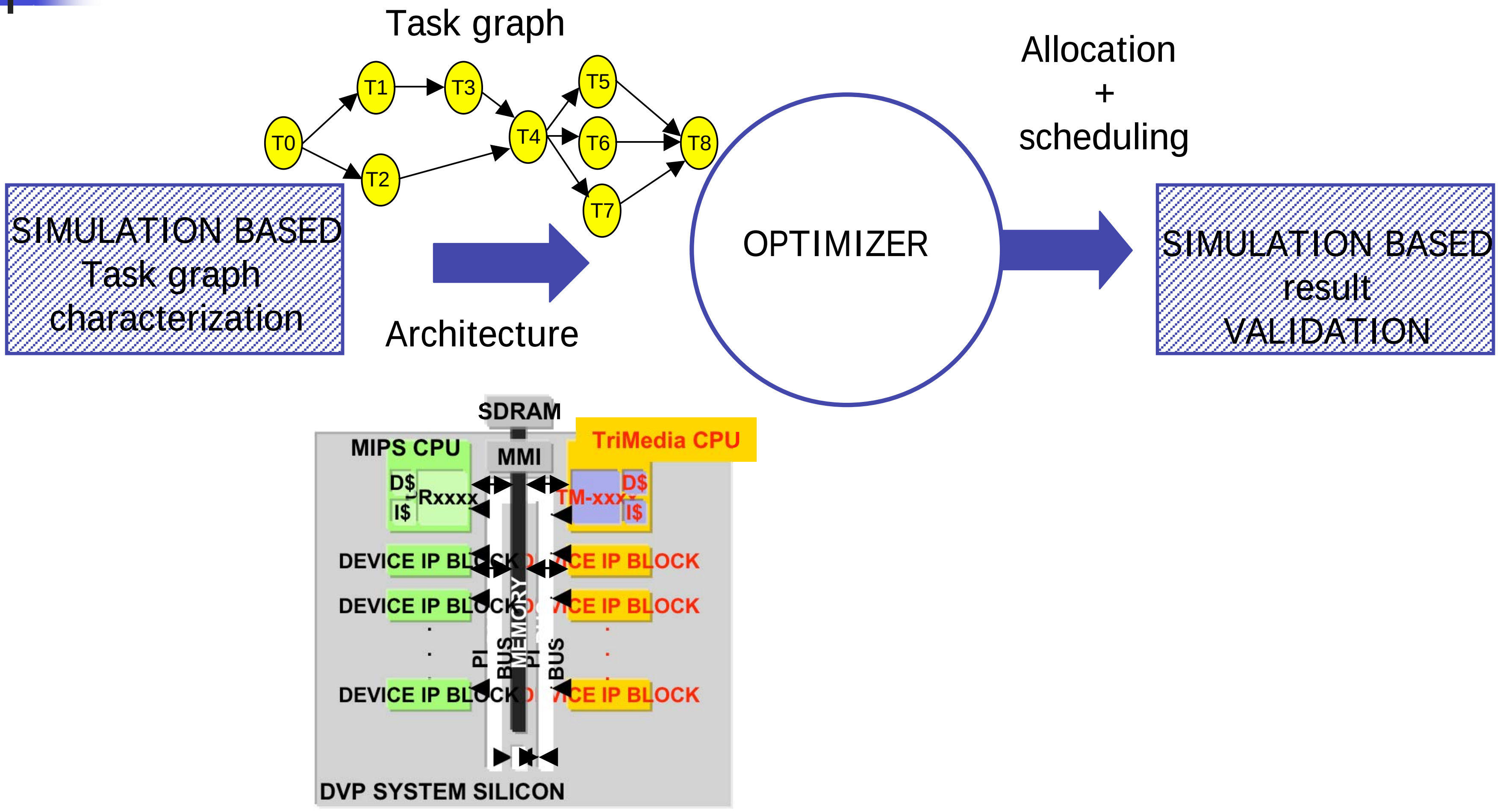
- WCET
- Memory requirements
  - ➢ Queues for inter-processor communication
    in TCM for efficiency reasons
  - ➢ Program data
    in TCM (if space) or on-chip memory
  - ➢ Internal state
    in TCM (if space) or on-chip memory

Resource assignment and scheduling

THE SYSTEM



Our approach



Previous approaches

- Complete approaches: find the optimal solution and prove its optimality
  - The System Design community uses Integer Programming techniques for every optimization problem despite the structure of the problem itself
  - Scheduling is poorly handled by IP
- Incomplete approaches: find a good solution, in general a local minima
  - Many of such approaches
  - Require a lot of tuning

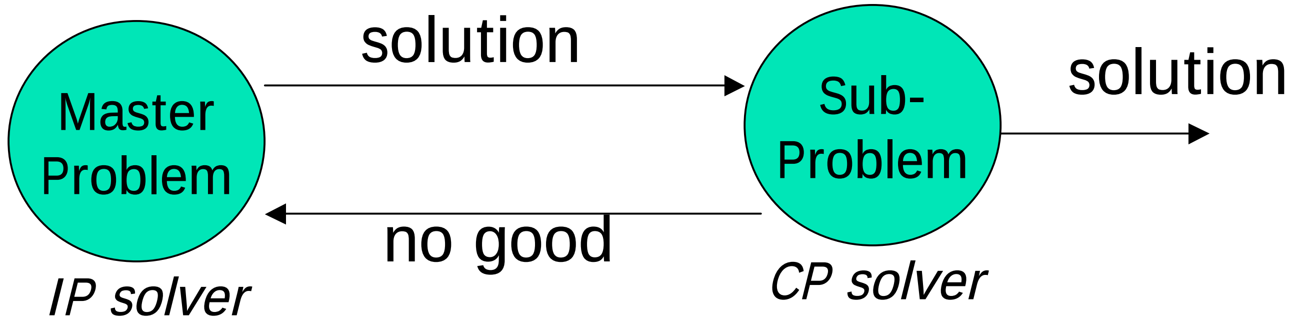
Our approach

- Let us analyze the structure of the problem:
  - As a whole it is a scheduling problem with alternative resources: very tough problem
  - It smoothly decomposes into allocation and scheduling
    - Allocation better handled with IP techniques
      - Not with CP due to the complex objective function
    - Scheduling better handled with CP techniques
      - Not with IP since we should model for each task all its possible starting time with a 0/1 variable
    - *INTERACTION REGULATED VIA NO-GOOD*

Problem decomposition

Objective function:
Min(Communication Cost)

- Assignment of tasks and memory slots *(master problem)*
  - ✓ Obj. Func. Relates alternative resources to couples of tasks
  - ✓ Not a good scenario for Constraint Programming
- Task scheduling with static resource assignment *(subproblem)*
  - ✓ Integer Programming does not handle time efficiently
  - ✓ Constraint Programming is instead effective



Master Problem model

- Assignment of tasks and memory slots *(master problem)*
  - ✓ T_{ij}= 1 if task i executes on processor j, 0 otherwise,
  - ✓ Y_{ij} =1 if task i allocates program data on processor j memory, 0 otherwise,
  - ✓ Z_{ij} =1 if task i allocates the internal state on processor j memory, 0 otherwise
  - ✓ X_{ij} =1 if task i executes on processor j and task i+1 does not, 0 otherwise

✓ Each process should be allocated to one processor $\sum_i T_{ij} = 1$ for all j

✓ Link between variables X and T: $X_{ij} = |T_{ij} - T_{i+1\,j}|$ for all i and j (can be linearized)

✓ If a task is NOT allocated to a processor nor its required memories are:

$$T_{ij} = 0 \Rightarrow Y_{ij} = 0 \text{ and } Z_{ij} = 0$$

✓Objective function $\sum_i \sum_j \text{mem}_i (T_{ij} - Y_{ij}) + \text{state}_i (T_{ij} - Y_{ij}) + \text{data}_i X_{ij} /2$

Improvement of the model

- With the proposed model, the allocation problem solver tends to pack all tasks on a single processor and all memory required on the local memory so as to have a ZERO communication cost: TRIVIAL SOLUTION

- To improve the model we should add a relaxation of the subproblem to the master problem model:
  - For each set S of consecutive tasks whose sum of durations exceeds the Real time requirement, we impose that their processors should not be the same

$$\sum_{i \in S} WCET_i > RT \Rightarrow \sum_{i \in S} T_{ij} \leq |S| - 1$$

Sub-Problem model

- Task scheduling with static resource assignment *(subproblem)*
- We have to schedule tasks so we have to decide when they start
  - Activity Starting Time: $Start_i :: [0..Deadline_i]$
  - Precedence constraints: $Start_i + Dur_i \leq Start_j$
  - Real time constraints: for all activities running on the same processor

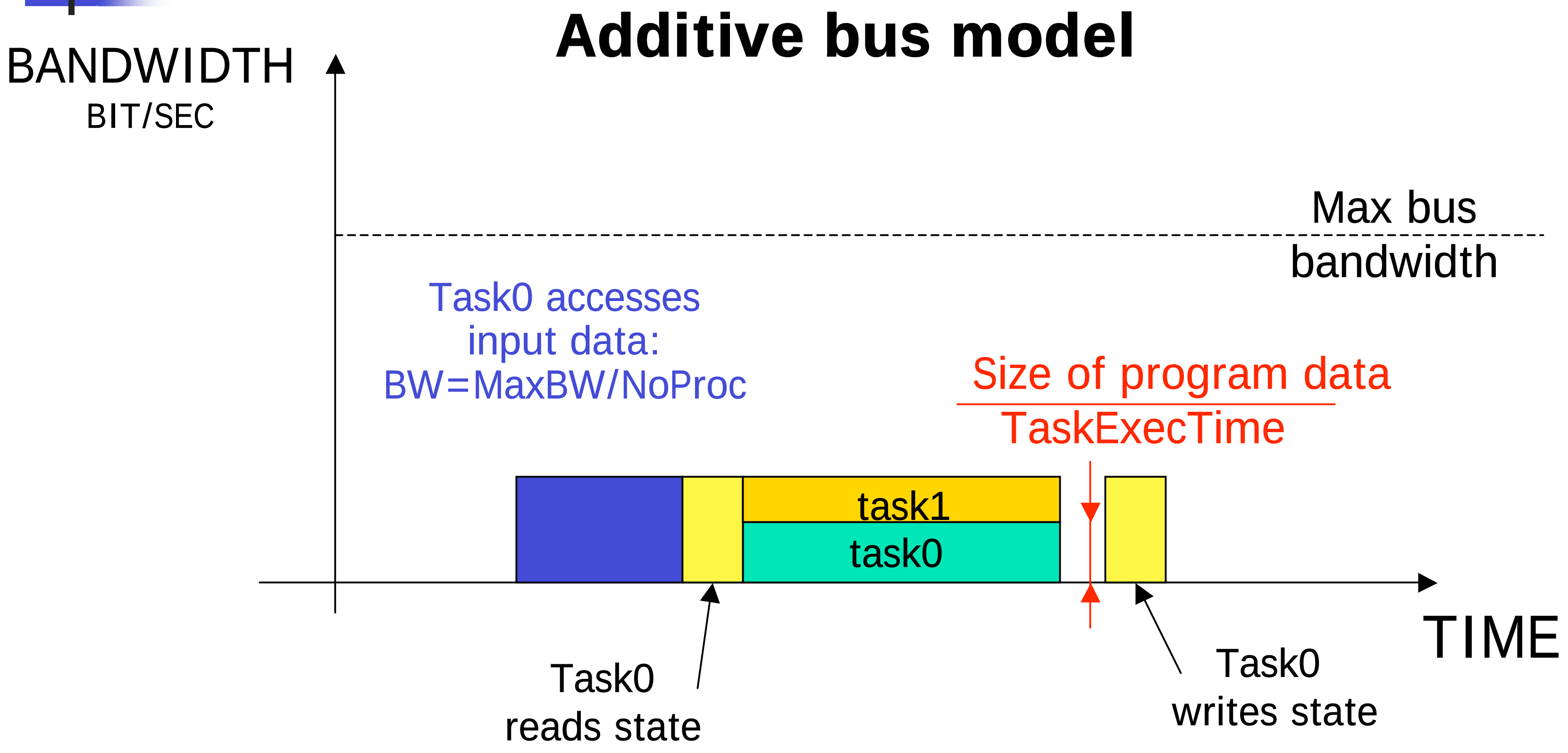
$$\sum_i (Start_i + Dur_i) \leq RT$$

  - Cumulative constraints on resources

    processors are unary resources: cumulative([Start], [Dur], [1],1)
    memories are additive resources: cumulative([Start],[Dur],[MR],C)

    What about the bus??

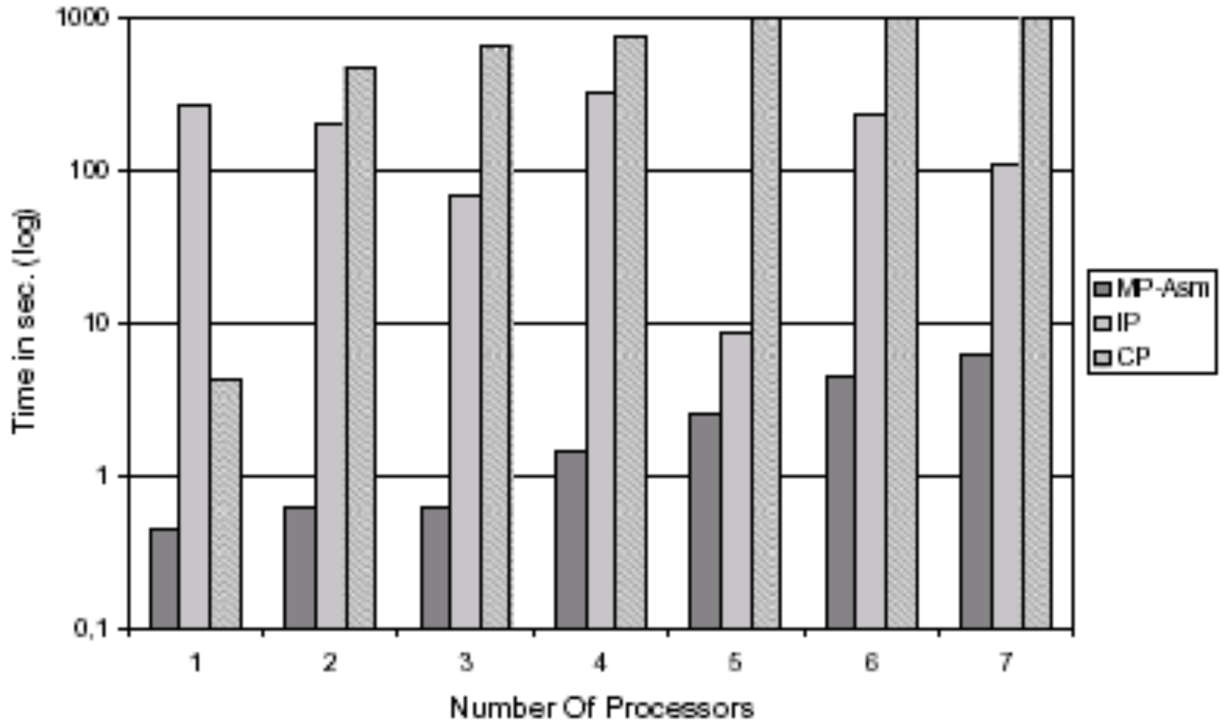
Bus model



The model does not hold under heavy bus congestion
Bus traffic has to be minimized

Results

Algorithm search time



The combined approach dominates, and its higher complexity comes out only for simple system configurations

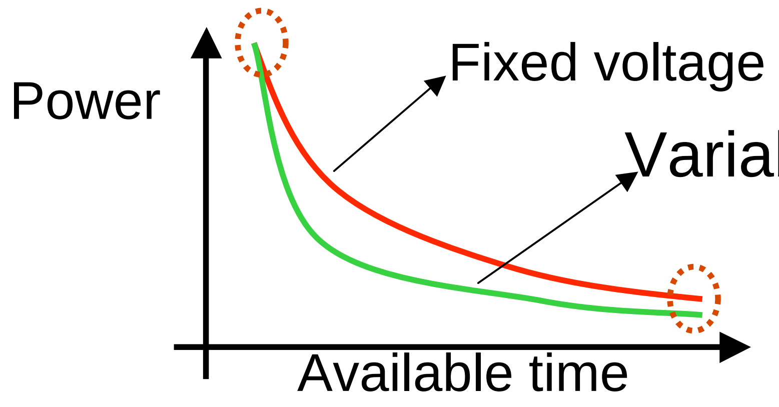
Energy-Efficient Application mapping for MPSoCs

Given a platform

- **Achieve a specified throughput**
- **Minimize power consumption**

Exploiting Voltage Supply

- Supply voltage impacts power and performance
  - Circuit slowdown $T=1/f=K/(V_{dd}-V_t)^a$
  - Cubic power savings $P=C_{eff}*V_{dd}^2*f$
- Just-in-time computation
  - Stretch execution time up to the max tolerable



In recent MPSoCs it is possible to change the processors working frequency.

Dynamic Voltage Scaling Problem (DVSP)

Given

- An hardware platform with processors, a communication channel, a set of discrete frequencies and the power consumption at each frequency,
- A pre characterized task graph representing a functional abstraction of the application we should run

Find

- An allocation and a scheduling of tasks to resources and of frequencies to tasks respecting
  - ✓ Real time constraints
  - ✓ Task deadlines
  - ✓ Precedence constraints among tasks
  - ✓ Capacity of all resources

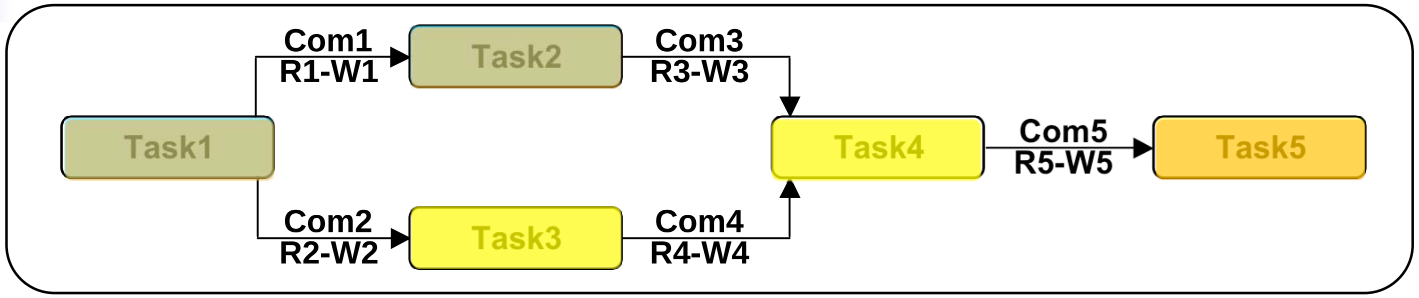
Such that

- the total power consumption is minimized. Power is consumed when a task executes, when 2 tasks communicate and when a processor changes its frequency.

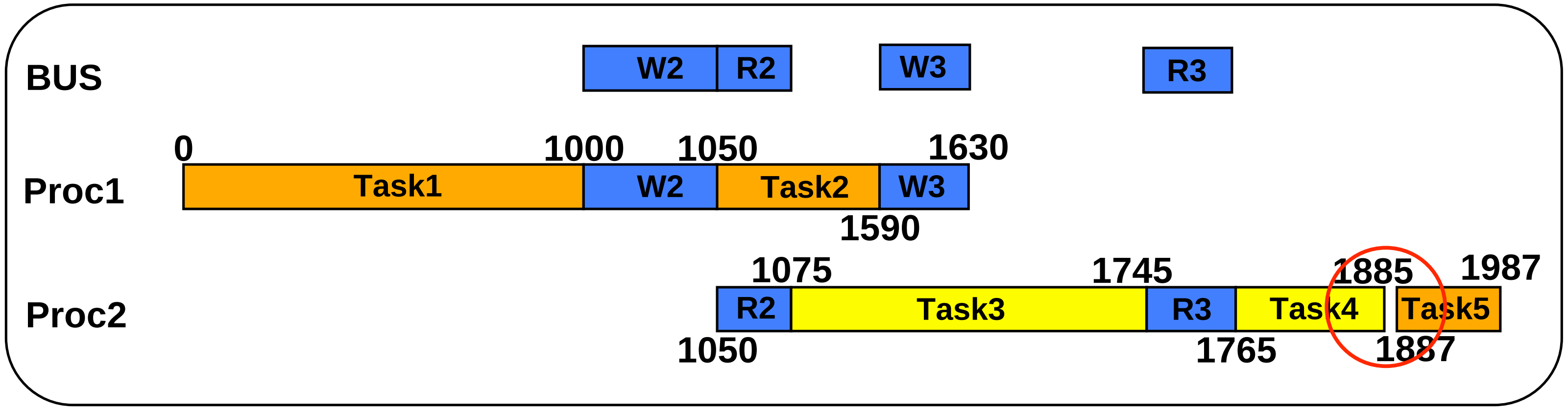
Example of DVSP

2 cores, running at 200MHz or 100MHz. The power consumption is 10mW at 200MHz and 3mW at 100MHz.

Switching from 200 MHz to 100 MHz needs 2ns and 2pJ, while the opposite needs 3ns and 3pJ.



Name	Task1	Task2	Task3	Task4	Task5	Com1	Com2	Com3	Com4	Com5
Duration	100	54	134	24	10	20	10	8	8	8



Task1, Task2 and Task5 at 200MHz.
Task3 and Task4 at 100 MHz.

Our approach

Let us analyze the structure of the problem:

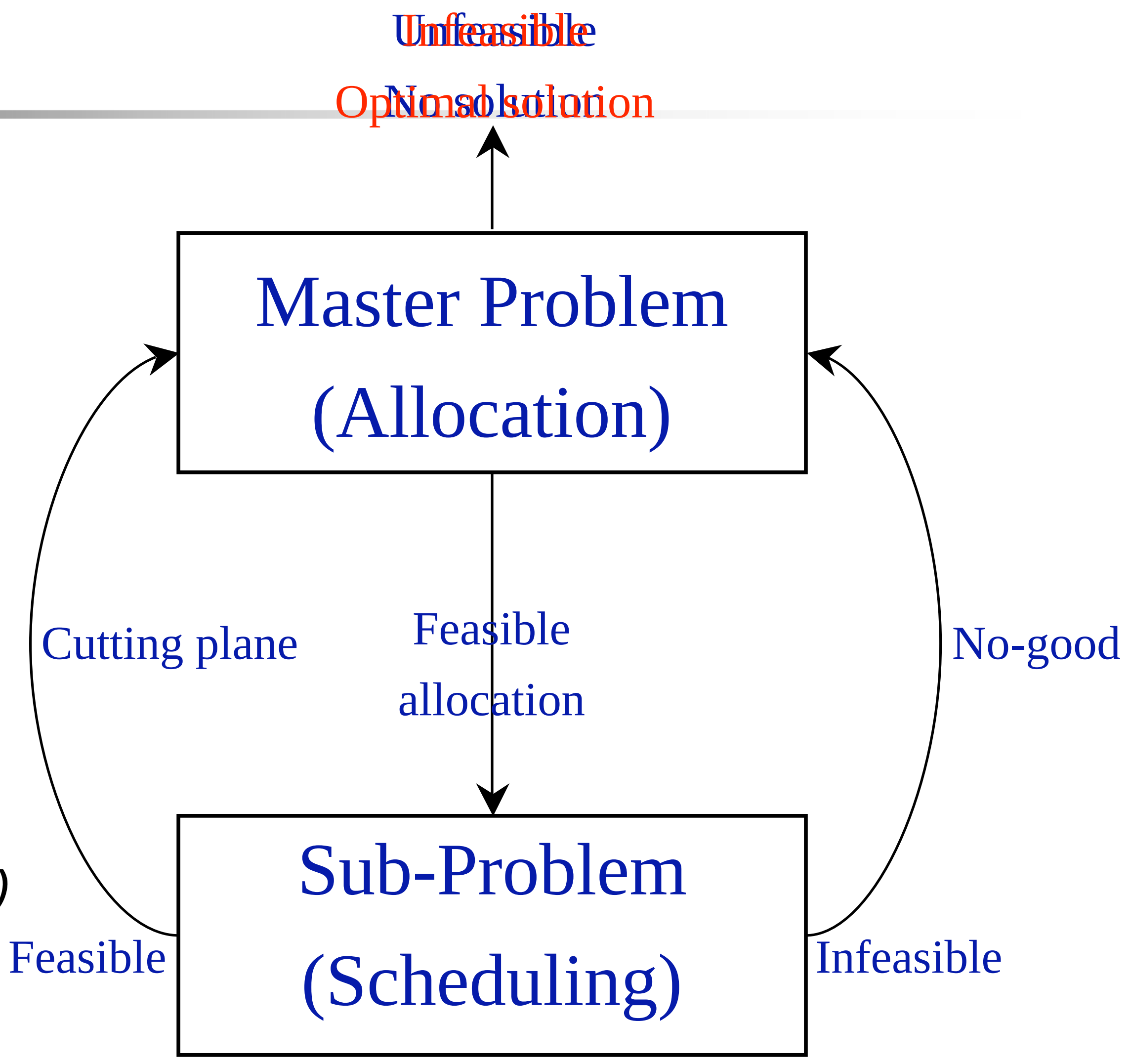
- As a whole it is a scheduling problem with alternative resources (each processor at each frequency is an alternative): **very tough problem**, it has never been solved to optimality by the system design community.
- It smoothly decomposes into allocation and scheduling
  - Allocation and frequency assignment better handled with IP techniques
  - Scheduling better handled with CP techniques
  - The objective function depends both on allocation and scheduling variables
  - We exploit Logic Benders Decomposition to solve the problem
  - *INTERACTION REGULATED VIA NO-GOODS and CUTTING PLANES*

Problem decomposition

Objective Function:
Min(Power consumption)

- Assignment of tasks to processors and of frequencies to tasks *(master problem)*
  - ✓ OF relates alternative resources to couples of tasks
  - ✓ Not a good scenario for CP

- Task scheduling with static resource assignment *(subproblem)*
  - ✓ IP does not handle time efficiently
  - ✓ CP is instead effective



Master Problem model (I)

- Assignment of tasks to processors and frequencies (modes) to task
  - ✓ $\mathbf{X_{ptm}}$ = 1 if task *t* executes on processor *p* at mode *m*, 0 otherwise,
  - ✓ $\mathbf{R_{pt1t2m}}$ =1 if task t_1 running on processor *p* at mode *m* reads data from task t_2 not running on *p*
  - ✓ $\mathbf{W_{pt1t2m}}$ =1 if task t_1 running on processor *p* at mode *m writes* data for task t_2 not running on *p*
- Each task should be allocated to one processor at one mode:

[illegible]

- Communications between tasks happen at most once:

[illegible]

Master Problem model (II)

- Task deadlines are captured:

[illegible]

[illegible]

- The objective function has three contributions:

[illegible]

A task executes

[illegible]

A task reads data from another processor

A task writes data to another processor

[illegible]

Sub-Problem model (I)

➢Variables representing tasks and communications starting times (durations are fixed)

- $Start_i$: starting time of task i , $duration_i$=WCN_i / f_i
- $StartWrite_{ij}$: starting time of task i writing activity , $dWrite_{ij}$=WCN_{Wij} / f_i
- $StartRead_{ij}$: starting time of task j reading activity , $dRead_{ij}$=WCN_{Rij} / f_j

➢ For each couple of tasks (*i*, *j*), s.t. *i* communicates with *j*, we introduce the constraints:

- $StartWrite_{ij} + dWrite_{ij} <= StartRead_{ji}$
- $Start_i + duration_i <= StartWrite_{ij}$
- $StartRead_{ji} + dRead_{ji} <= Start_j$

Sub-Problem model (I)

Precedence constraints:

Starti + durationi <= Startj (same processor)

Starti + durationi + dWriteij + dReadji <= Startj (different processors)

Resource modelling

processors – *cumulative* (StartListp,DurationListp,[1],1)

bus – *cumulative* (StartReadWriteList,DurationList, Fraction, TotBWidth)

Capturing deadlines:

Starti + durationi <= dlti , for each task ti

Starti + durationi <= dlp , for each task *i* running on processor *p*

Sub-Problem model (II)

- Modelling transition times:
- we label each task with its frequency f_i and we consider a transition table defining the time that must elapse between the execution of two tasks with different labels
  - $Start_{f1} + duration_{f1} + TransTime_{f1f2} <= Start_{f2}$

- Modelling transition costs:
- we label each task with its frequency f_i and we consider a transition table defining the cost that must be paid to switch from one frequency to another

- The objective function is: [illegible]

where S_p is the set of tasks running on processor *p*

Improving the Master Problem: Benders Cuts

When the sub-problem is solved, we can add Benders Cuts to the master problem model:

- If the sub-problem is unfeasible, the Benders Cut is a no-good. The same allocation, and all the symmetric, must not be found again.

[illegible]

where J_p is the set of couples (task, mode) allocated to p

- If the sub-problem has a solution whose cost is Setup*, we state that this is the best solution unless a better one can be found with a different allocation.

[illegible]

[illegible]

where $Setup_p$* is the setup cost of processor p,

Improving the Master Problem: relaxation of the subproblem

Let introduce a new variable Z_{pm} in the master problem model, taking value 1 if the mode *m* appears at least once on processor *p*. Calling E_m the minimum energy for switching to mode *m*

[illegible]

Is a valid lower bound for the setup costs of the processors, calculated in the sub-problem.

A similar lower bound can be calculated for the setup time

[illegible]

so in the master problem we can capture also processor deadlines

[illegible]

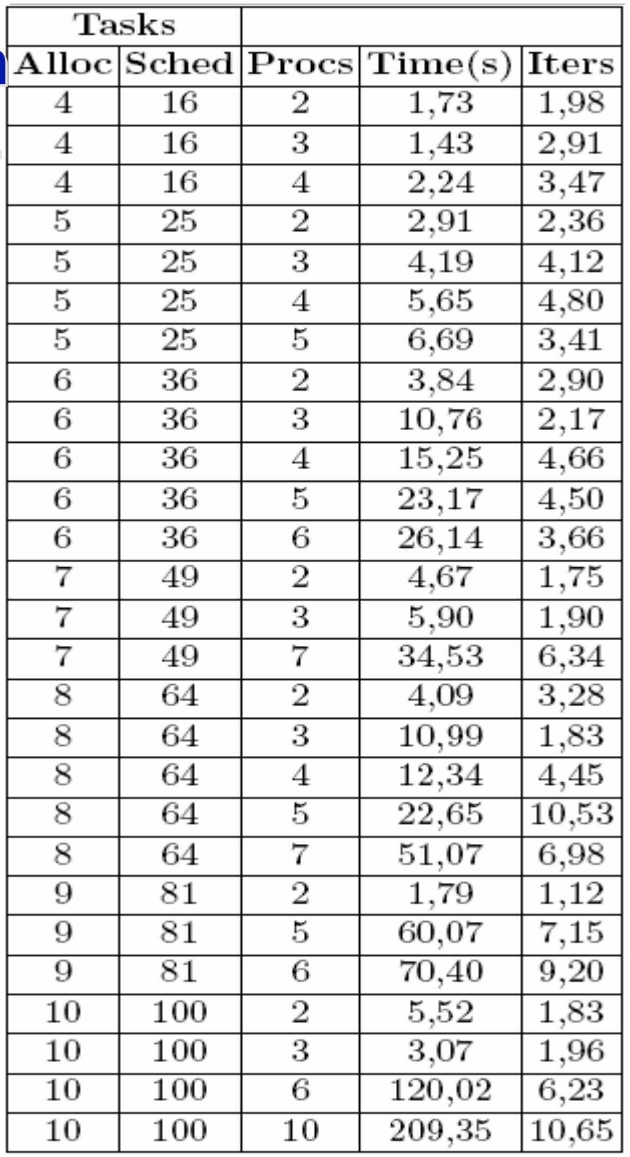
Experimental results

Task graph representing a pipeline: tasks execute iteratively

Generic task graph

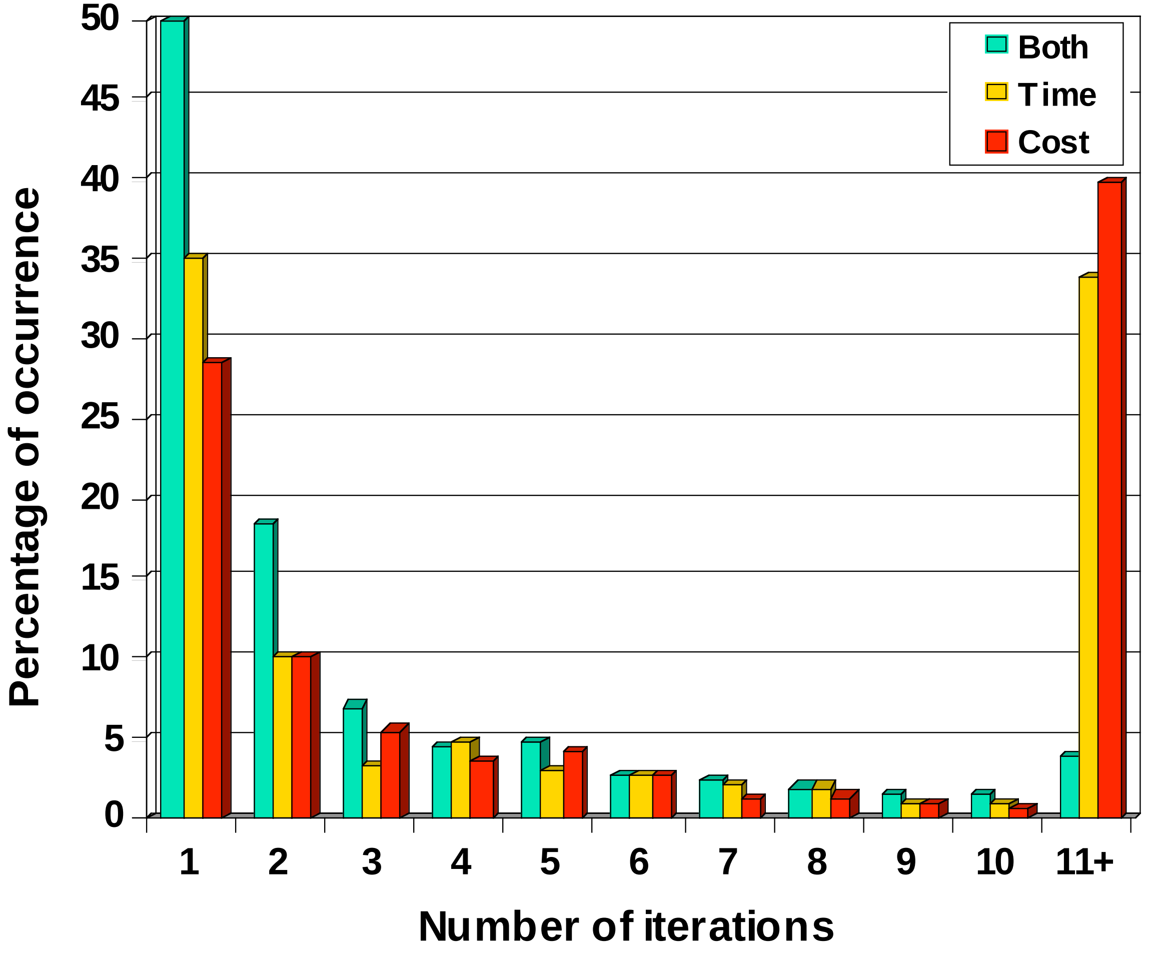
GSM applications

MPEG encoding



Tasks				
Alloc	Sched	Procs	Time(s)	Iters
4	16	2	1,73	1,98
4	16	3	1,43	2,91
4	16	4	2,24	3,47
5	25	2	2,91	2,36
5	25	3	4,19	4,12
5	25	4	5,65	4,80
5	25	5	6,69	3,41
6	36	2	3,84	2,90
6	36	3	10,76	2,17
6	36	4	15,25	4,66
6	36	5	23,17	4,50
6	36	6	26,14	3,66
7	49	2	4,67	1,75
7	49	3	5,90	1,90
7	49	7	34,53	6,34
8	64	2	4,09	3,28
8	64	3	10,99	1,83
8	64	4	12,34	4,45
8	64	5	22,65	10,53
8	64	7	51,07	6,98
9	81	2	1,79	1,12
9	81	5	60,07	7,15
9	81	6	70,40	9,20
10	100	2	5,52	1,83
10	100	3	3,07	1,96
10	100	6	120,02	6,23
10	100	10	209,35	10,65

Experimental results: Number of iterations distribution



Allocation and scheduling of CTG

- On going research
- Up to now only the optimization part has been completed, the validation still missing
- Objective function: communication cost
- Technique: Logic based Benders Decomposition. We transform a stochastic problem in an approximation based on the CTG analisys. The approximation turns out to be exact.
- Pipelined and non-pipelined applications
- Performances comparable with the deterministic case

  Some extremely hard instances: possibly solved with randomization in complete search

Objective function

$$\sum_{i=0}^{n-1} \boxed{f_i(X(\omega))}\left[\overline{m}_i\Big(1-\sum_{j=0}^{p-1}\boxed{M_{ij}}\Big)+\overline{s}_i\Big(1-\sum_{j=0}^{p-1}\boxed{S_{ij}}\Big)+\sum_{r\in\mathcal{E}_{i\to k}}\boxed{f_k(X(\omega))}\,\overline{e}_r\Big(1-\sum_{j=0}^{p-1}\boxed{E_{rj}}\Big)\right]$$

- Depends on decision variables and on stochastic variables
- When the allocation is fixed only on stochastic vars.

$$\sum_{\omega\in\Omega} p(\omega)\left[\boxed{C_1}f_i(X(\omega)) + f_i(X(\omega))\sum_{r\in\mathcal{E}_{i\to k}} f_k(X(\omega))\boxed{C_{2_k}}\right]$$

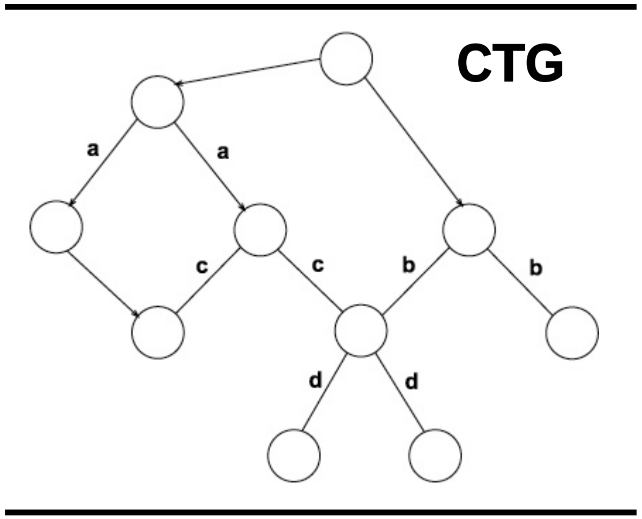
Substituting: $\sum_{\omega\in\Omega} p(\omega) f_i(X(\omega)) C_1 \;\blacktriangleright\; C_1 \sum_{\omega\in\Omega_i^1} p(\omega)$

$$\sum_{\omega\in\Omega} p(\omega) f_i(X(\omega)) \sum_{r\in\mathcal{E}_{i\to k}} f_k(X(\omega)) C_{2_k} \;\blacktriangleright\; \sum_{r\in\mathcal{E}_{i\to k}} C_{2_k}\left(\sum_{\omega\in\Omega_i^1\cap\Omega_k^1} p(\omega)\right)$$

The expected value reduces to a deterministic function

CTG analysis

- **We need to know the probability of existence and co-exostence of nodes**
  - We have developed polynomial algorithms



Data structures:
Activation set (AS)
Sequence matrix

Coexistence set

Probability of AS existence

Coexistence probability

Complexity O(c^3)

Allocation and scheduling of Multiple Task Graphs

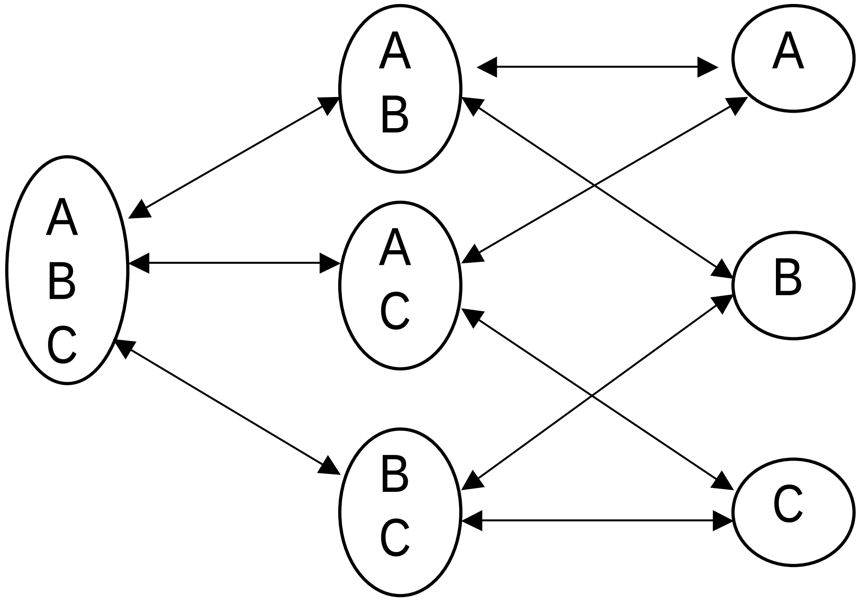
- On going research
- Up to now we are developing the optimization part, the validation still missing
- Objective function: communication cost + migration cost
- Technique: Logic based Benders Decomposition.
- We start from a situation where a TG1 is running and the second TG2 starts. We minimize the communication cost overall plus the migration cost of TG2.
- Many pareto optimal solutions, choose at runtime
- Pipelined applications
- Problem: transition graph with multiple nodes for each configuration

Allocation and scheduling of Multiple Task Graphs

- First solution
  - TG1 is running and TG2 starts its execution. We optimally allocate the second task by possibly migrating some tasks in TG1.
  - Various combination of communication cost and migration cost. Try to find pareto optimal points
  - Choose at run-time
- Same technique when a task graph stops its execution.

Allocation and scheduling of Multiple Task Graphs

- Second solution
  - Compute different minimum communication cost transition graphs with a bounded migration cost
  - Example: task graphs A, B and C



Each arc is labelled with the minimum delta communication cost. Each node is an allocation

Conclusions

- Analize problem structure
- Important to choose the correct solver and representation
- CP and IP have different strenghts: exploit both!!