Using Speculative Computation and Parallelizing techniques to improve Scheduling of Control based Designs

Roberto Cordone

Fabrizio Ferrandi, Gianluca Palermo, Marco D. Santambrogio, Donatella Sciuto

Università Statale di Milano - DTI

Politecnico di Milano - DEI

ASP-DAC 2006 Yokohama, Japan 24-27 January 2006

Outline

- Previous approaches on scheduling with speculation
- Analysis of IRs for HLS:
  - PDG/SDG
  - CDFG/HTG
- Scheduling:
  - Speculation
  - Transformation
- Scheduling ILP formulation:
  - new conditional resource sharing constraint
  - Speculative computation
- Experimental results

Previous approaches

- Code motion techniques
  - SW compilers: Fisher81, Nicolau89
  - HLS: Santos99, Rim95
- Speculation: Jha99, Gupta03, Brewer96
- No exact methods with the exception of Brewer96
  - CDFG, Unit time, no pipeline unit
- Previous ILP-based scheduling (Gebotys93): does not well express control constraints
- Intermediate Representation: CDFG, HTG

What PDGs and SDGs are

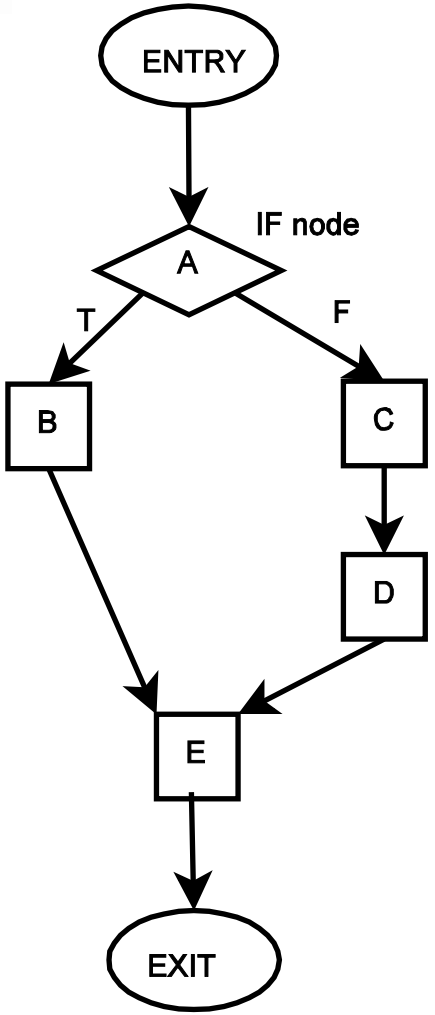
- PDGs are the starting point: they represent a single procedure
- A PDGs is a directed graph
  - Its nodes represent:
    - Statements
    - Predicates (loop/control conditions)
  - Its edges represent:
    - Data dependencies
    - Control dependencies
- A System Dependency Graph SDG is a collection of PDGs connected by call and parameter edges
- System Dependency Graphs:
  - Abstract code representation
  - Explicit representation of all dependencies between statements
  - Easy detection of parallelizable code

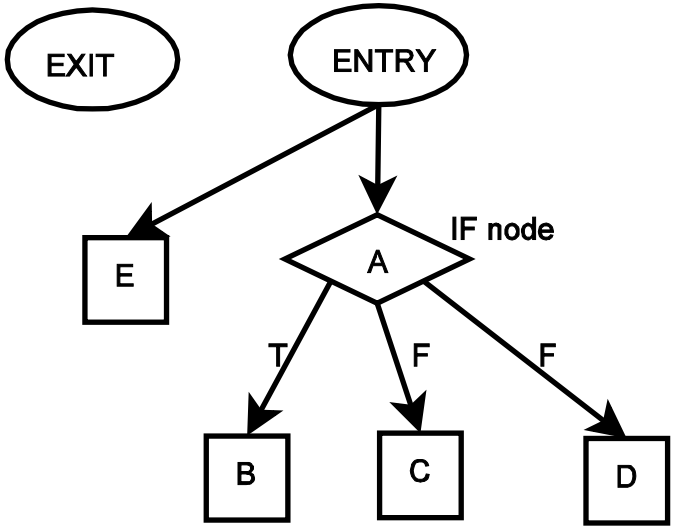
Control dependencies

- Intuition:
  - **Node A is control dependent on node B if B may change whether A is executed or not**

- Formal definition **(Ferrante et al.)**:
  - Y **is control dependent on** X **iff:**
    - There exists a path P from X to Y in the CFG with any node Z in P post-dominated by Y
    - X is **not** post-dominated by Y

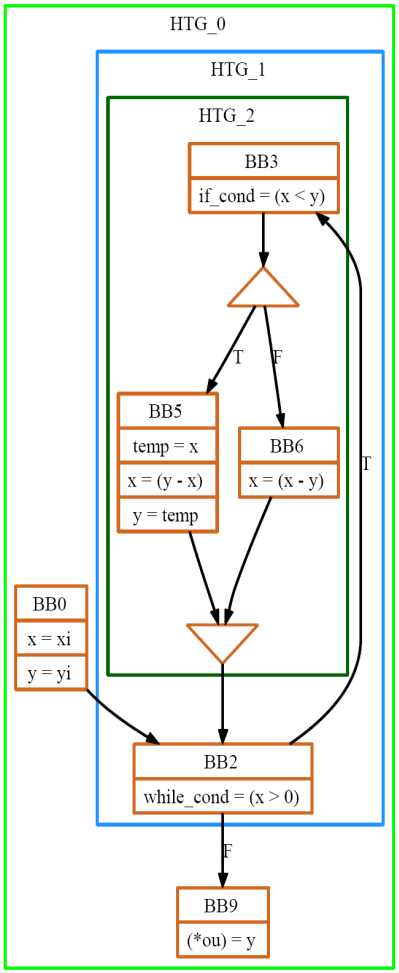
CFG vs CDG (Ferrante + Girkar & Polychronopoulos)





HTG (Gupta et al.)

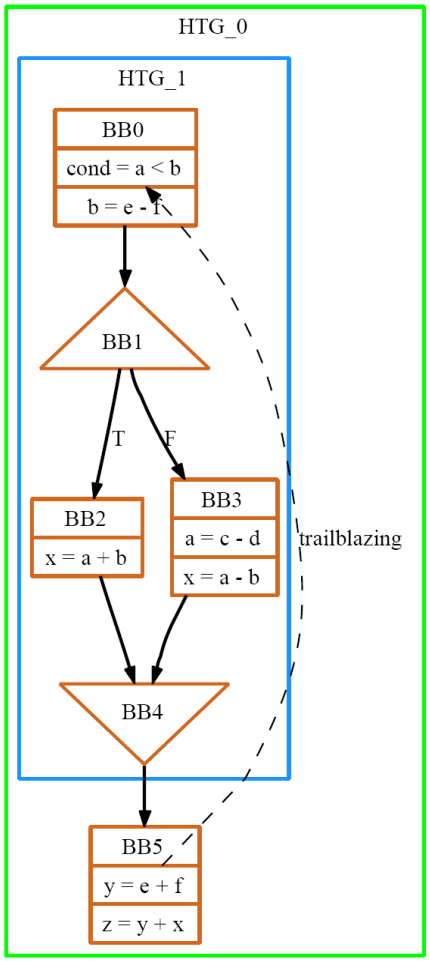
```
void
gcd (int xi, int yi, int *ou)
{
    int x, y, temp;
    x = xi;
    y = yi;
    while (x >0)
    {   if (x <= y)
        {   temp = x;
            x = y-x;
            y = temp;
        } else
            x = x - y;
    }
    *ou = y;
}
```

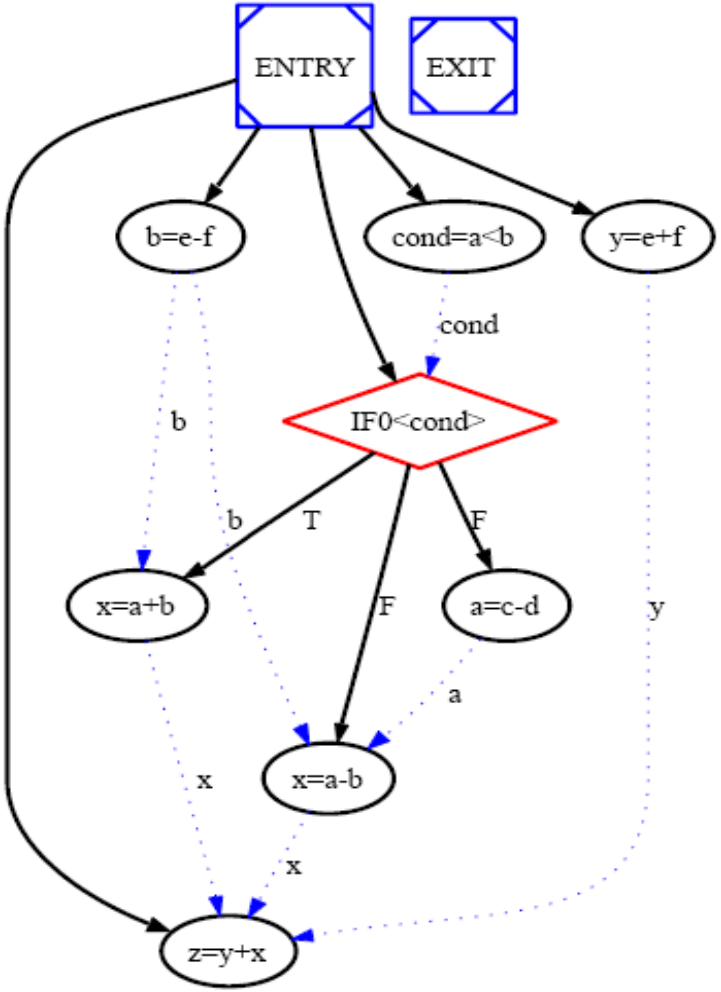


Transformations vs scheduling
Gupta et al.

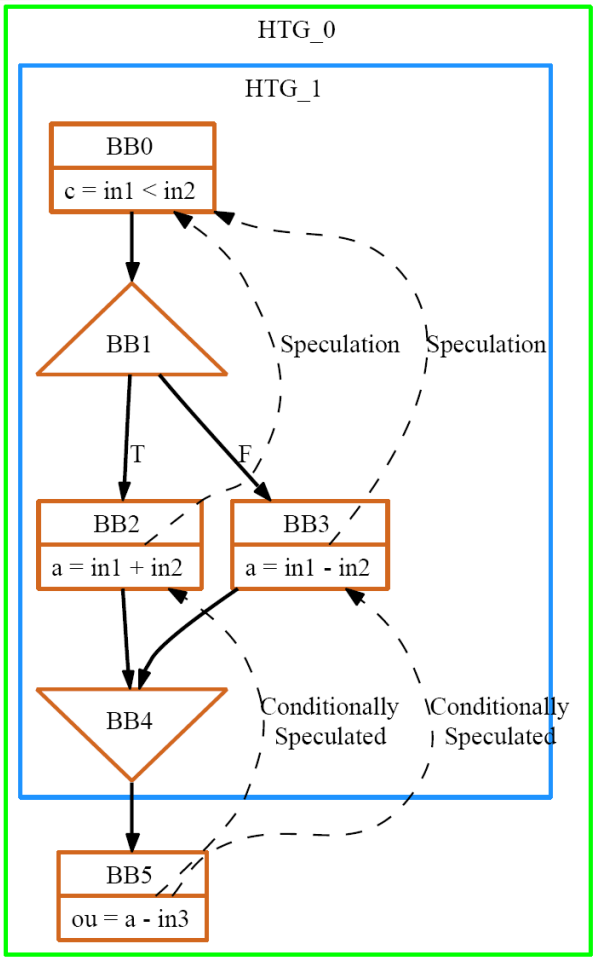
- Across Hierarchical Blocks
  - movement of operations across entire hierarchical blocks
- Speculation
  - unconditional execution of operations that were originally supposed to have executed conditionally
- Reverse Speculation
  - where operations before conditionals are moved into subsequent conditional blocks and executed conditionally
- Conditional Speculation
  - in which an operation is moved and duplicated up into preceding conditional branches and executed conditionally

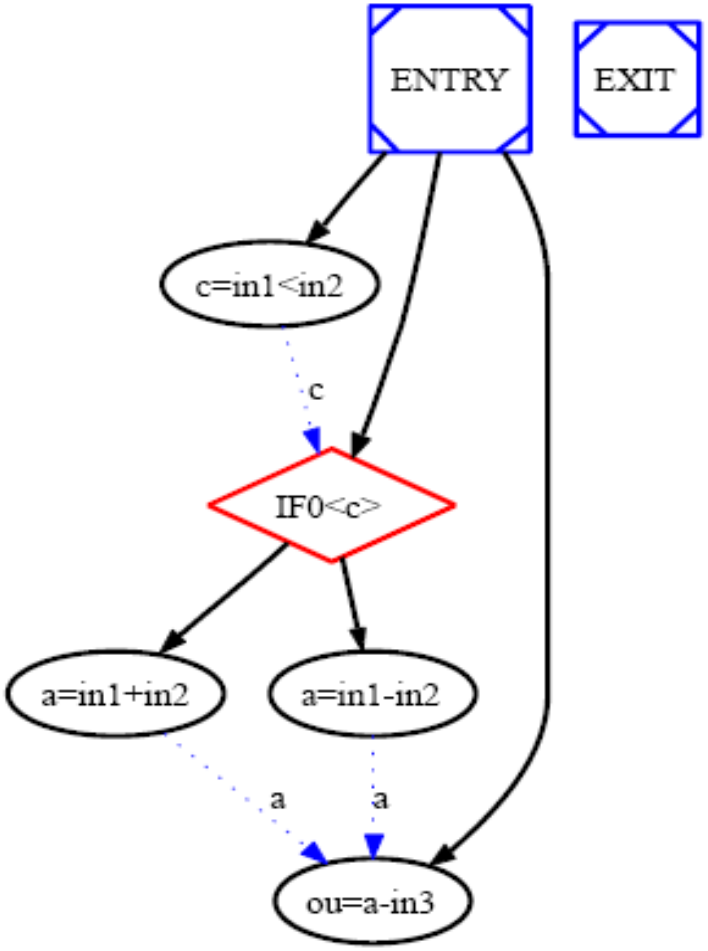
Across Hierarchical Blocks



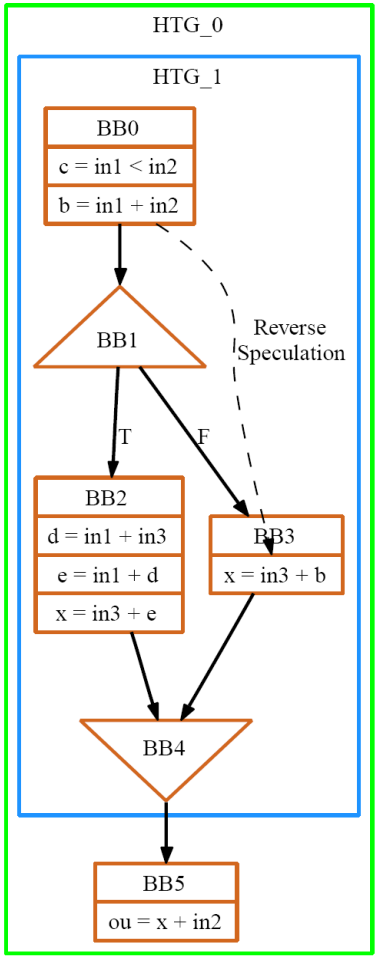


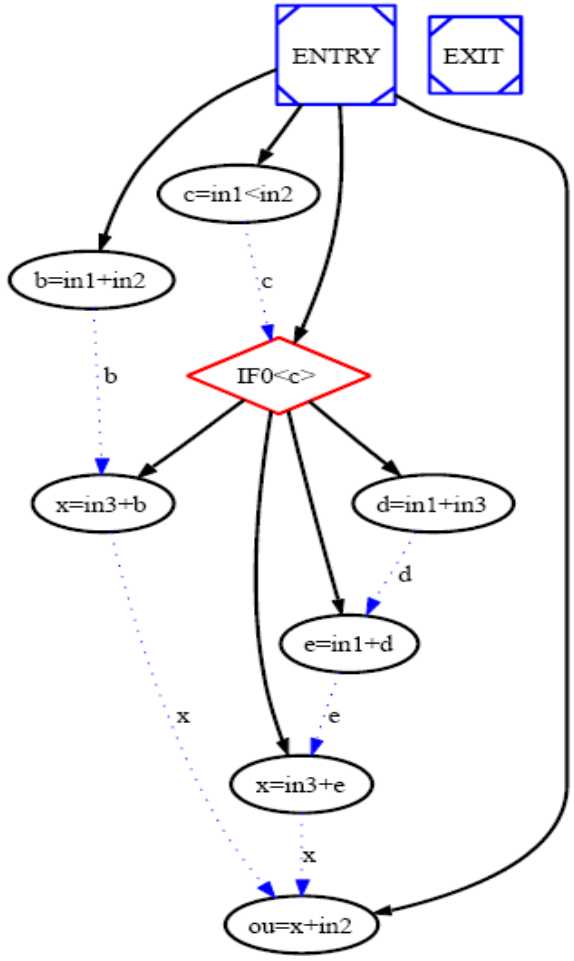
Speculation & Conditional Speculation





Reverse Speculation





Scheduling: ILP formulation (Gebotys)

Considered scheduling problem

min(w)

subject to

$w \geq \sum_i (j + C_{k,i} - 1) x_{i,j,k} \quad j \epsilon J,\ k \epsilon K$

- where
  - w is a variable representing the last control step
  - $x_{i,j,k}$ =1 when operation k starts executing at control step j and it is assigned to a functional unit of type i and
  - $C_{k,i}$ is the execution time of operation k mapped on functional unit i
  - $L_{i,k}$ is the initiation time of operation k mapped on functional unit i

Scheduling constraints (Gebotys)

- Assignment constraint
  - Each operation is assigned to a specific control step

$$\sum_i \sum_j x_{i,j,k} = 1 \qquad \forall\, k \in K$$

- Precedence constraint

$$\sum_i \left(\sum_{\substack{j' \le j_c, \\ asap(k') \le j' \le alap(k')}} x_{i,j',k'} + \sum_{\substack{j \ge j_c - C_{k,i} + 1, \\ asap(k) \le j \le alap(k)}} x_{i,j,k} \right) \le 1$$

Node packing problem

$\forall\, k \in K$, $k' \in K$, $j_c \in J$ subject to

$k \angle k'$, $k \notin N$, $k' \notin W$ and

$\max(asap(k'), asap(k)+\min_i(C_k, i)-1) \le jc \le \min(alap(k'), alap(k)+\max_i(C_k, i)-1)$

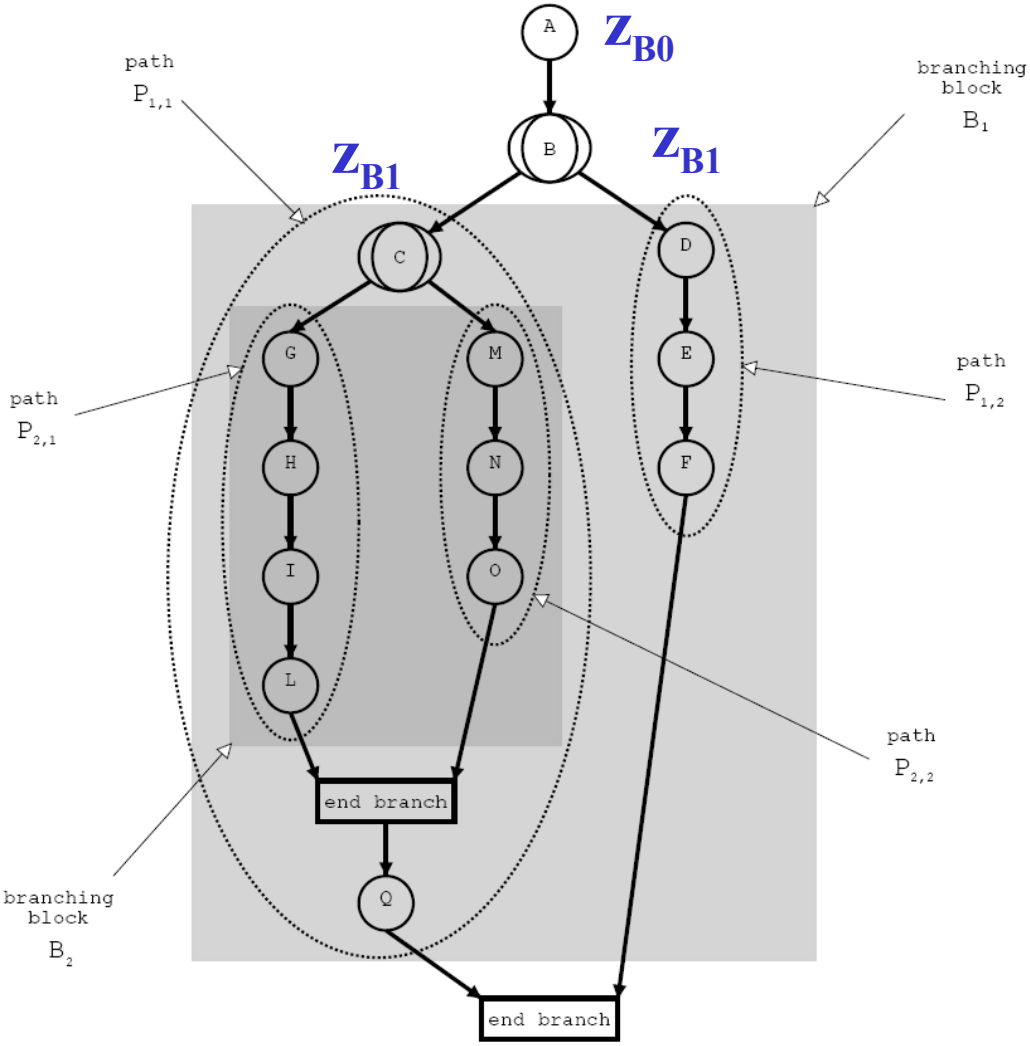
Resource constraints

- Conditional branch
- Gebotys et al.
  - for each path defined a capacity constraints
- Our approach
  - recursive capacity constraints

$$z_{i,j,B_0} \leq N_i \qquad \forall\, i \in I, j \in J$$

$$\sum_{k \in O_P} \sum_{j' = j - L_{k,i} + 1}^{j} x_{i,j',k} + \sum_{B' \in B_P} z_{i,j,B'} \leq z_{i,j,B}$$

$$\forall\, i \in I, j \in J, B \in \beta, P \in B$$



Resource constraints with speculation

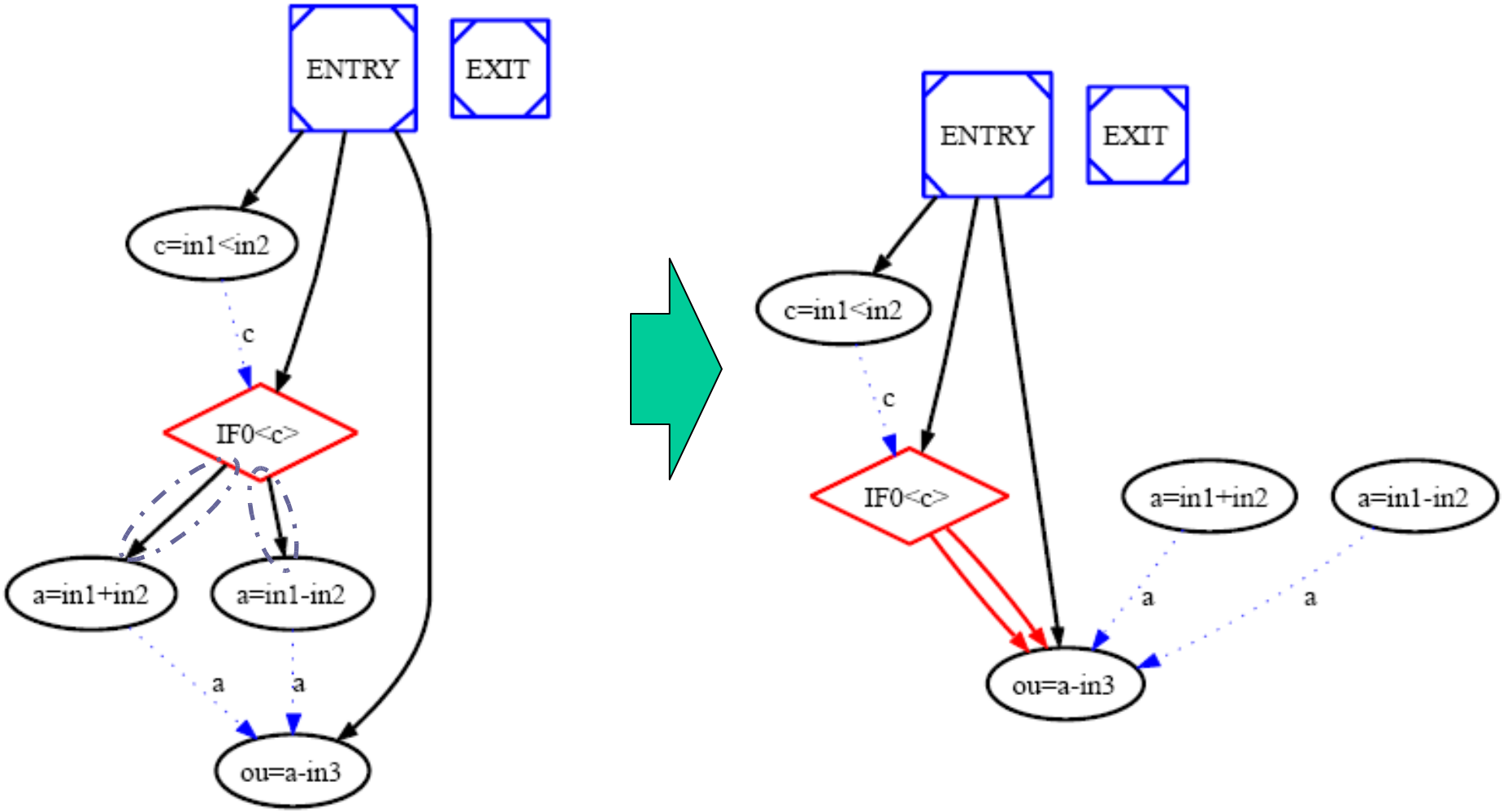
- To take into account speculation we modify the resource constraint:

$$\sum_{P \in B} \left(\sum_{k \in P} \sum_{j'=j}^{j-L_{ki}+1} x_{ij'k} + \sum_{B' \in P} z_{ijB'} \right) \leq z_{ijB} + M \left(1 - \sum_{i' \in I_B} \sum_{j' \geq j - C_{k_B i'} + 1} x_{ij'k_B} \right)$$

$$i \in I, j \in J, B \in \mathcal{B} \setminus \{B_0\}$$

- M is a constant large enough to make the constraint redundant when no speculation is performed

SDG & Resource constraint with speculation



Branch and Cut

- Based on the open source package COIN-OR
  - (http://www.coin-or.org)
- provides a set of tools among which an ILP solver with the capability of generating the most important families of valid inequalities.

- The inequalities effective to solve the scheduling problem are:
  - Gomory
  - Clique
  - Probing
  - Knapsack

Experimental Results

- compare three different scheduling techniques
  - SPARK: Gupta et al. Framework
  - LIST: standard list based adapted to SDG
  - ILP: our ILP formulation of the scheduling problem with speculation
- Benchmarks:
  - A set of standard HLS benchmarks
  - Two media benchmarks:
    - MotionVector
    - Adpcm(Decode/Encode)

Experimental Results: speculation for ARCH1

1-Add, 1-Sub, 1-Mul, 1-Cmp, 1-Sh, 2[]	SPARK		LIST		ILP	
	CS	Time(s)	CS	Time(s)	CS	Time(s)
Kim	10	0.030	10	0.013	10	0.169
Sehwa	8	0.046	9	0.015	8	0.332
Maha	9	0.049	9	0.013	9	0.125
MotionVector	15	0.309	12	0.173	11	28.2
AdpcmDecode	18	0.110	13	0.212	13	3.357
AdpcmEncode	18	0.144	14	0.210	14	2.011

Experimental Results: speculation for ARCH2

1-Add, 1-Sub, 1-Mul, 2-Cmp, 1-Sh, 2[]	SPARK		LIST		ILP	
	CS	Time(s)	CS	Time(s)	CS	Time(s)
Kim	10	0.054	10	0.012	9	0.163
Sehwa	7	0.045	7	0.014	7	0.132
Maha	9	0.054	9	0.012	9	0.129
MotionVector	13	0.274	12	0.177	11	26.2
AdpcmDecode	15	0.122	11	0.190	11	0.807
AdpcmEncode	18	0.145	13	0.519	13	1.237

Experimental Results: speculation for ARCH3

2-Add, 2-Sub, 1-Mul, 2-Cmp, 1-Sh, 2[]	SPARK		LIST		ILP	
	CS	Time(s)	CS	Time(s)	CS	Time(s)
Kim	8	0.036	8	0.010	8	0.074
Sehwa	6	0.049	6	0.013	6	0.085
Maha	9	0.054	9	0.011	9	0.123
MotionVector	10	0.323	10	0.162	9	0.968
AdpcmDecode	15	0.114	10	0.164	10	0.474
AdpcmEncode	18	0.148	13	0.367	13	0.843

Experimental Results: B&B vs B&C

- Set of benchmarks enriched with some well known data-intensive high level synthesis benchmarks
- Two architecture considered: ARCH1 and ARCH3
- ILP branch & Bound:
  - Solution proved optimally in less than 1000sec: 15 vs 11
- ILP branch & Cut:
  - Solution proved optimally in less than 1000sec: 24 vs 2

- Further COIN-OR customization allow to optimally solve all the problems

Future works

- Improve the GCC interface
- Better support of reverse and conditionally speculation techniques
- Analysis of heuristics and inequalities to better support B&C: approximations, lower bound estimations
- Coin-Or branching customization
- Analysis and integration of register binding, module and interconnect allocation
- Exploitation of SDG to perform partitioning for
  - HW/SW Codesign
  - Dynamic reconfiguration

Any Questions?

ferrandi@elet.polimi.it