Automated RTR Temporal Partitioning for Reconfigurable Embedded Real-Time System Design

C. Tanougast, Y. Berviller, P. Brunet and S. Weber

L. I. E. N.

Laboratoire d' Instrumentation Electronique de Nancy
Faculté des Sciences de Nancy I, BP239
54506 Vandoeuvre-lès-Nancy, cedex, France

e-mail:
philippe.brunet@lien.uhp-nancy.fr

Outline

I. Introduction

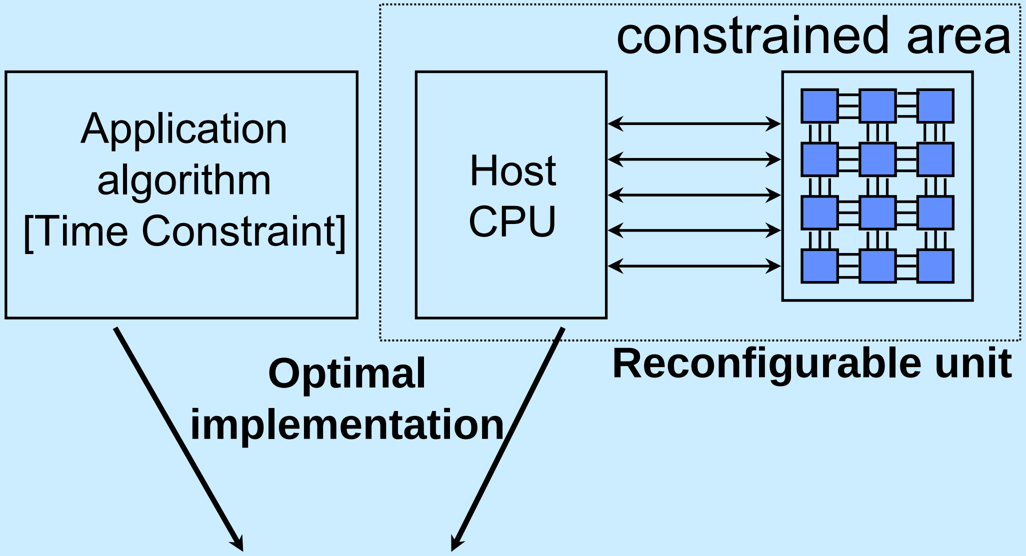
II. Automatic RTR temporal partitioning

III. Application example

IV. Conclusion & Future Works

I. Introduction

- Application development approach



constrained area

Application algorithm [Time Constraint]

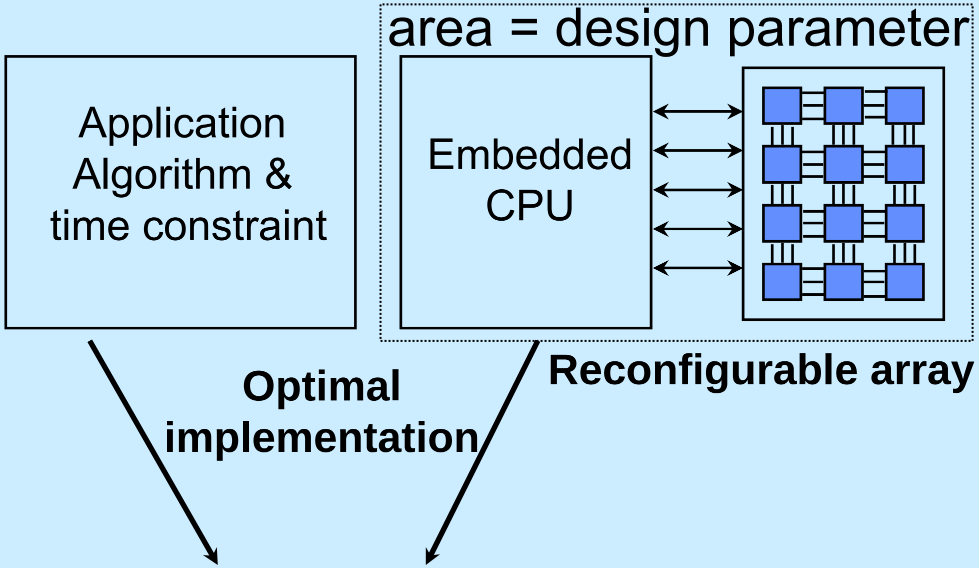
Host CPU

Reconfigurable unit

Optimal implementation

Minimize total processing time, memory bandwidth or number of reconfigurations

- Application specific system design approach



area = design parameter

Application Algorithm & time constraint

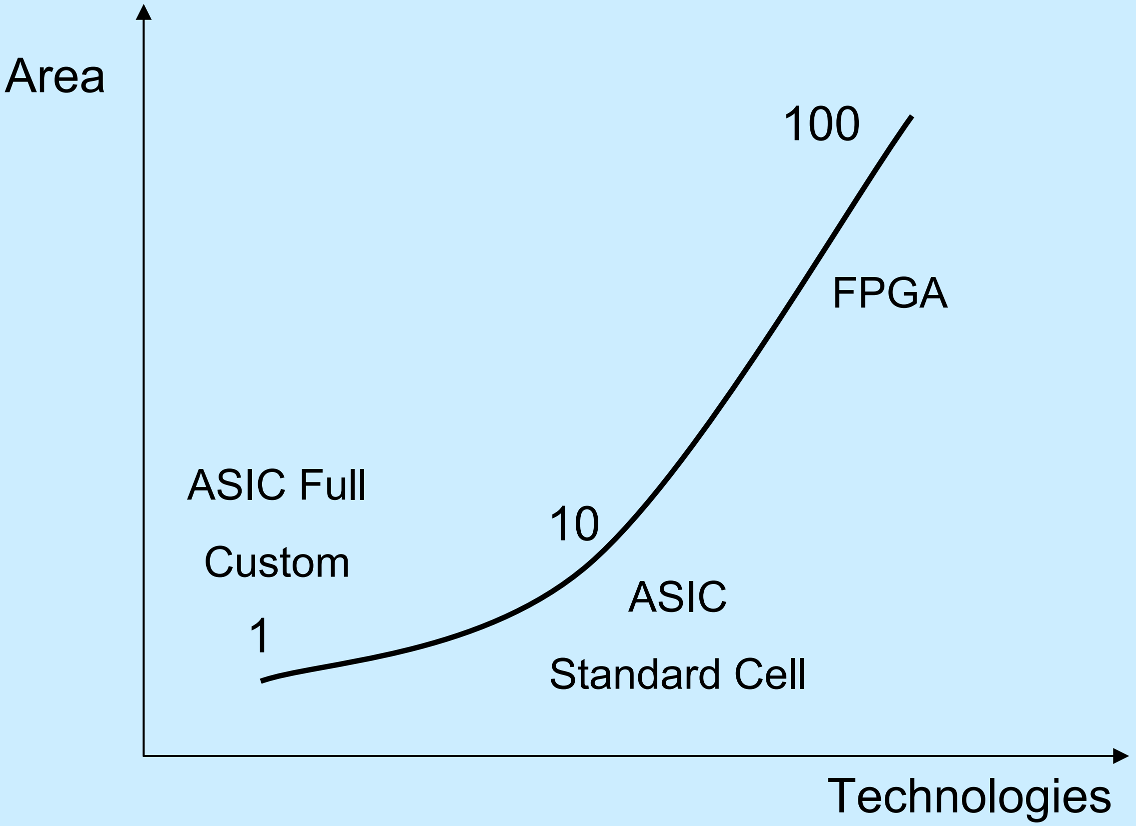
Embedded CPU

Reconfigurable array

Optimal implementation

Minimize area of the reconfigurable array which implements the data-path of the application

Purposes



Silicon area for a same operator type in different technologies

Purposes

→ Optimization of the FPGA logic resources

→ Prevent memory bandwidth overhead

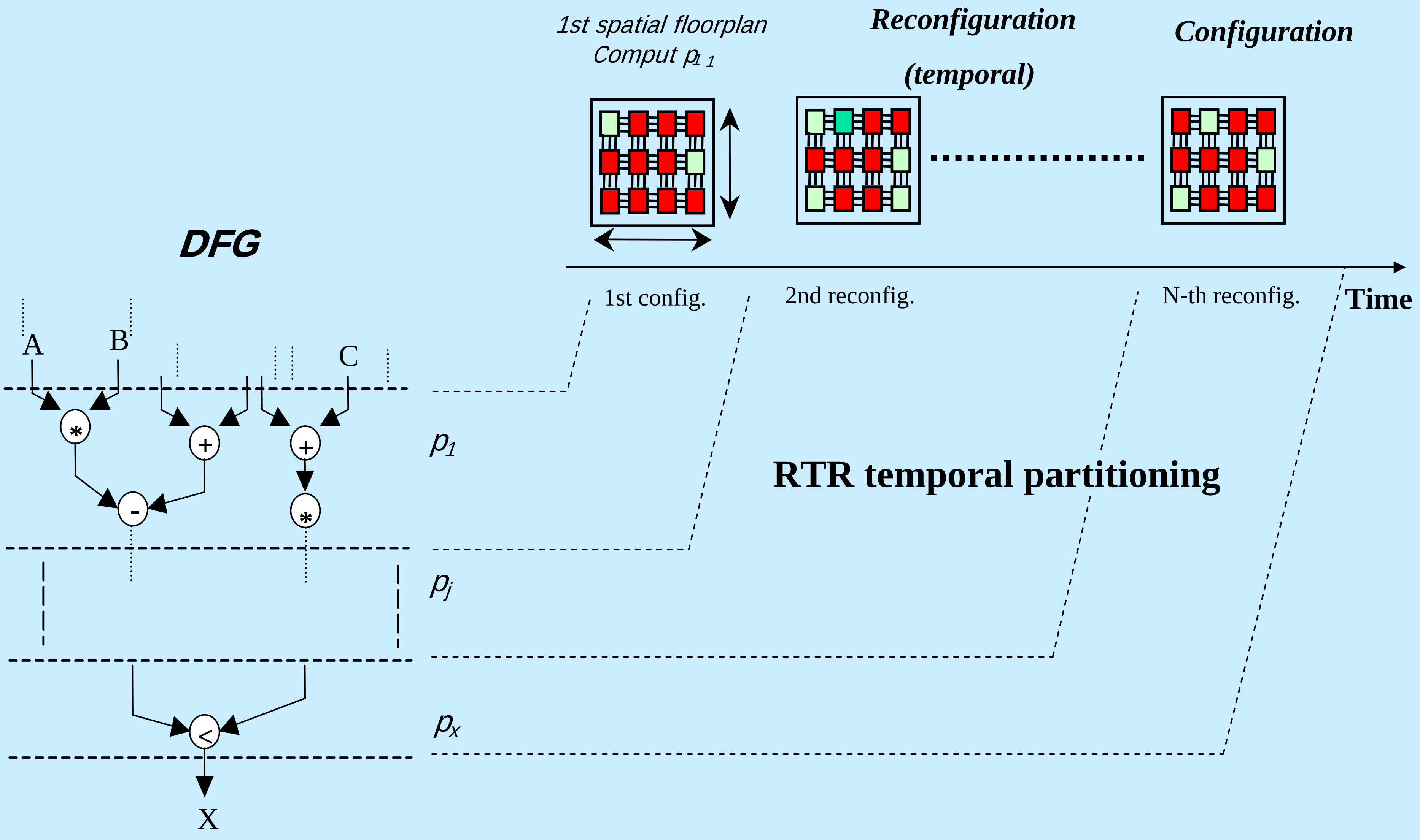
→ Respect of a Time Constraint

→ Speed up conception choices

↓

Computer aided method to maximize efficiency of reconfigurable hardware

How to partition in RTR ?



I. INTRODUCTION II. Automatic RTR temporal partitioning III. Application example IV. Conclusion

How to partition in RTR ?

→ How many temporal partitions are possible ?

→ Where are the partition boundaries ?

Outline

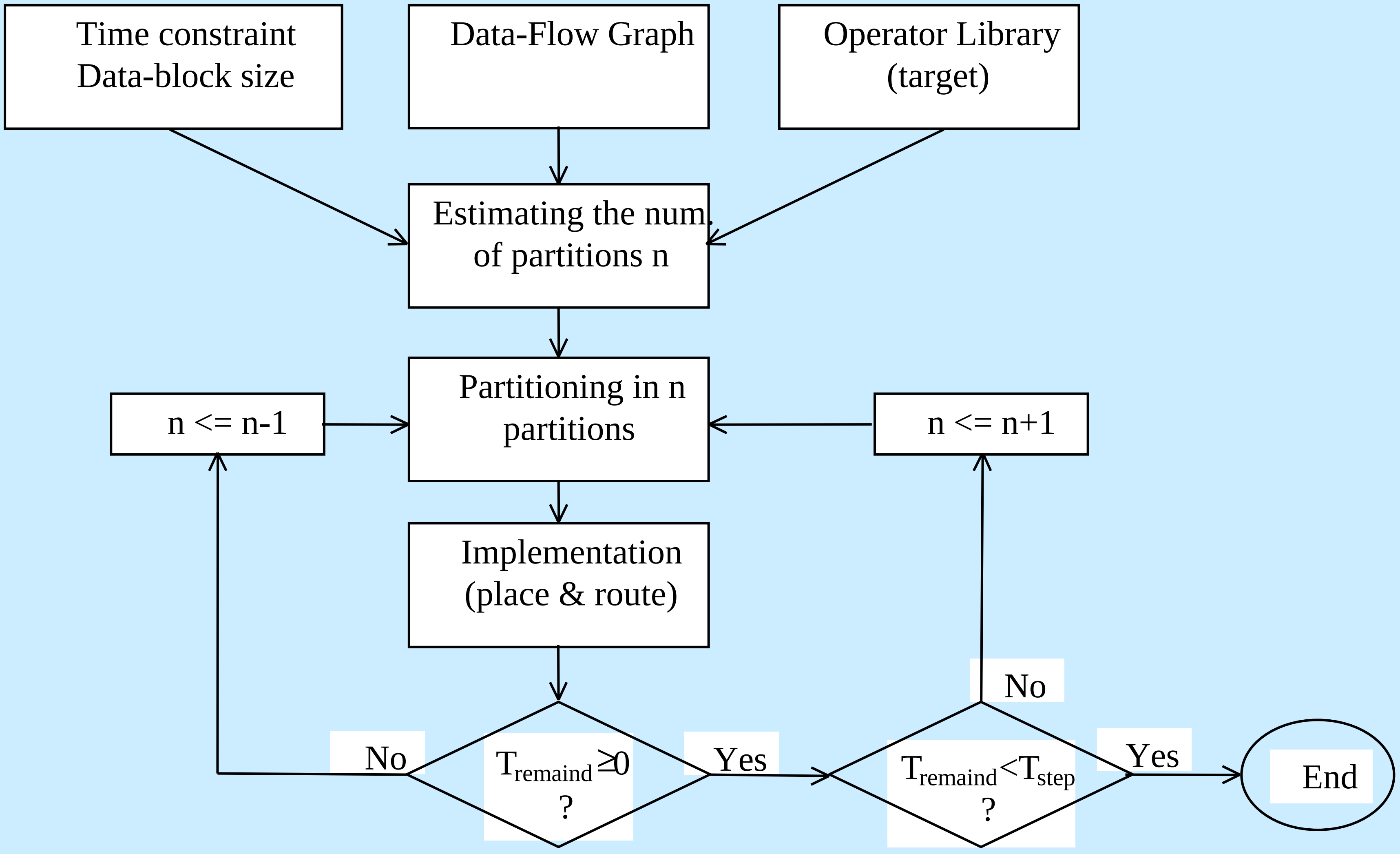
I. Introduction

II. Automatic RTR temporal partitioning

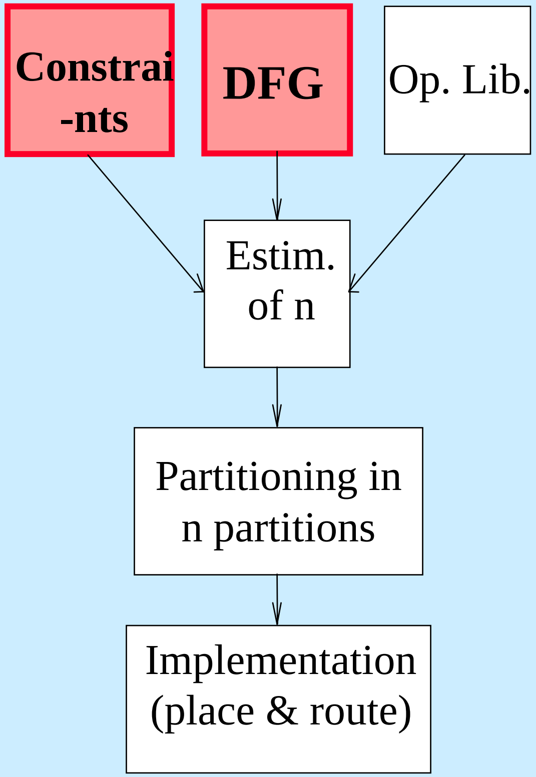
III. Application example

IV. Conclusion & Future Works

Automation using Operator Library



Problem formulation



Algorithm modeled as a **DFG : G(V, E)**

↗ **V** : Arithmetic and *logic operators*

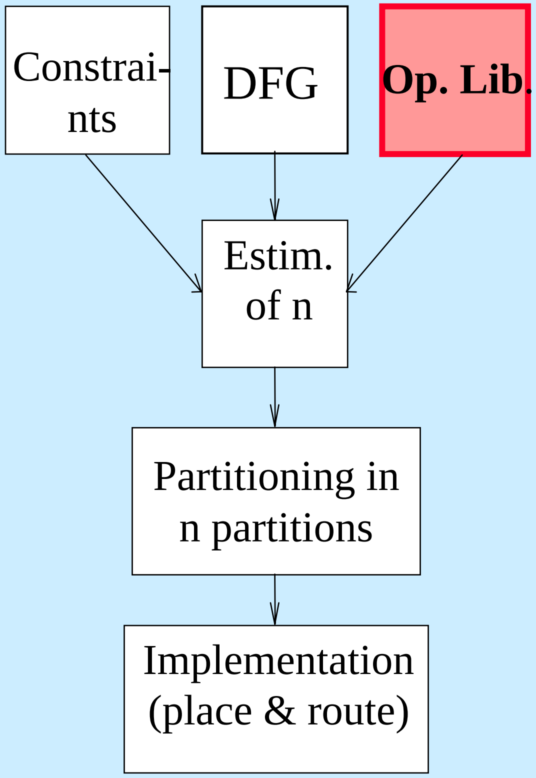
↘ **E** : *Data dependencies*

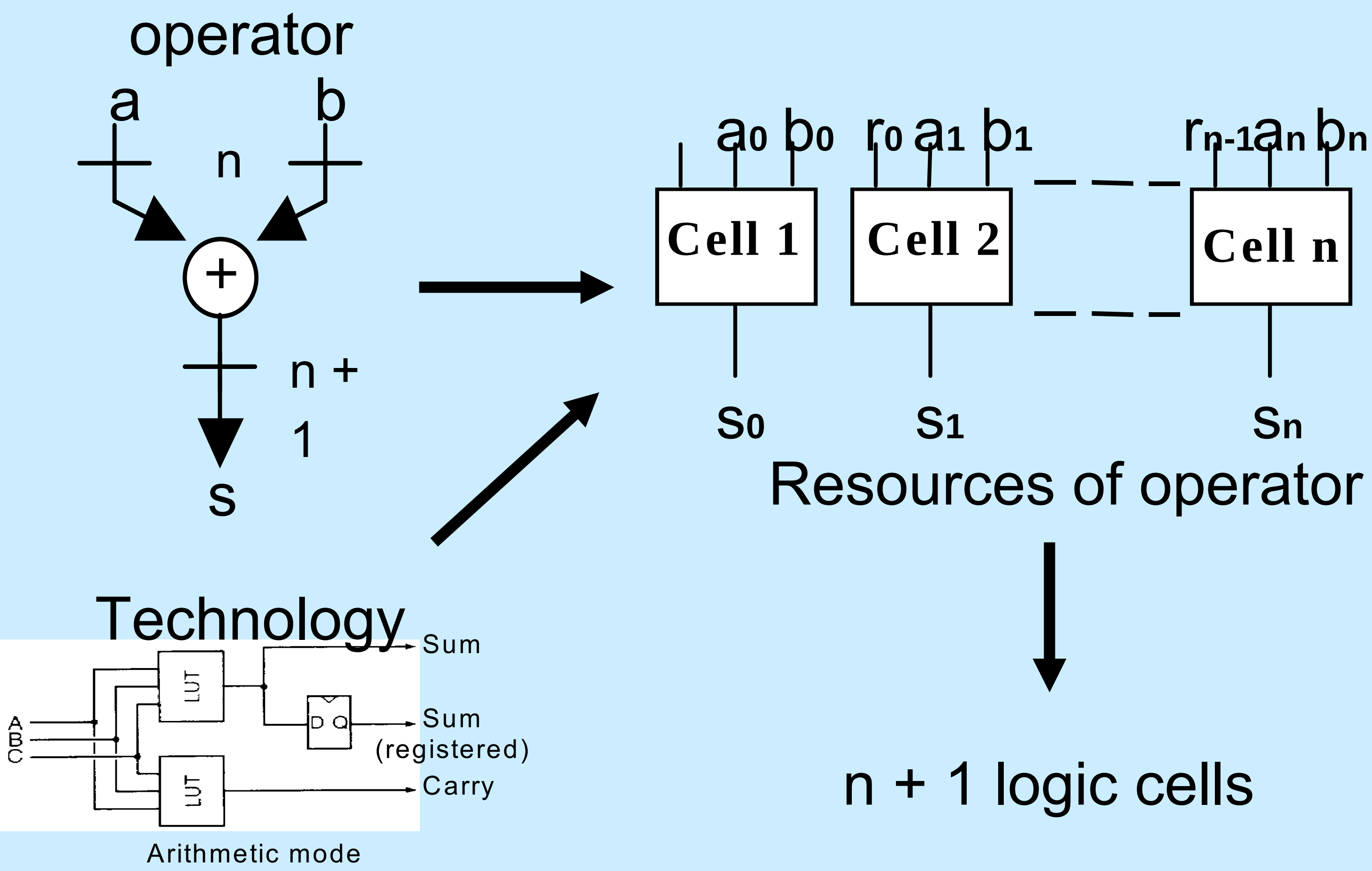
Time constraint → **T**

Reconfiguration speed → V_C

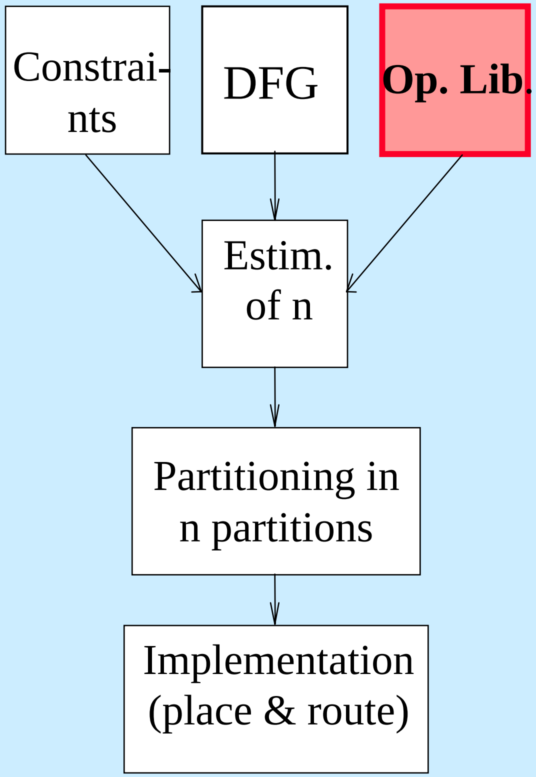
Characterization of operators

1. Resources of operators

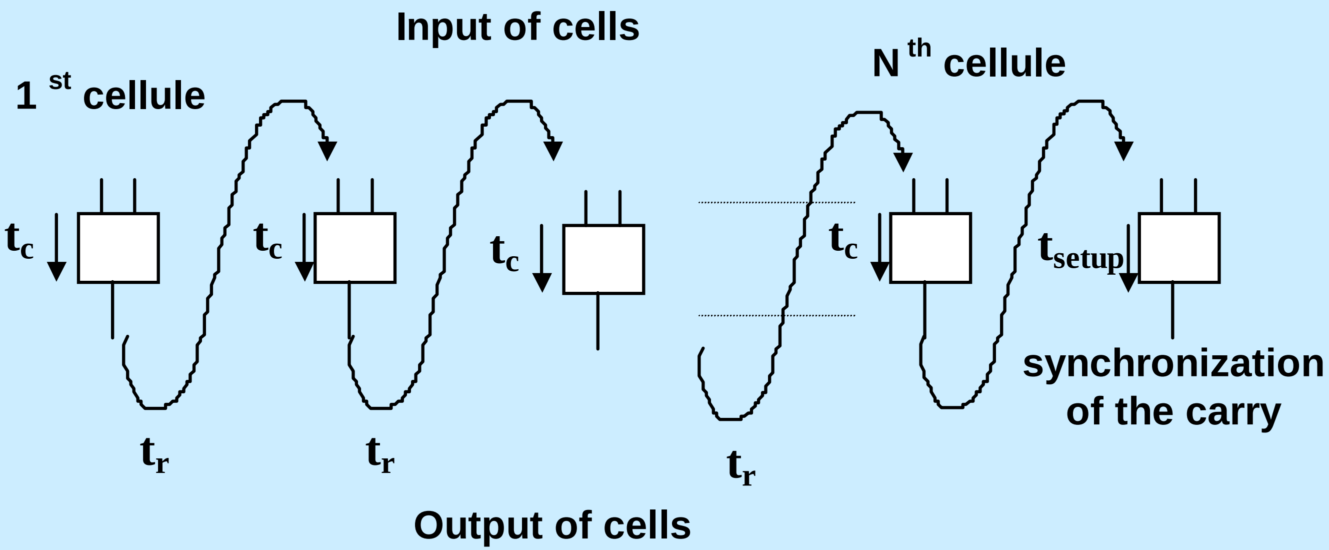




Characterization of operators



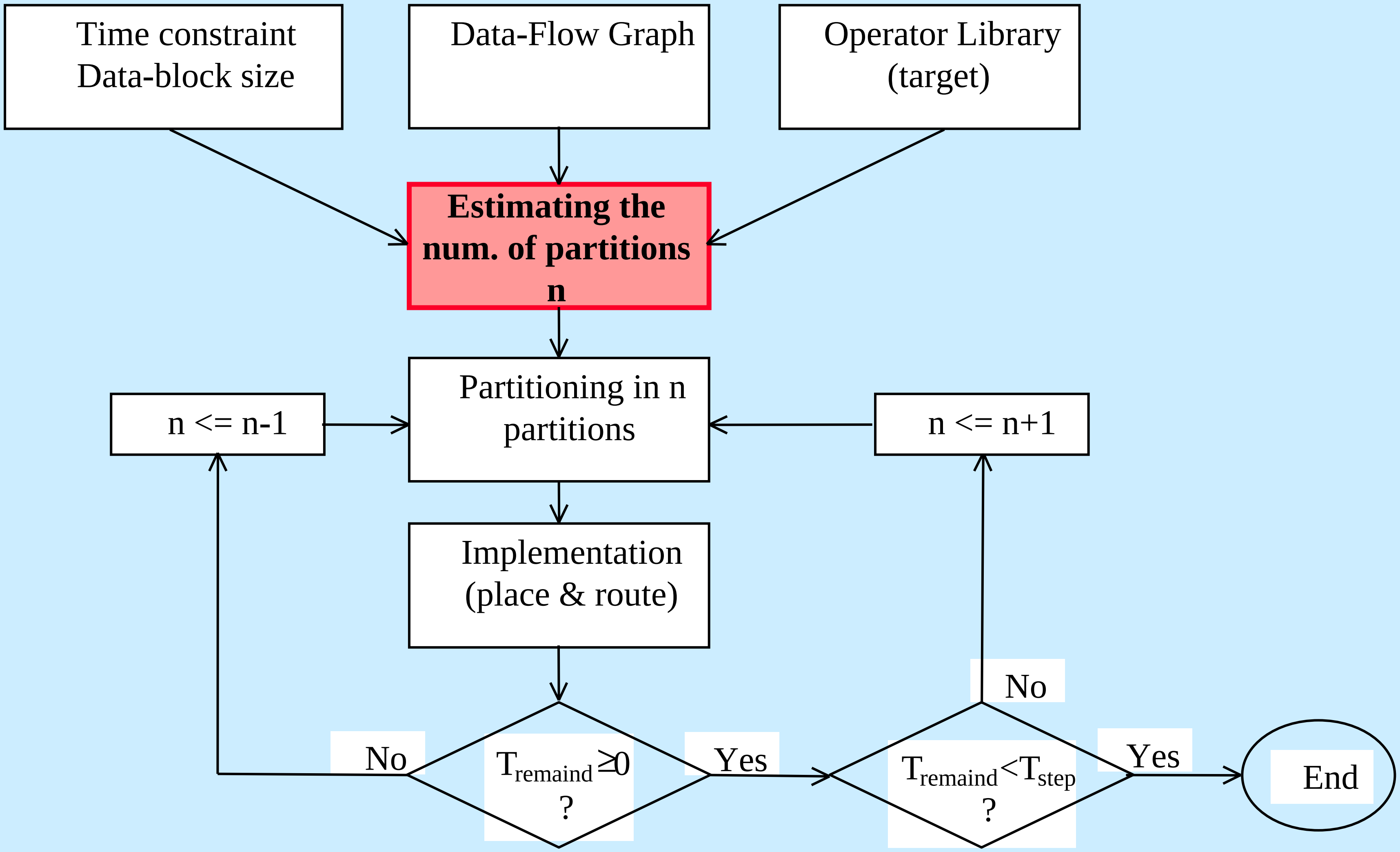
2. Execution time of operators



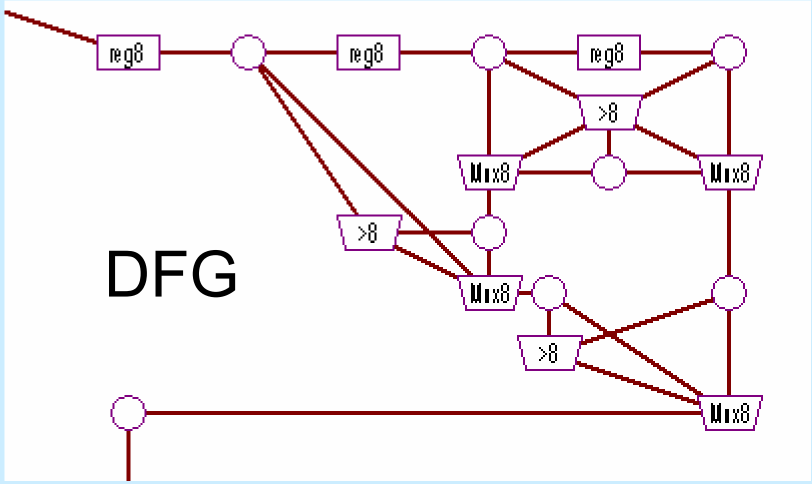
Execution time is function of:

- Type of operation
- Size of data to process
- Characteristical parameters of target technology

Automation using Operator Library



Estimation of DFG performances



Characterization of operators (target)

Total resources evaluation → Sum of each operator resources

Execution time → Slowest operator execution time

Memory needs evaluation → Sum of edges properties along the graph

Evaluation of partitions number

```
V, N, T <= constants                 //Capture constant parameters of
                                     //target and constraints
G <= DFG                             // DFG capture of the application
C <= 0                               // Total area variable
TO <= 0                              //Maximal operator execution time

for each node NDi in G
TO <= max (TO, $NDi.t_i$)            //return current max execution time
C <= C + NDi.Area                    //add area of current node
end for
n <= T / [(N·TO) + rt( )]            // compute n and $C_n$
$C_n$ <= C / n
```

n : number of partitions, **Cn**: optimal area on each partition

Partition number discussion

1)<u>n > 2:</u> Possible RTR partitioning

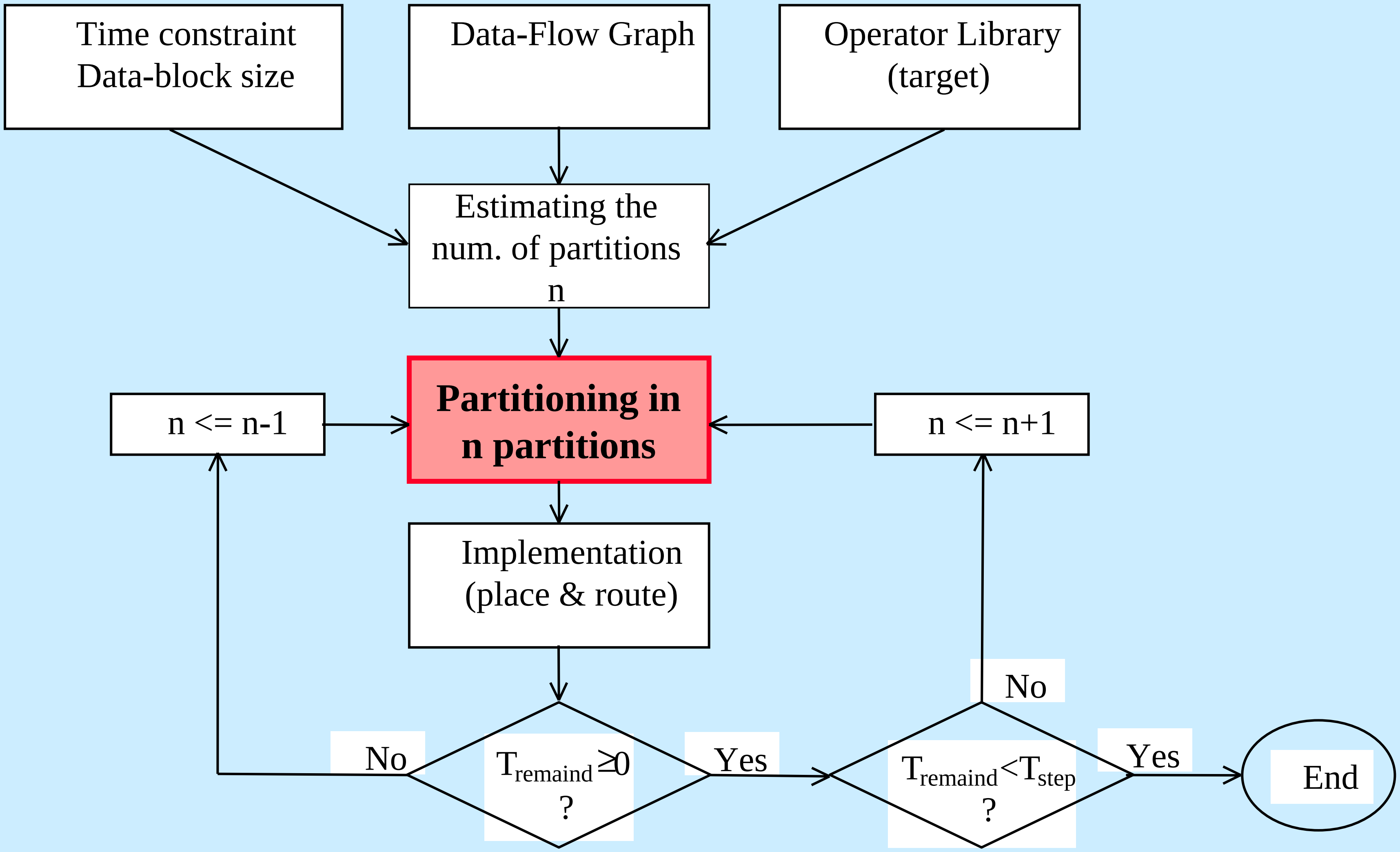
Integer part of **n** → number of partitions

<u>2)n < 2:</u> RTR partitioning is not possible.

If n > 1: Only static implementation is possible.

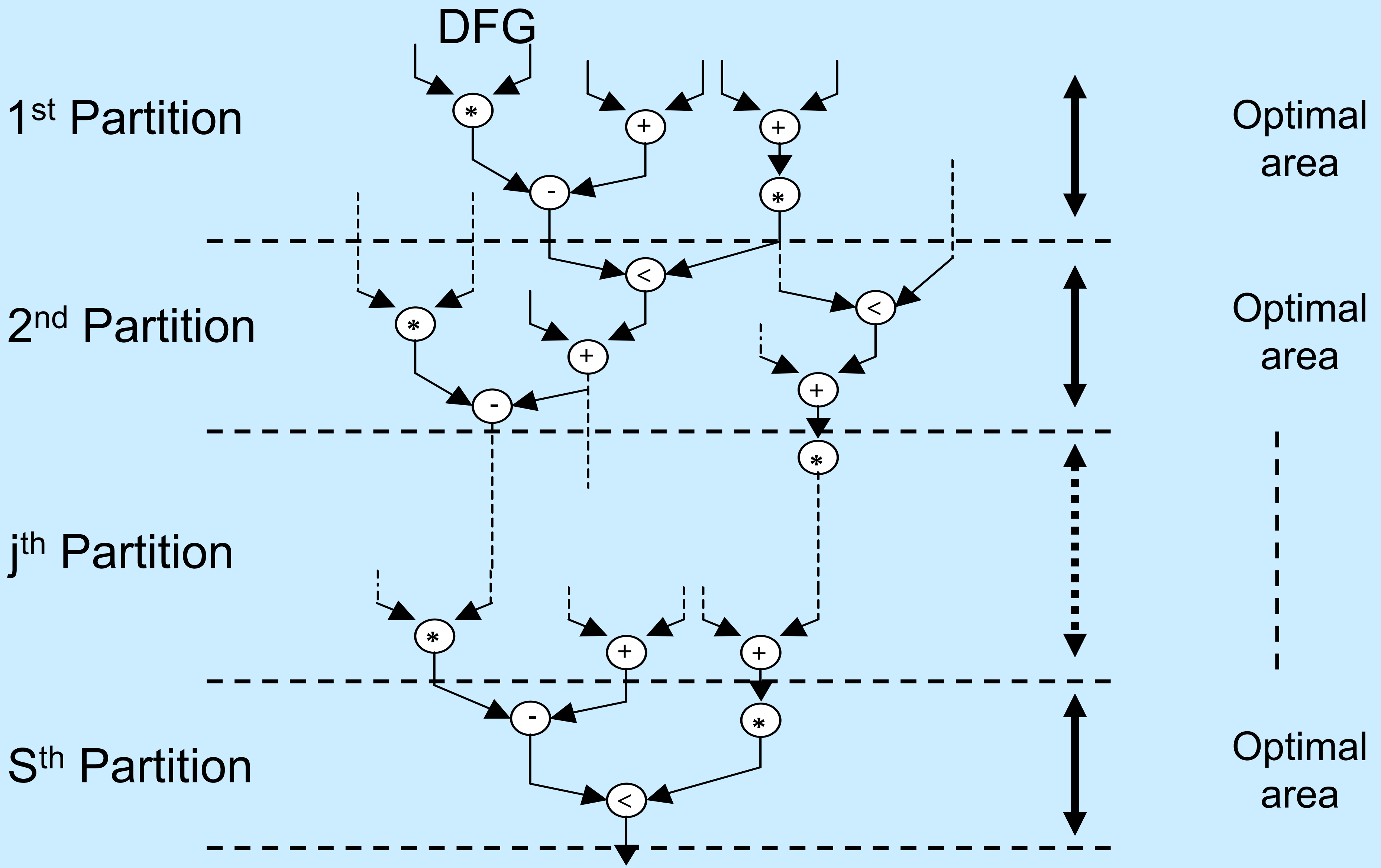
If n < 1: To ensure the constraint it is necessary to modify the algorithm to add a processing parallelism. Integer part of **1/n** gives the degree of processing parallelism.

Automation using Operator Library

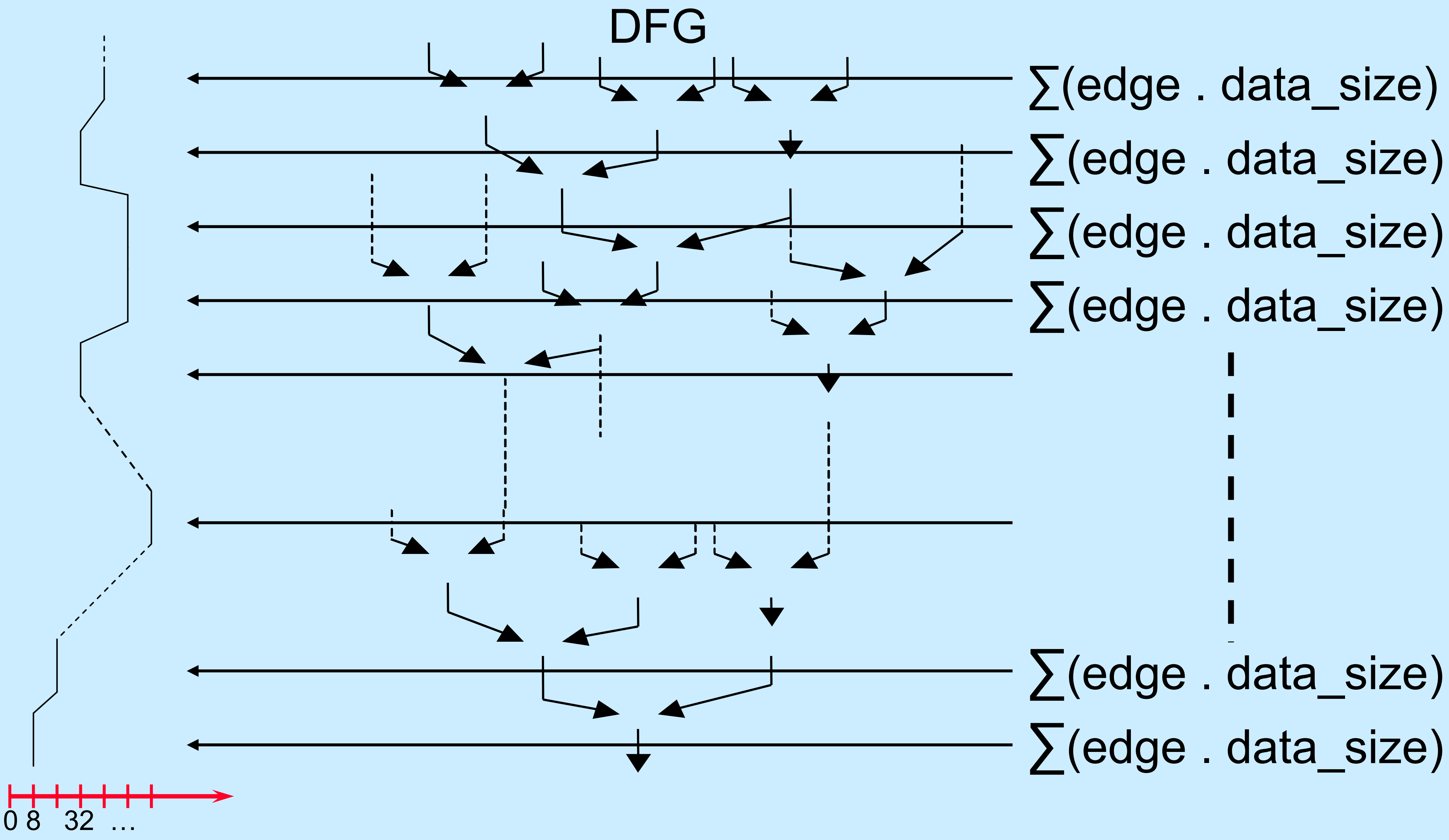


I. INTRODUCTION II. Automatic RTR temporal partitioning III. Application example IV. Conclusion

First partitioning



Memory bandwidth evaluation



$\sum(\text{edge . data\_size})$

$\sum(\text{edge . data\_size})$

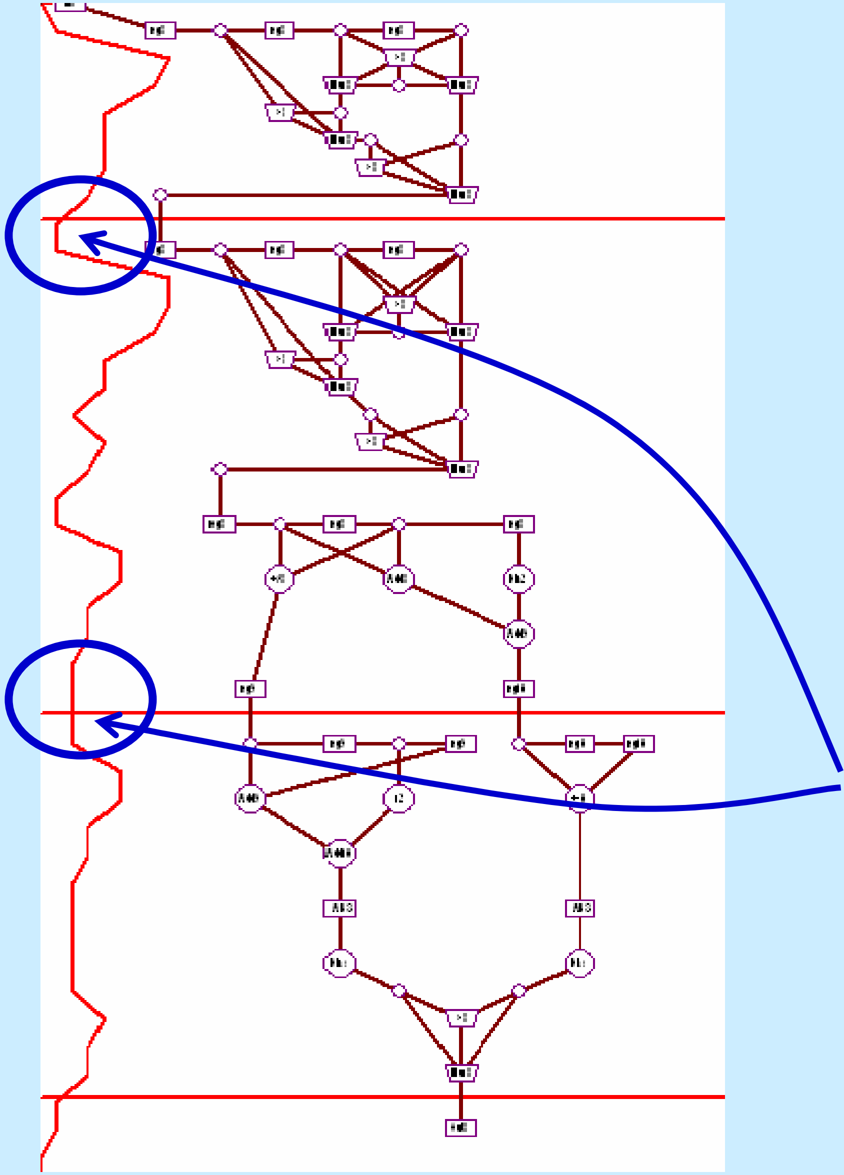
$\sum(\text{edge . data\_size})$

$\sum(\text{edge . data\_size})$

$\sum(\text{edge . data\_size})$

$\sum(\text{edge . data\_size})$

Final partitioning (refinement)

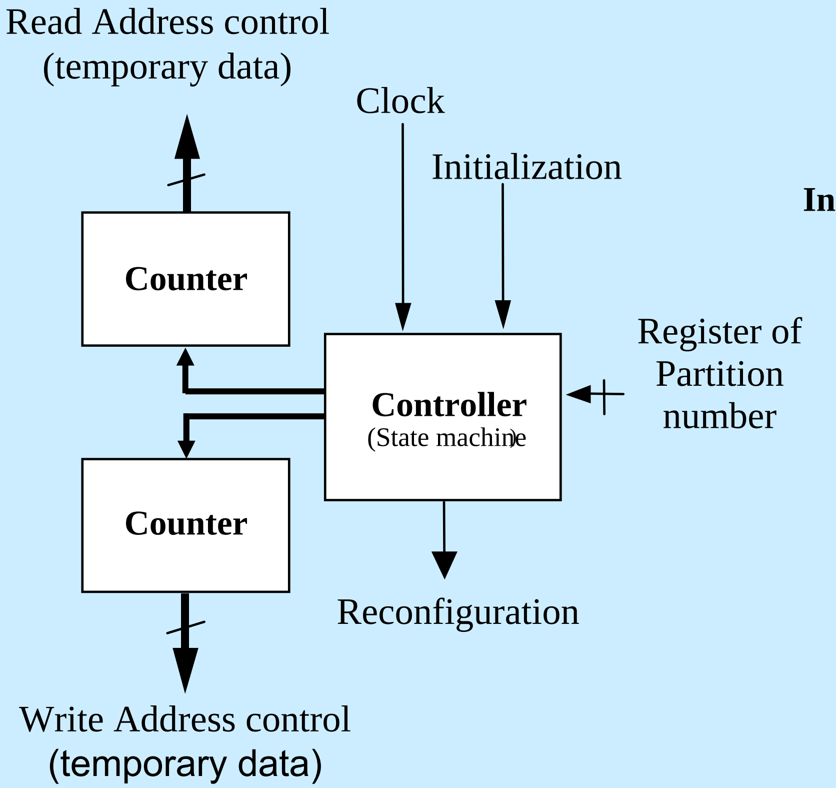


After this first partitioning step, our tool allow to move manually or automatically the splitting boundary to reduce future needs of memory bandwidth.

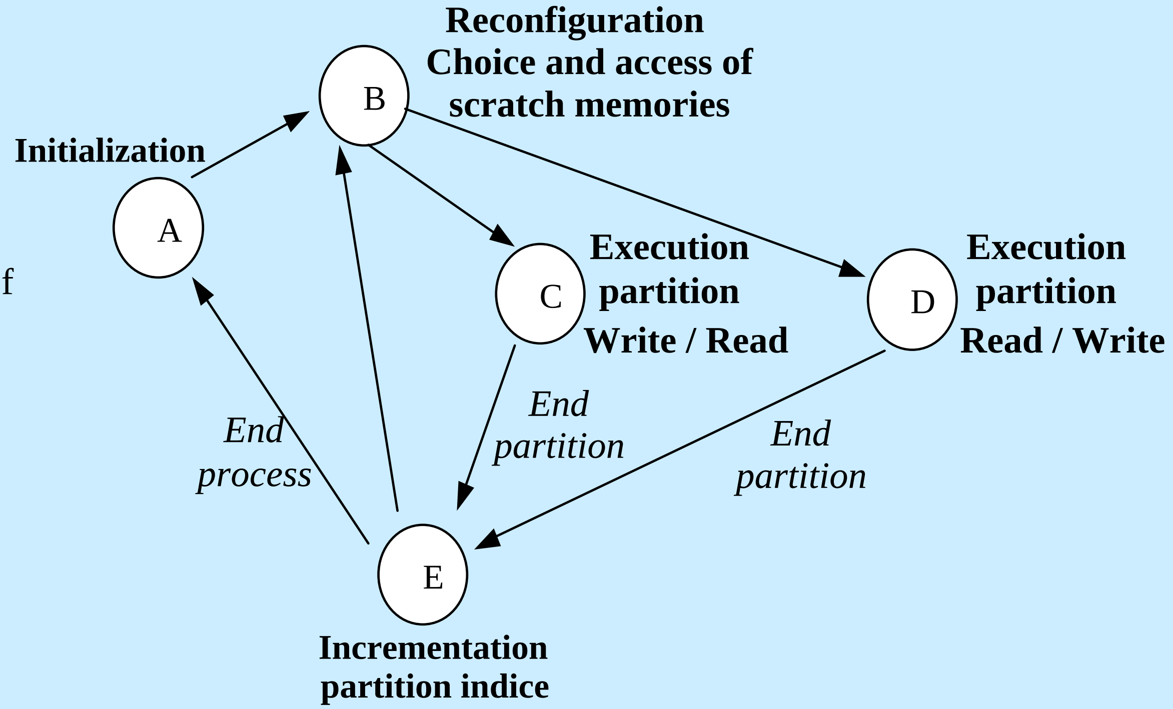
The automated refinement is limited in an adjustable neighborhood of the first splitting to keep future partition's area as homogeneous as possible.

This refinement is done by finding the local minimum of the bus width sum computed before along the DFG.

Configuration and memories controller



Dynamic configuration and memories management



State machine of controller

Outline

I. Introduction

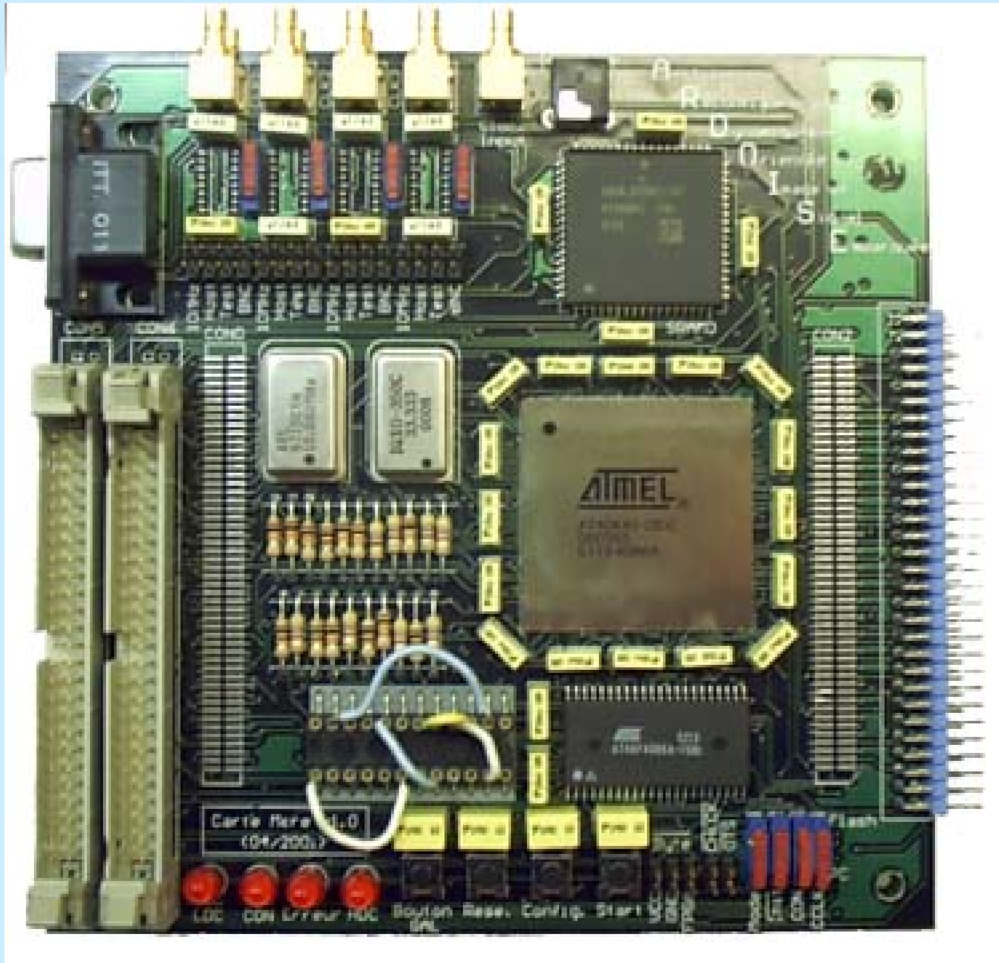
II. Automatic RTR temporal partitioning

<u>III. Application example</u>

IV. Conclusion & Future Works

Hardware plate-form

« ARDOISE » reconfigurable module

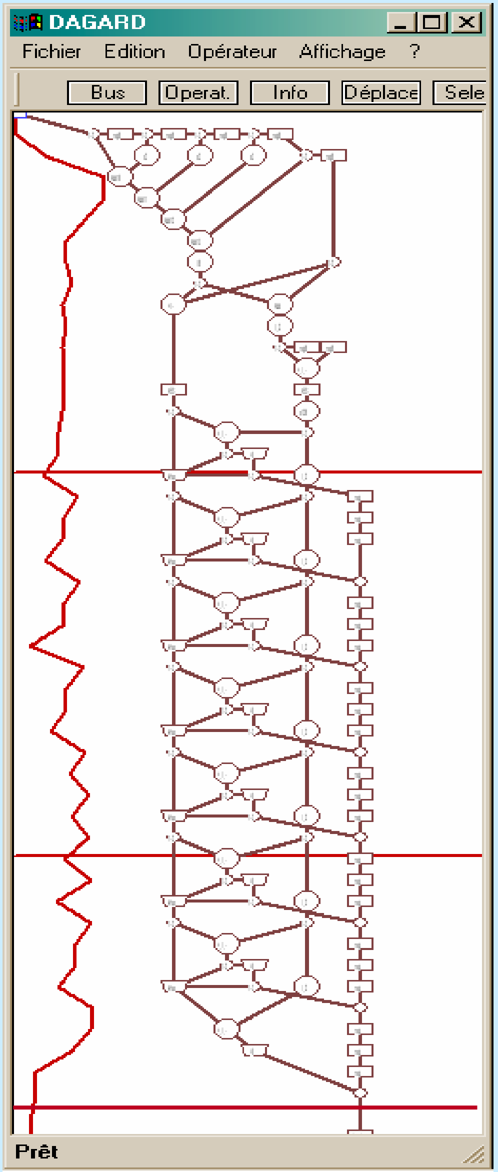


ATMEL AT40K40 FPGA

2304 cells, 1365 cells/ms

Two scratch memories
256K x 32

Implementation example



Application to image processing

Edges motion estimator:

Static implementation	→	896 cells
T (time constraint)	→	40 ms
N(data block size)	→	512^2

Implementation example

Partition	Total number of Cells	Operator execution time (ns)	Total reconfiguration time (µs)	Partition processing time (ms)
1	225	27.1	173	7. 1
2	241	38.7	180	10.15
3	248	38.7	180	10.15
4	294	37.8	190	9.91

Implementation example

Total resources:

Static: 896 cells

Max RTR: 294 cells

Silicon Efficiency improvement:

896/294 ≈ 3

Total processing time ≈ 38 ms

Time remaining ≈ 2 ms

=> To short for another partition

Outline

I. Introduction

II. Automatic RTR temporal partitioning

III. Application example

IV. Conclusion & Future Works

Conclusion & Future Works

We propose a temporal partitioning methodology and tool which:

→ Leads to Better usage of the logical area:

- Increase the logic efficiency,
- Reduction of physical parameters (area),
- Keep application flexibility provided by FPGAs

→ Help designer to:

- Quickly specify his architecture needs
- Quickly estimate if his application can be implemented on his platform.

→ Can be used for SoC including FPGA array

Future Works

More work is needed to simplify RTR design flow :

- → Auto-Synthesis of memory and configuration controller
- → Auto-generation of the VHDL code associated with each partition.
- → Give an estimation of the power consumption and include it in the partitioning.
- → Studying and including non-regular DFG

Thank you for your attention.

Automated RTR Temporal Partitioning for Reconfigurable Embedded Real-Time System Design

C. Tanougast, Y. Berviller, P. Brunet and S. Weber

L. I. E. N.

Laboratoire d’ Instrumentation Electronique de Nancy
Faculté des Sciences de Nancy I, BP239
54506 Vandoeuvre-lès-Nancy, cedex, France

e-mail:
philippe.brunet@lien.uhp-nancy.fr