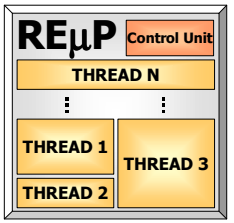


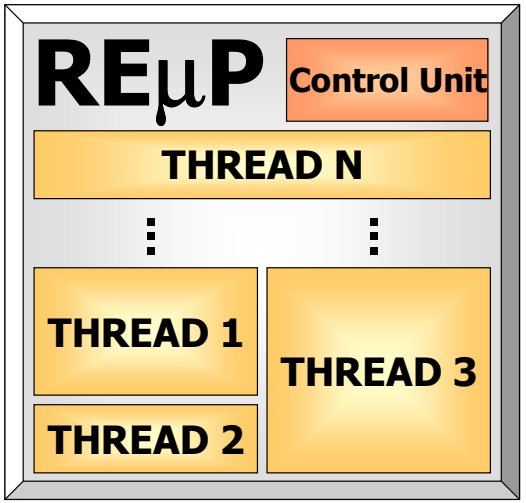
Instructionless processor architecture using dynamically reconfigurable logic

Rafał Kiełbik, Grzegorz Jabłoński, Bartłomiej Świercz, Piotr Amrozik

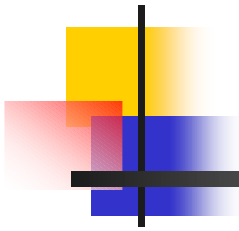
Technical University of Łódź
Department of Microelectronics and Computer Science
Łódź, Poland

Our Idea

Dynamically **RE**configurable FPGA
as general purpose µ**P**rocessor



Configuration bitstreams instead of
a sequence of instructions



Current Trends

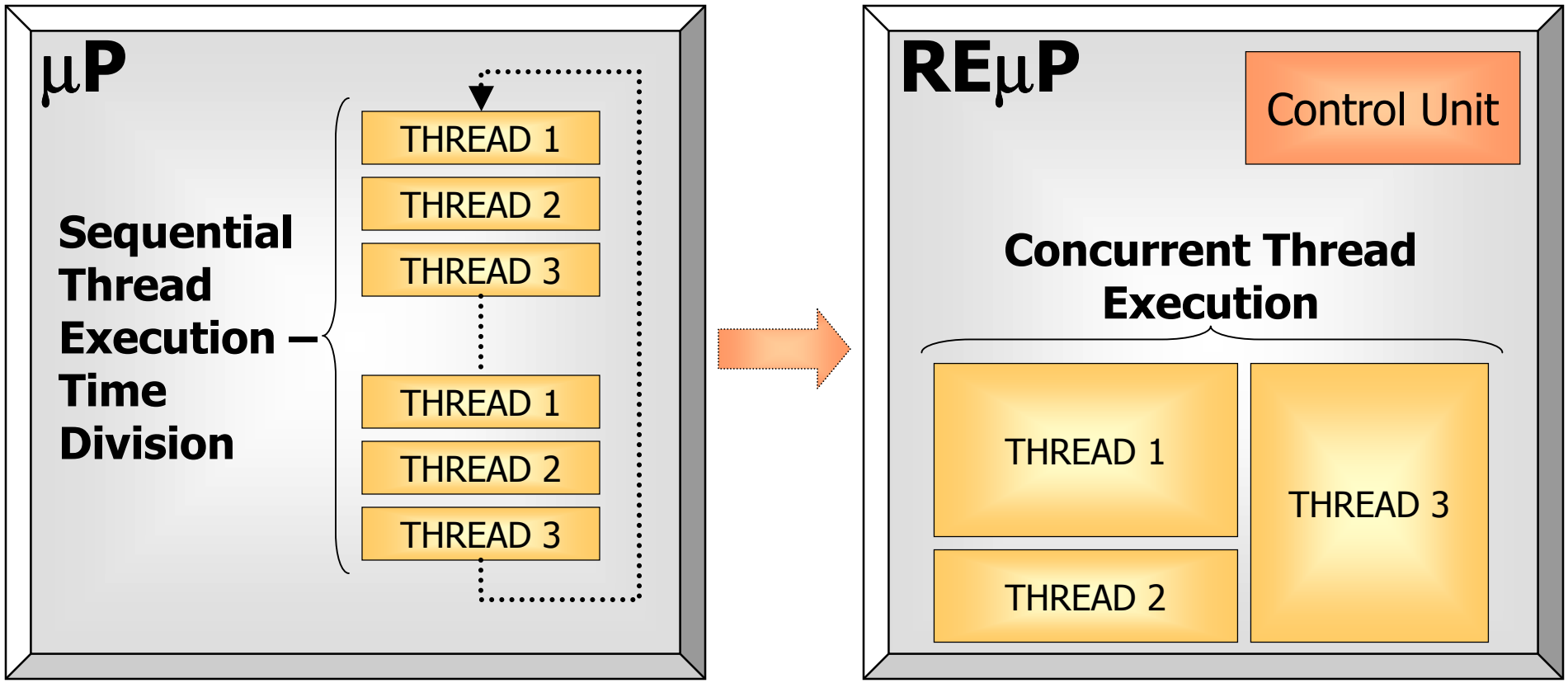
HiPEAC Clusters & Task Forces

- Reconfigurable computing
- Programming models and operating systems
- Adaptive compilation

Typical applications of reconfigurable devices

- **External coprocessors** (for the most demanding calculations)
- **Integrated coprocessors** (with the microprocessor core implemented in FPGA)
- **Application-Specific-Instruction set-Processor** – ASIP (the pipeline structure can be customized and utilized in the program through custom instructions)
- **No-Instruction-Set-Computer** – NISC (the application is mapped directly to the datapath, the pipeline structure remains constant throughout the entire execution of the program)
- **PipeRench, KiloCore** – A virtualized Programmable Datapath

FPGA as Processor



Remarks:

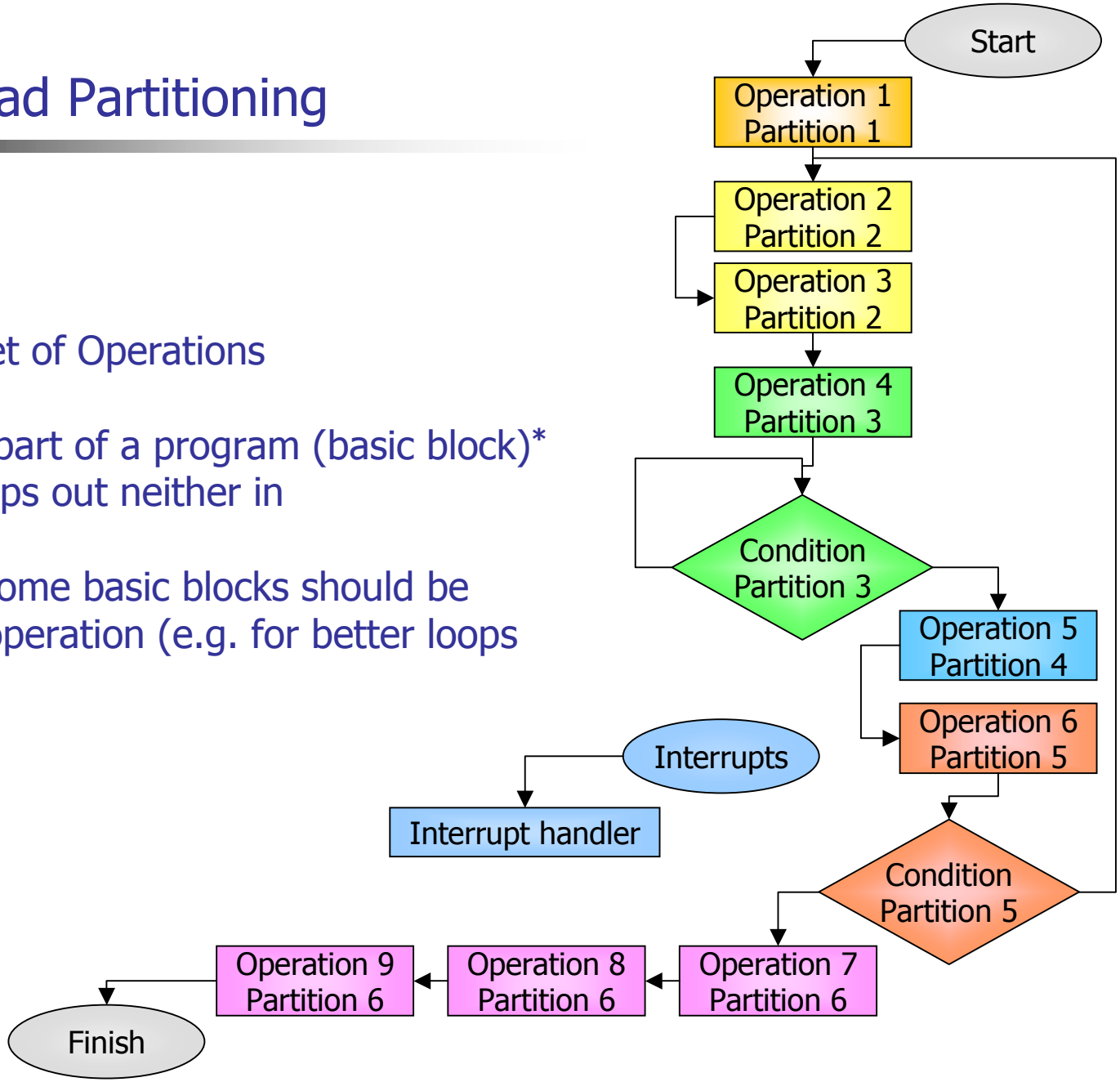
- Concurrent execution of many threads constitutes a **first source of data processing acceleration**.
- Most threads are „too big" to be implemented outright in the FPGA at the same time!!!

Thread Partitioning

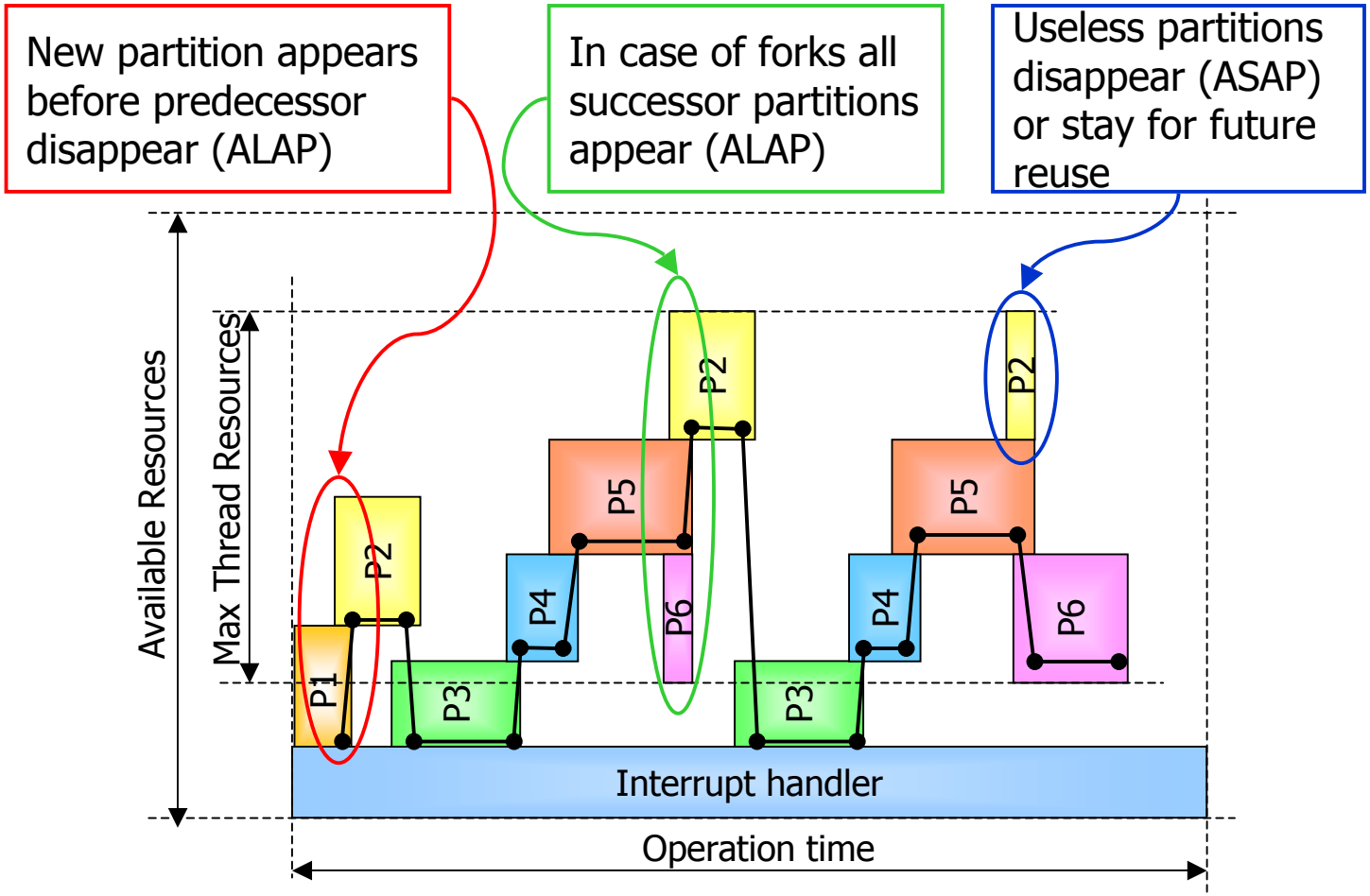
Partition – a set of Operations

Operation – a part of a program (basic block)* without any jumps out neither in

(*) – probably some basic blocks should be merged in one operation (e.g. for better loops optimization)



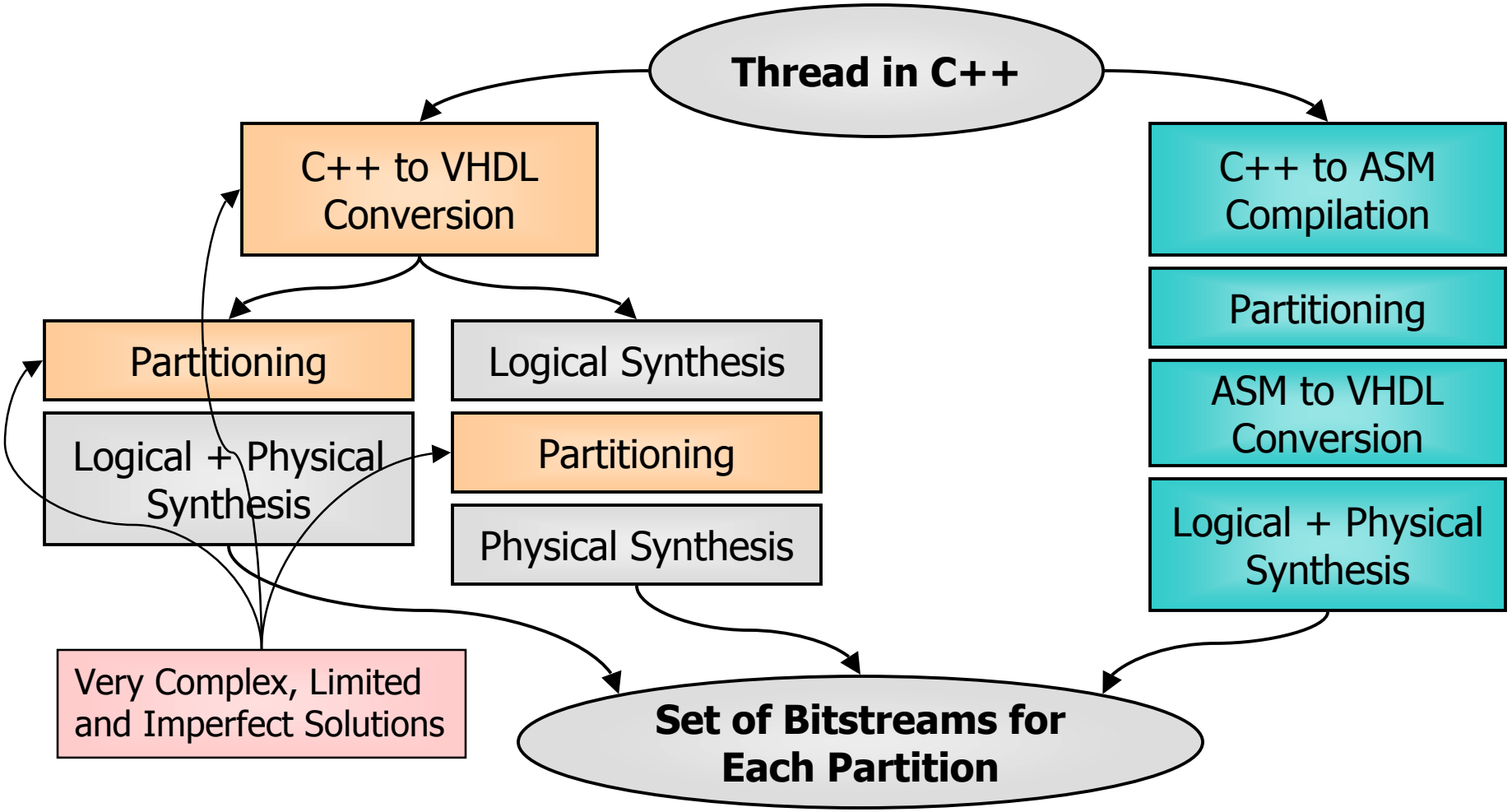
Scheduling the Partitions



Remarks:

- It should be possible to place each partition in any part of FPGA
- Each branching partition must inform the configuration Control Unit about the chosen branch

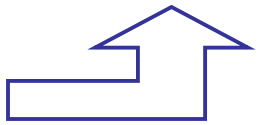
Software to Hardware Conversion



ASM to VHDL Conversion

```
char fun(char a, char b)
{
        char c, d, r;
        c = a * 2;
        d = b * 4;
        r = c - d;
        return r;
}

int main(void)
{
        ...
        c = fun(a,b);
        ...
}
```

```
LDS     R25,0x0186      Load direct from data space
LDS     R24,0x0182      Load direct from data space

LSL     R25             Logical Shift Left
LSL     R24             Logical Shift Left
LSL     R24             Logical Shift Left
SUB     R25,R24         Subtract without carry

STS     0x0183,R25      Store direct to data space
```



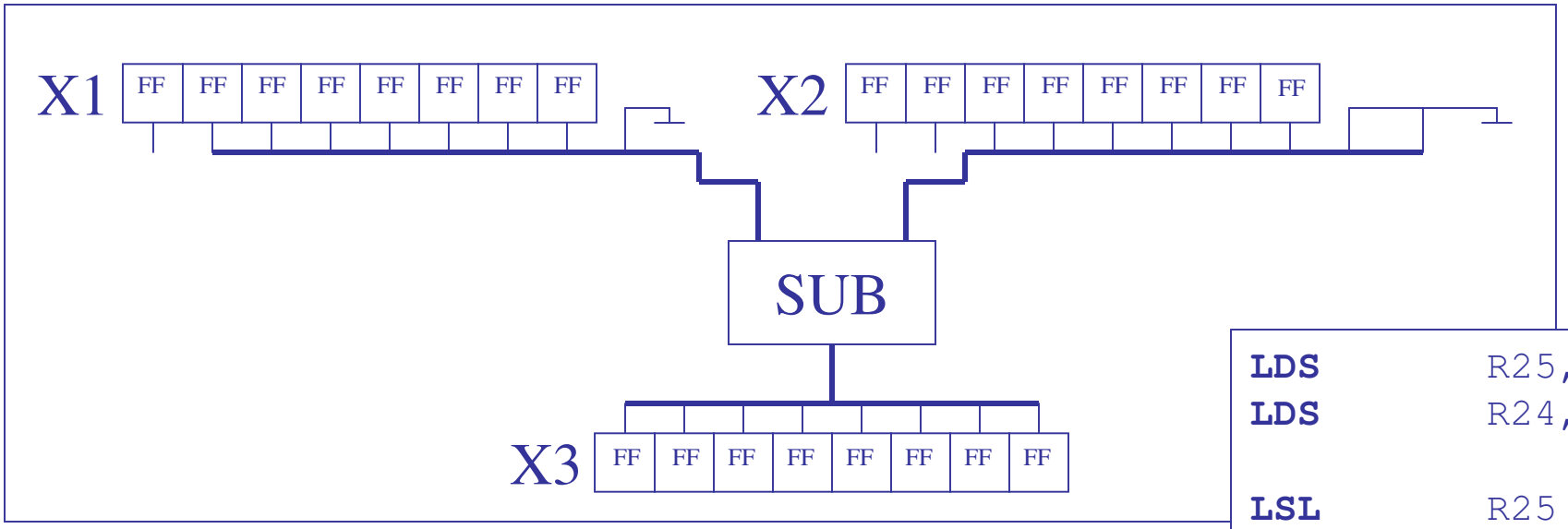
```
LDS   R,X
                        X : in std_logic_vector(N-1 downto 0);
                        variable R : std_logic_vector(X'high downto X'low);
                        R := X;
LSL   R
                        R := R(R'high-1 downto R'low) & '0';
SUB   Rx, Ry
                        Rx := Rx - Ry;
STS   X,R
                        X : out std_logic_vector(R'high downto R'low);
                        X <= R;
```

ASM to VHDL Conversion – Example

```
LDS     R25,X1
LDS     R24,X2

LSL     R25
LSL     R24
LSL     R24
SUB     R25,R24

STS     X3,R25
```

```
entity TEST is
        port(X1, X2 in: ...;
             X3 out: ...);

architecture TEST_behav of entity TEST is
begin

        process(clk)
                variable R25 : std_logic_vector(X1'high downto X1'low);
                variable R24 : std_logic_vector(x2'high downto X2'low);
        begin
                if rising_edge(clk) then
                        R25 := X1;
                        R24 := X2;
                        R25 := R25(R25'high-1 downto R25'low) & '0';
                        R24 := R24(R24'high-1 downto R24'low) & '0';
                        R24 := R24(R24'high-1 downto R24'low) & '0';
                        R25 := R25 - R24;
                        X3 <= R24;
                end if;
        end process;
end architecture;
```

Operation-dedicated Hardware

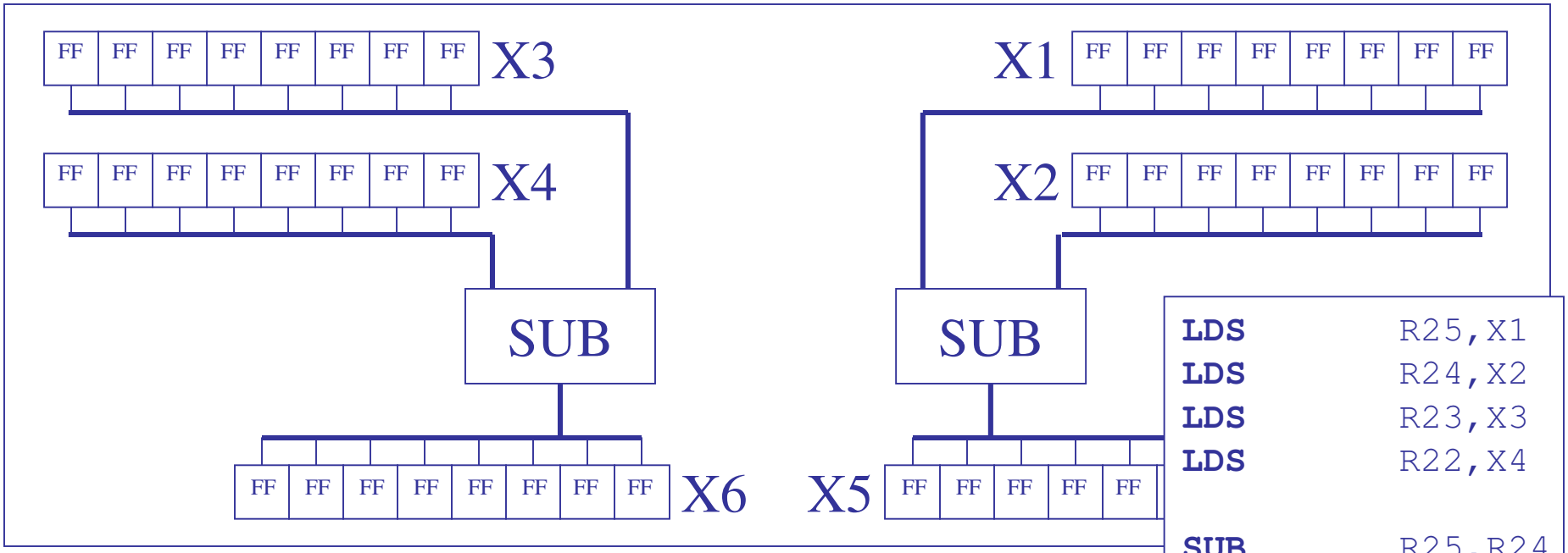



```
LDS     R25,X1
LDS     R24,X2

LSL     R25
LSL     R24
LSL     R24
SUB     R25,R24

STS     X3,R25
```

Remark:

- Generating the operation-dedicated hardware constitutes a **second (and essential) source of data processing acceleration**

Horizontal Parallelism

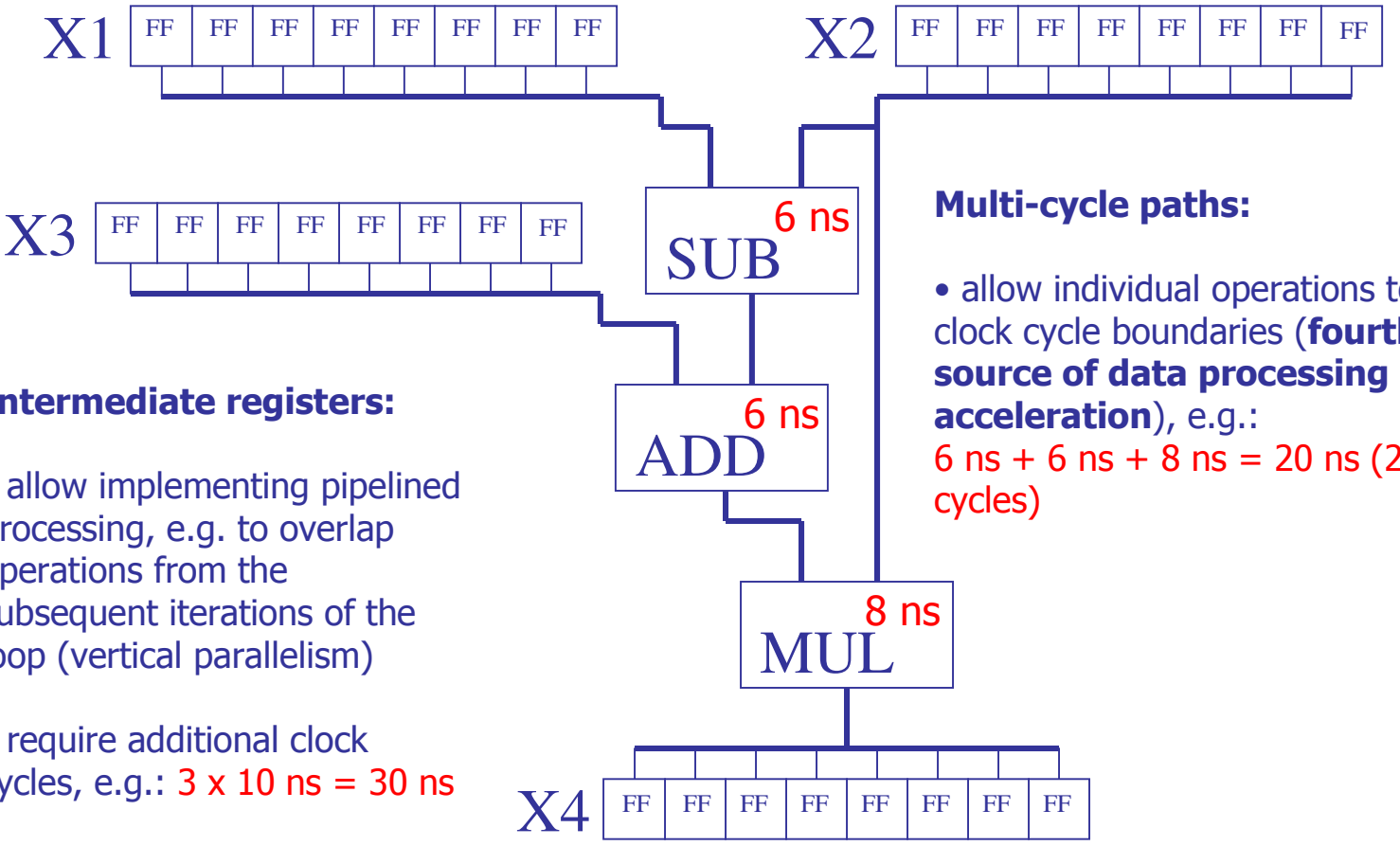


Remark:

• Independent logic or arithmetic instructions in the given software operation will be implemented separately and executed concurrently, without any effort on the programmer's side. This feature constitutes a **third source of data processing acceleration**

Long Combinational Paths

Clock period: 10 ns



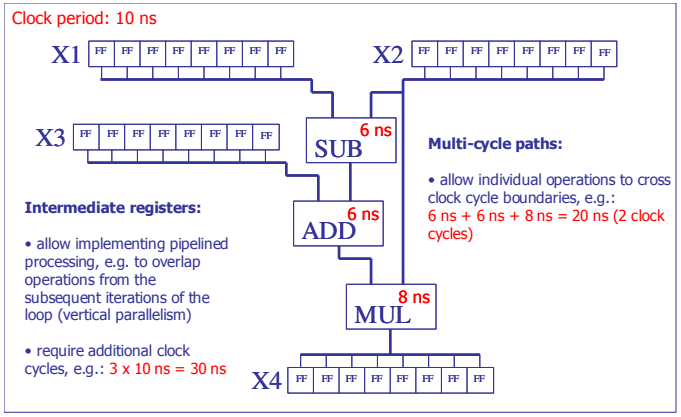
Intermediate registers:

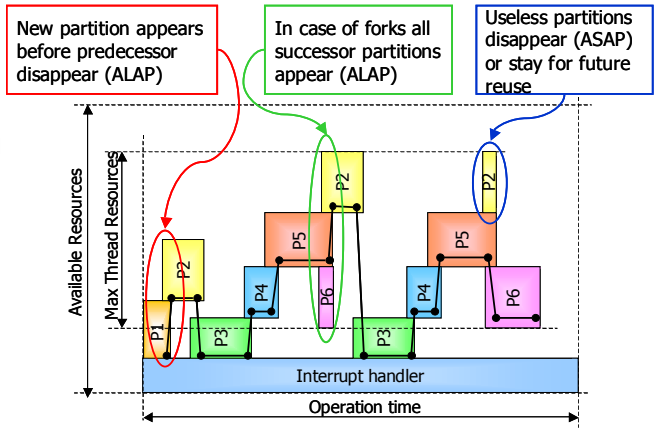
- allow implementing pipelined processing, e.g. to overlap operations from the subsequent iterations of the loop (vertical parallelism)
- require additional clock cycles, e.g.: 3 x 10 ns = 30 ns

Multi-cycle paths:

- allow individual operations to cross clock cycle boundaries (**fourth source of data processing acceleration**), e.g.:
  6 ns + 6 ns + 8 ns = 20 ns (2 clock cycles)

Reconfiguration Time

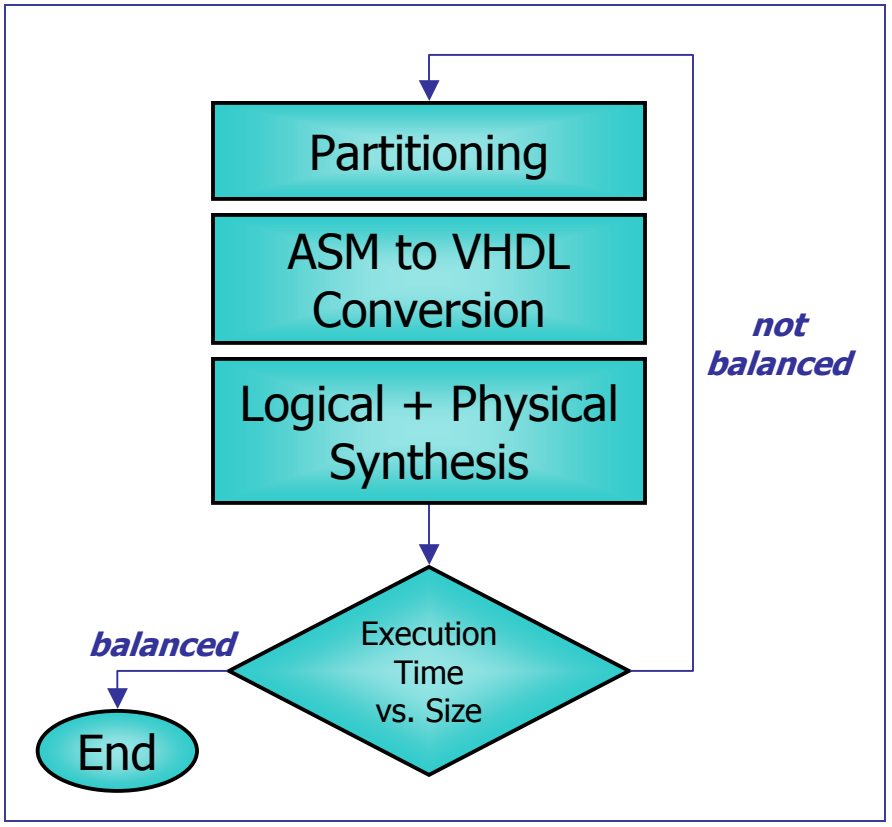




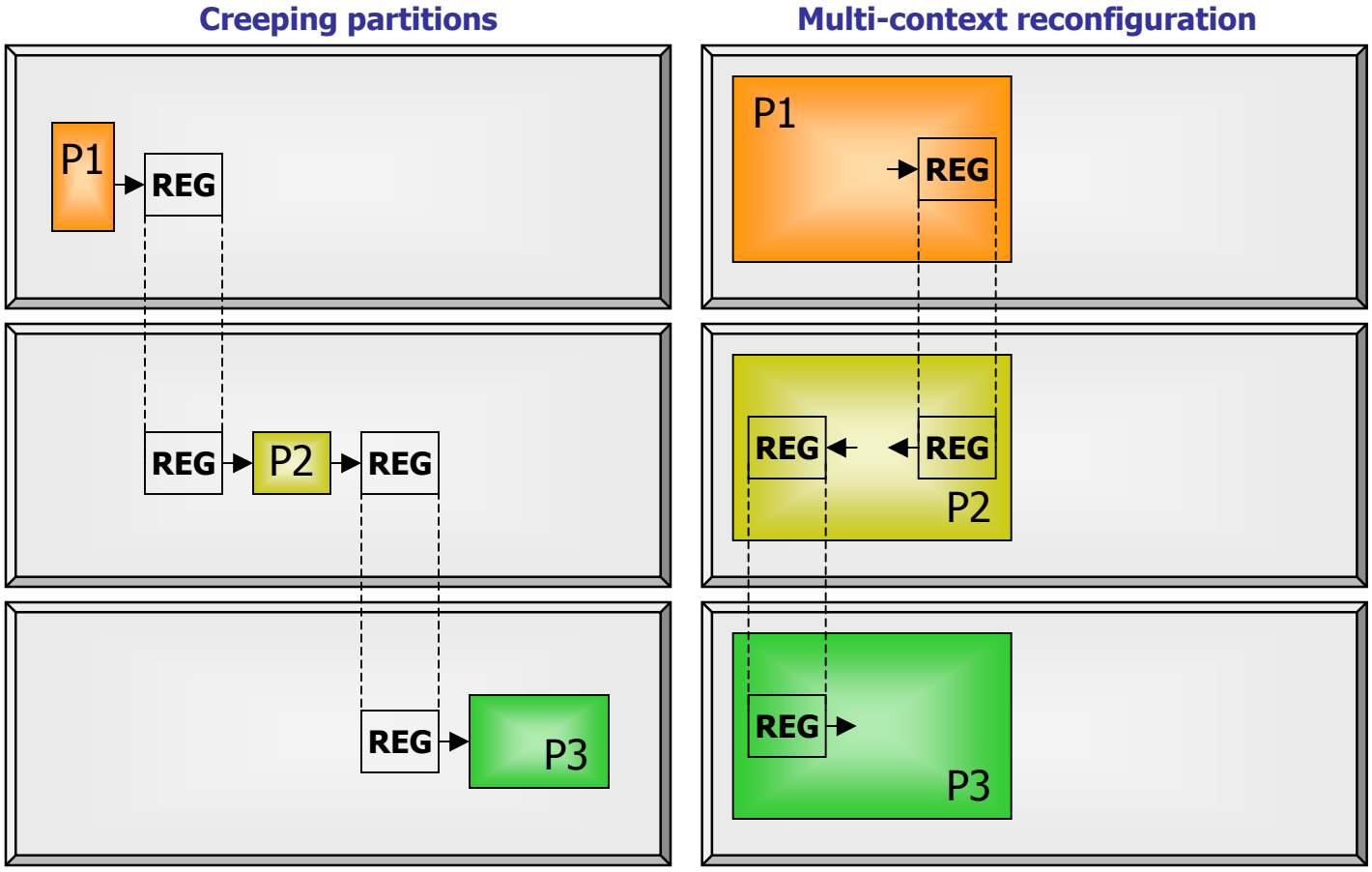
Remarks:

- Regardless of chosen technique of achieving the timing closure, the exact number of clock cycles for each partition will be determined during the synthesis process

- The size and the execution time of each partition should be balanced in order to have the possibility to:
  - configure the successive partition during the previous partition is „running”
  - have the possibility to implement as many concurrent partitions (threads) as possible



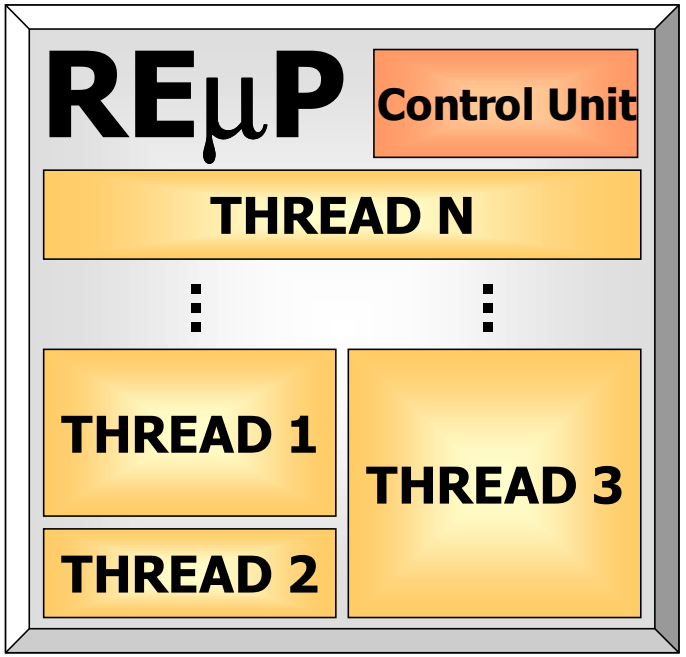
Reconfiguration Technique



Remark:

- Current off-the-shelf reconfigurable devices **do not** offer the demanded reconfiguration flexibility

Operating System



Required functionality

- Common programming interface (POSIX)
- Interrupt management
- Memory management
- Inter-process communication and synchronization
- Scheduling and **placing** the partitions to fully utilize the reprogrammable array, taking into account the:
  - priorities of partitions
  - sizes of partitions
  - dependencies among partitions
  - memory access requests of partitions
  - locked partitions
- **Loading** the partition **bitstreams**
- **Routing** the partitions

Remark:

- It must be stressed that the **placement** is understood here as a selection of the appropriate location of the entire partitions in the fabric. The placement in the traditional sense is performed during the bitstream generation phase. Similarly, the **routing** connects the interfaces of the internally routed partitions

Conclusions

- Interdisciplinary project
  - Computer architecture
  - Reconfigurable computing
  - Compilers
  - Operating systems
  - Application Specific Integrated Circuits
- High processing speed-up potential (four sources of acceleration)
- Energy saving solution (efficient resource utilization)
- Revolutionary architecture without impact on software development
- Dedicated dynamically reconfigurable ASIC required

Thank you