

Algorithm for processor datapath design from particular control flow blocks datapaths

Istraživački seminar iz računarske znanosti

Danko Ivošević

13. rujna 2012.



**Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva**



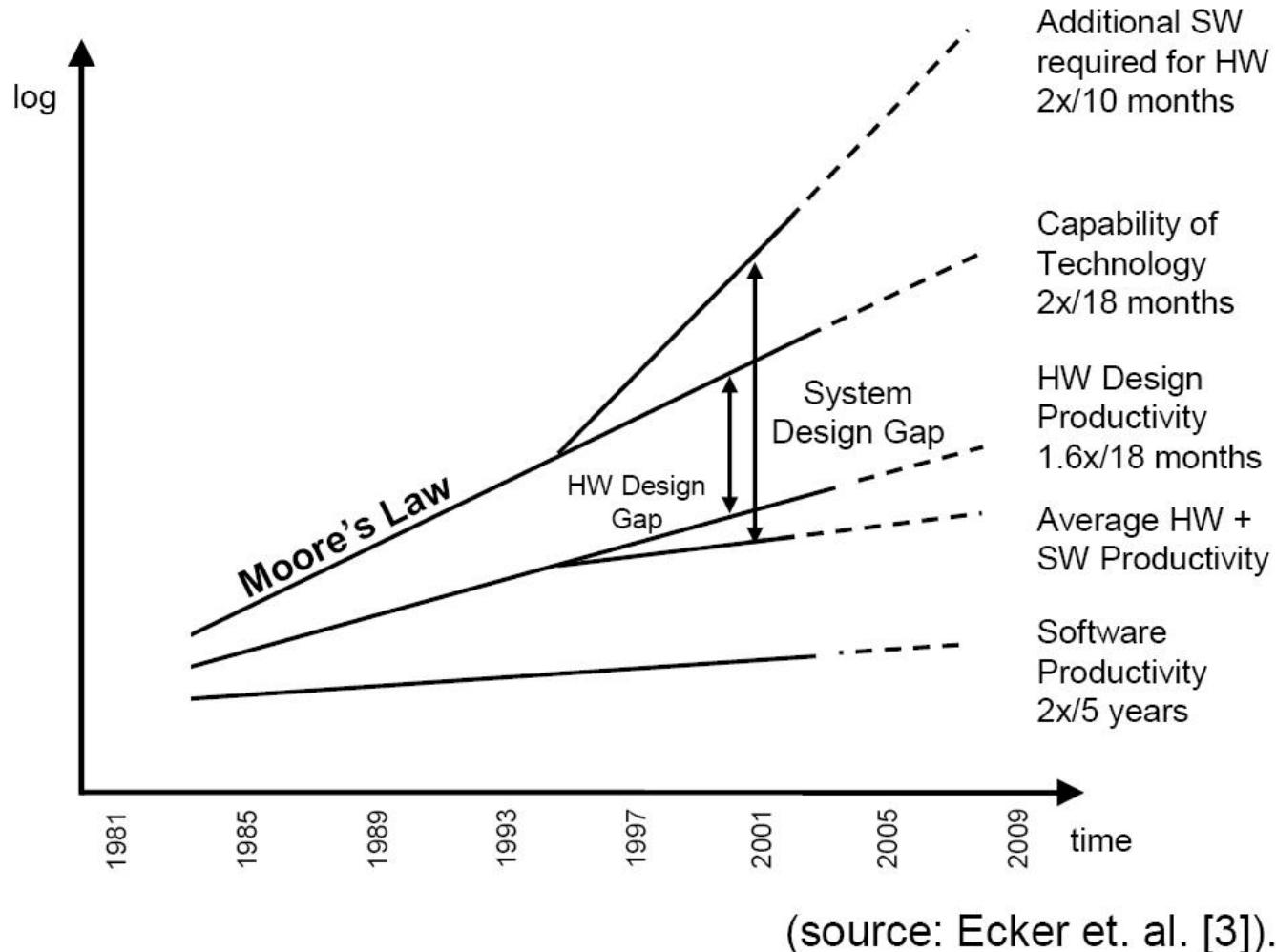


Contents

- **High-Level Synthesis References**
- **Automated Processor Modeling:**
 - **Flow**
 - **Algorithm**
- **Results**
- **Conclusion**



Motivation





References (I)

■ High-Level Synthesis (C-to-Hardware):

[Gaj92]

- Lexical Analysis
- Algorithm Optimizations
- Control and Data Dependencies Analysis
- Technology Library Processing
- Resource Allocation
- Operation Scheduling
- Functional Units & Register Binding
- Output: RTL (Register Transfer Level) Code

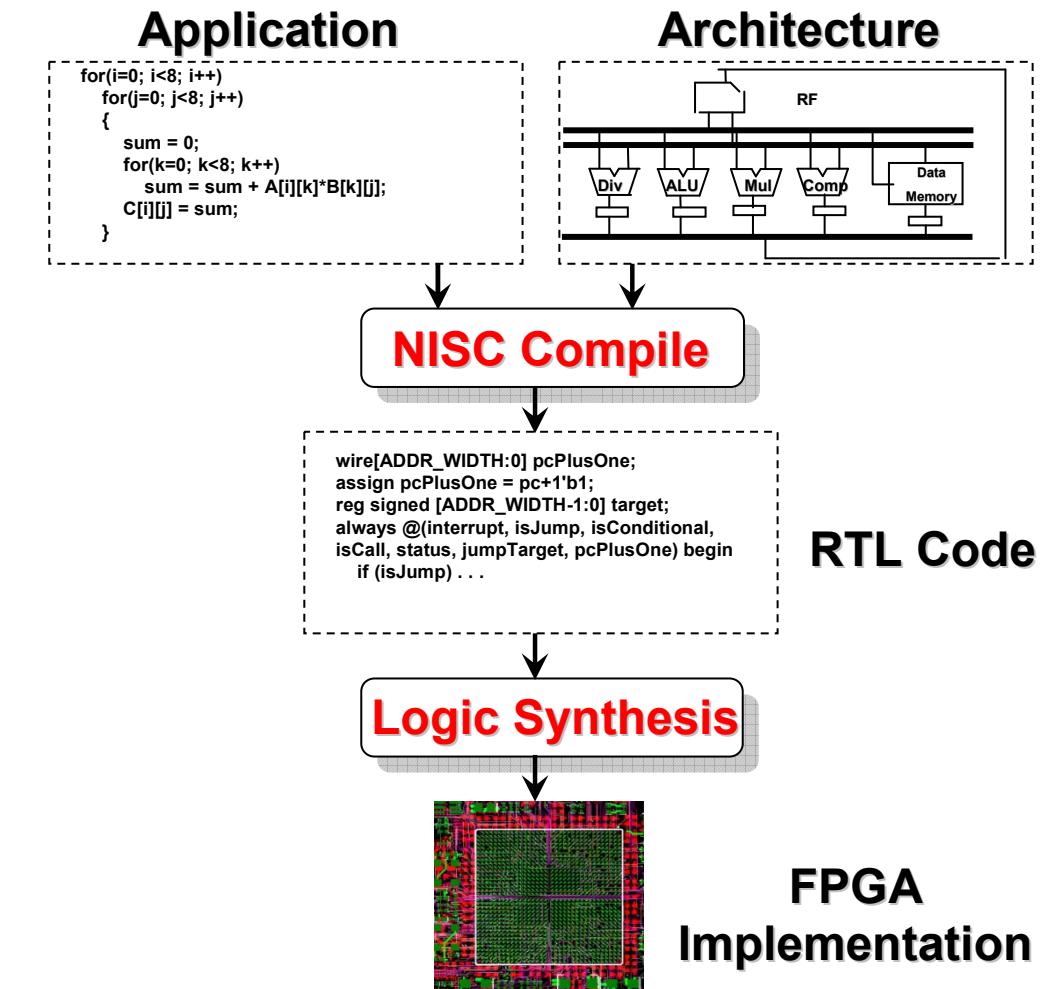
References (II)

- No-Instruction-Set Computer Concept

[CECS]

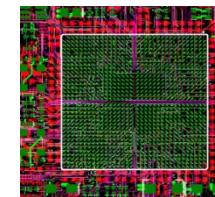
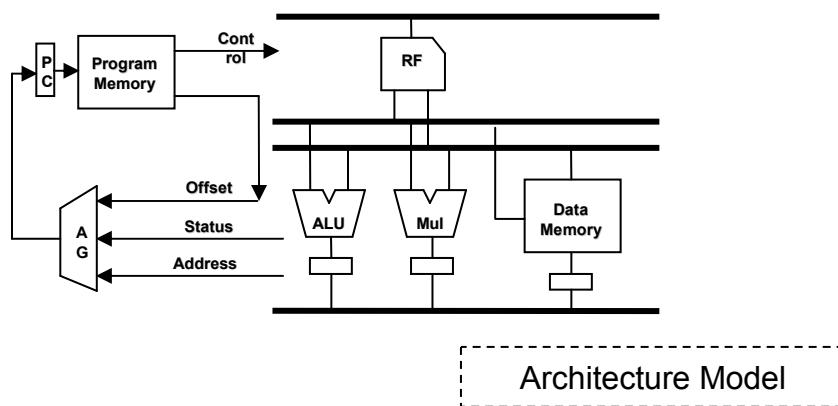
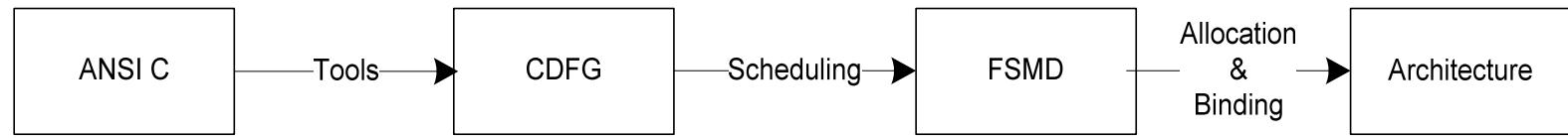
- Automated Processor Modeling

[NISC]





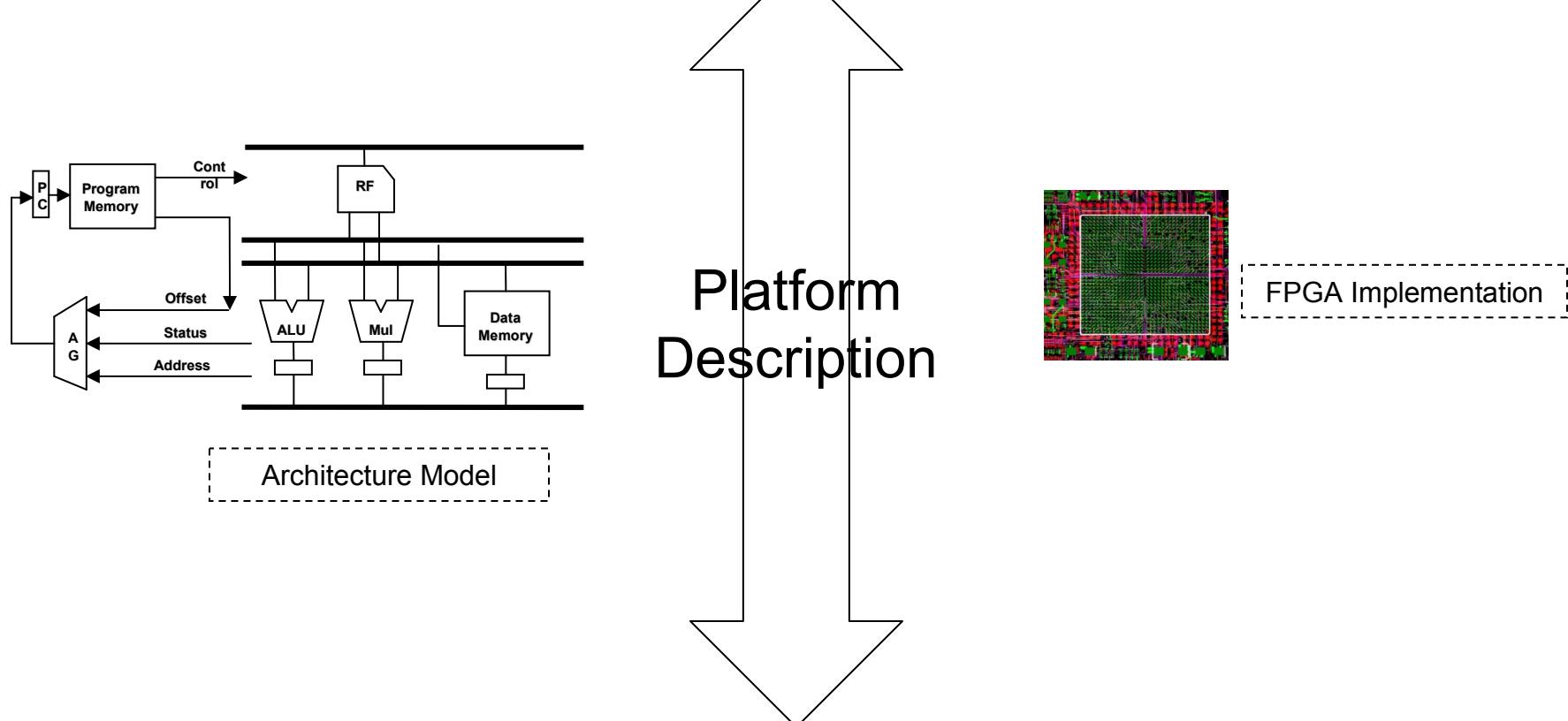
Proposed Flow



FPGA Implementation



Proposed Flow



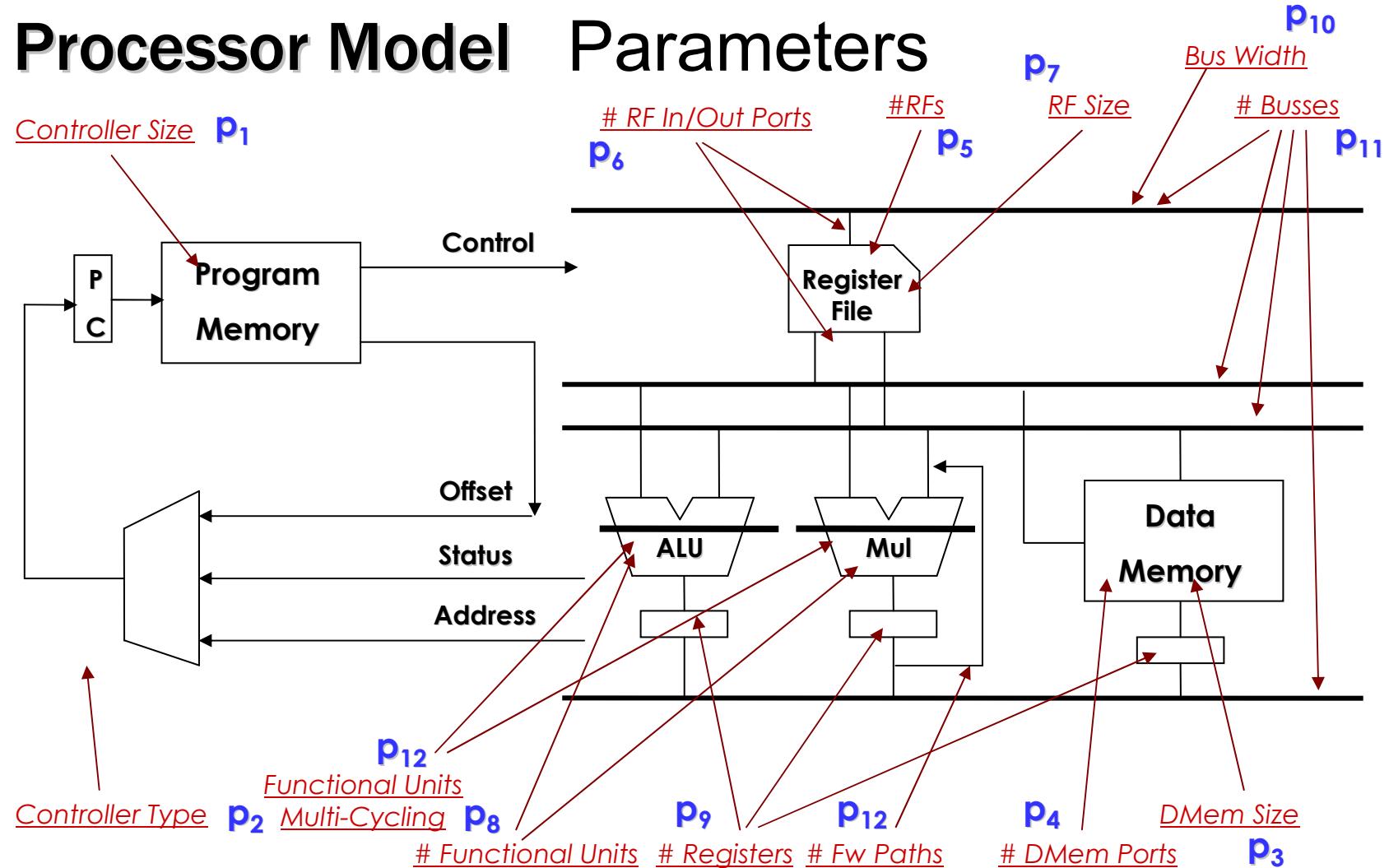
10.9.2012

7/29



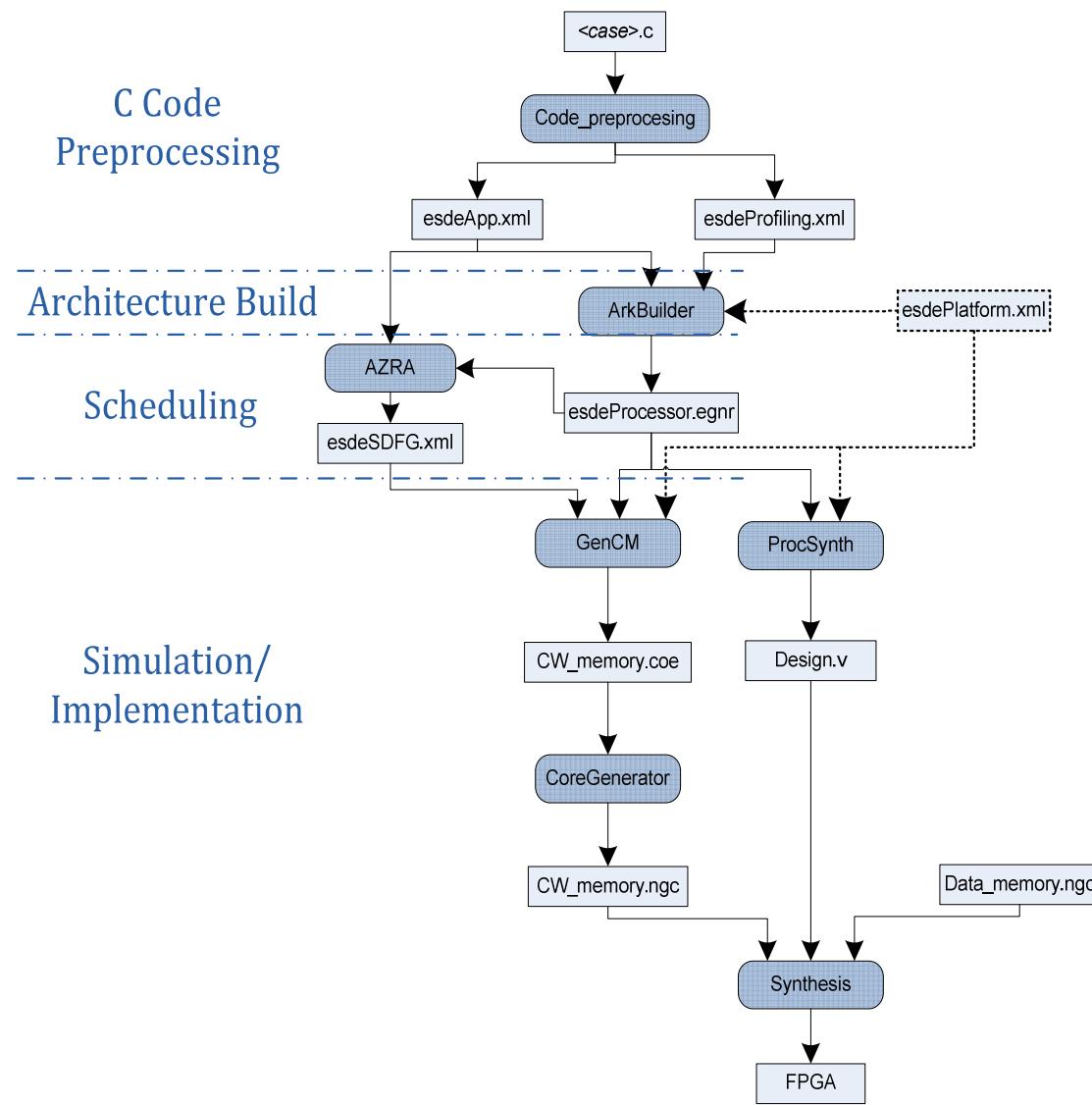
Public Talk (July 2009)

Processor Model Parameters





Developed Toolset





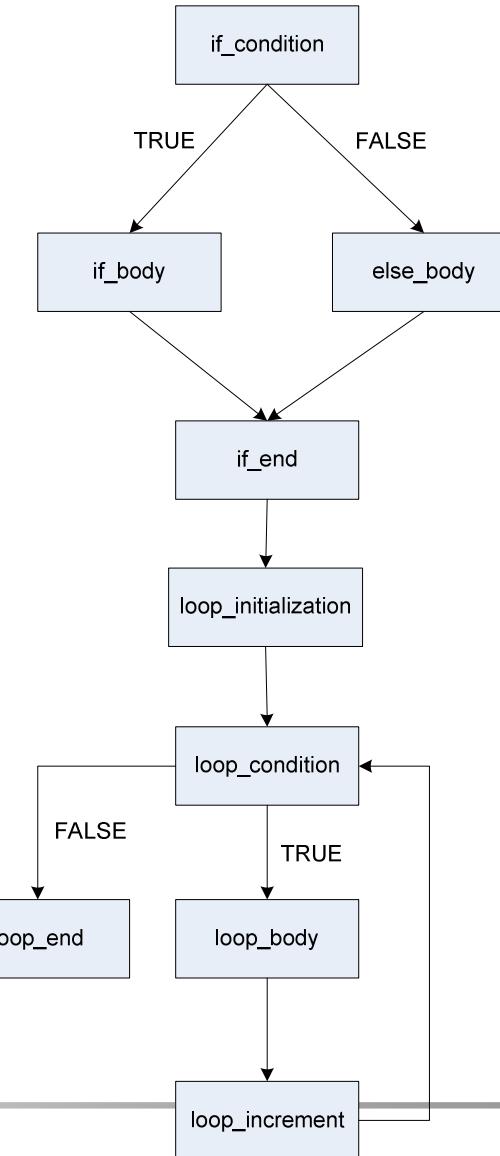
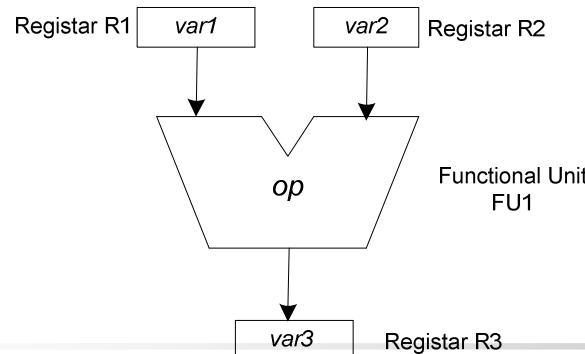
Flow – Intro

- C Code Preprocessing
- Hierarchical split
 - By modules
 - By procedures

Flow – CDFG Generation

■ CDFG Generation

- per routine
- Control Flow
- Basic Blocks of Code
- Three-Address Notation:
 - $\text{var3} = \text{var1 op var2}$





Flow – Scheduling and Optimization

■ Scheduling:

- ASAP
- ALAP
- RC

■ Allocation, Binding and Optimization:

- Usage Tables
- Compatibility Graphs



Flow – Scheduling

- RC with special constraints:
 - Data Memory Ports constraint
 - Allowed combination of units (*inside Platform description*)
 - Multicycling units support



Flow – Allocation, Binding, Optimization

■ Usage Tables Per Cycle:

- Variables
- Operations
- Connections

■ Compatibility Graphs:

- Variable Lifetimes
- Operation Overlaps



Flow – Profiling

- Global Flow Capturing
- Application Profiling
 - At (between) Routine Level
 - At Basic Block Level (in Routine Level)
- Basis for performance estimation:
 - Cycle count



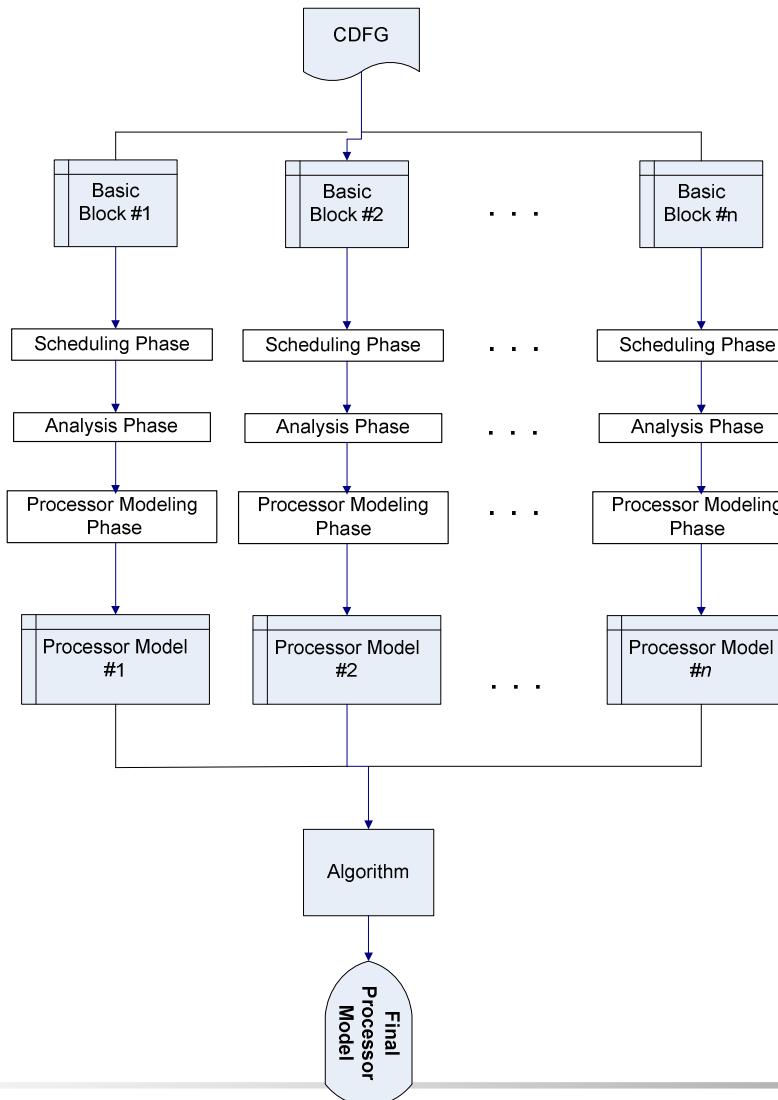
Flow – Integration Algorithm

■ Bases

- Profiling at basic block level
- Sorting basic blocks by significance
- Substatements dedicated to functional units
- Inputs:
 - C code – CDFG
 - Platform description
- Algorithm:
 - *for every basic block from sorted list*
 - *{ ... build final architecture ... }*



Flow – Integration Algorithm





Flow – Integration Algorithm

■ Integration of Architectures

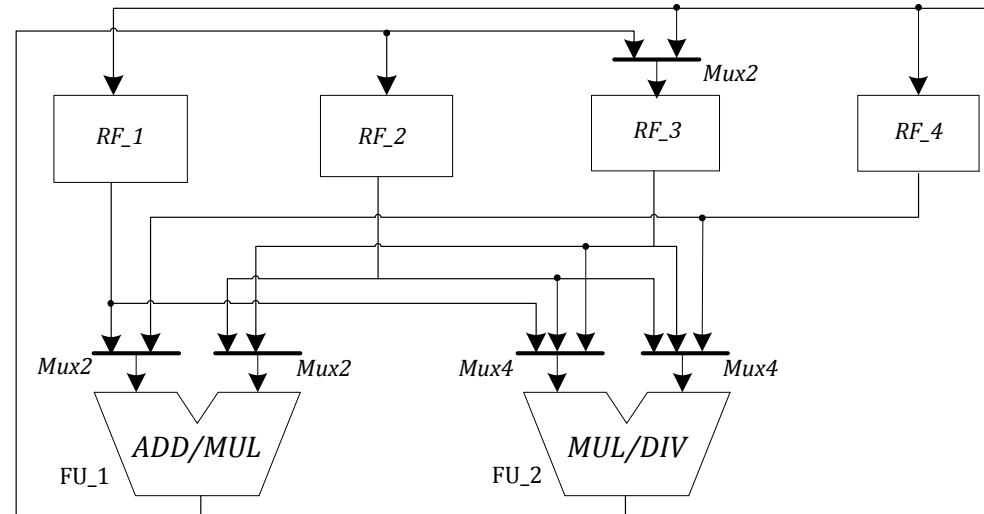
- Add components incrementally
- Test allocation – platform limitations
- Check “accompanying logic”
- Algorithm:
 - *For every basic block from sorted list*
 - *For every functional unit sorted by significance*
 - *Check allocation with accompanying logic*
 - *If fits, add it to final architecture*
 - *If not, stop and assign remaining substatements*
 - *Check connectivity and conflicts*

Integration Algorithm

■ Example Code:

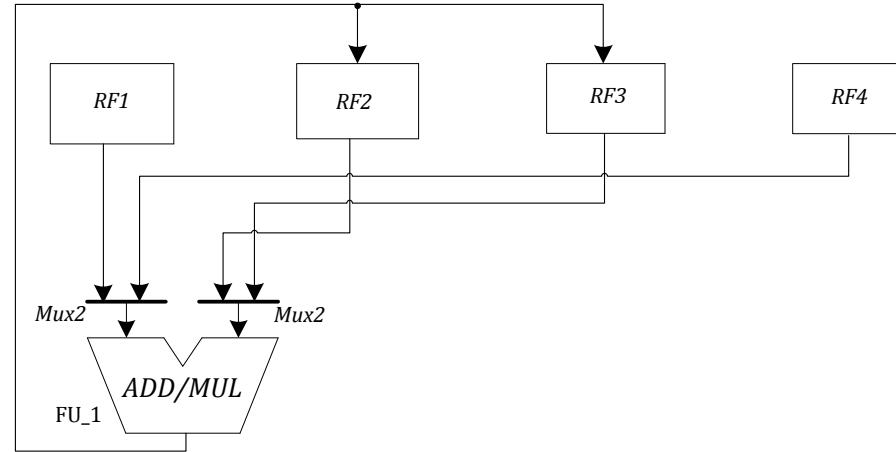
■ $g = ((a * b) + (c * d)) / (e * f);$

■ Expected Datapath:

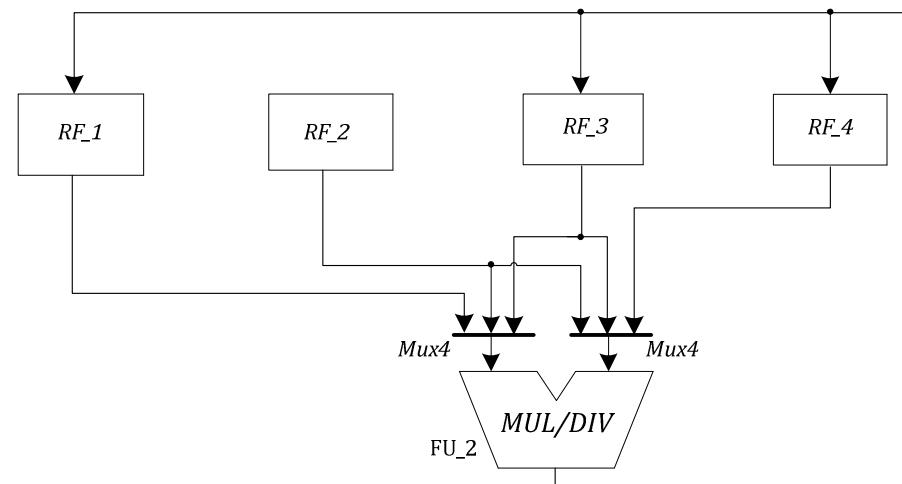


Integration Algorithm

- FU_1 with accompanying logic:



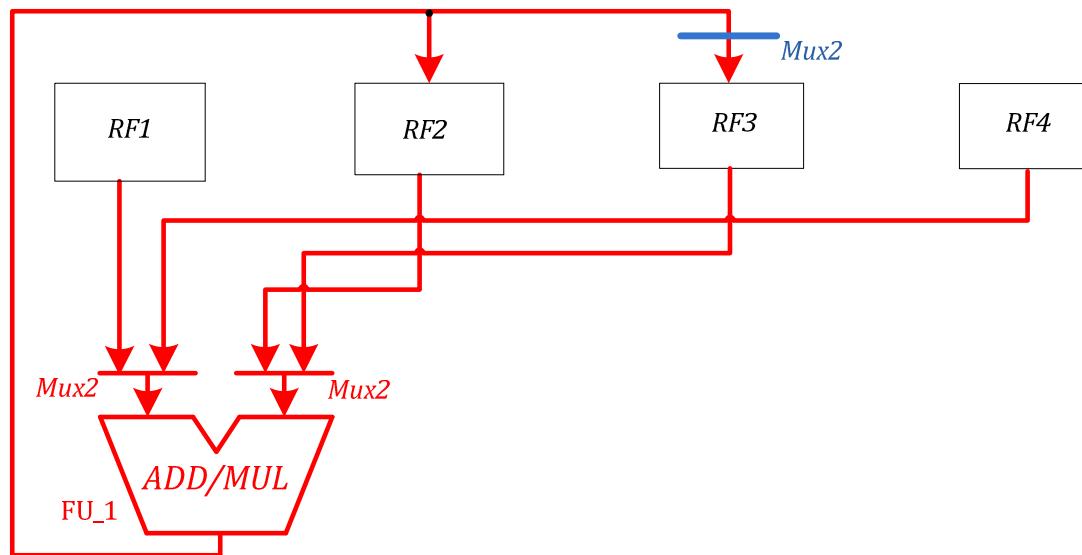
- FU_2 with accompanying logic:





Integration Algorithm

- Incremental approach:
 - **FU_2** is added first
 - Contribution (red) of **FU_1** (new **Mux** required):





Integration Algorithm - PseudoCode

```
DP = ∅
Occtotal = ∅
Existsop = NOTE_OPS(CDFG), op ∈ {ADD, SUB, ..., ASSIGN}
FUmaxop = GET_MAX_INSTANCES(CDFG), op ∈ {ADD, SUB, ..., ASSIGN}
base_logic = [CTRL, DMEM_WRAP]
DP += base_logic
Occbase = ESTIMATE_OCC(base_logic) + ESTIMATE_OCC(fuop, op ∈ Existsop)
Occtotal += Occbase
BBsorted = SORT_BB(CDFG, desc)
za bbcurrent ∈ BBsorted
    ako Occtotal < Resplatform
        ACCOMPdmem = ACCOMP_LOGIC(DMEM(bbcurrent))
        OccACCOMPdmem = ESTIMATE_OCC(ACCOMPdmem)
        ako (Occtotal + OccACCOMPdmem) < Resplatform
            DP += ACCOMPdmem
            Occtotal += OccACCOMPdmem
    ináče
        UPDATE_DP_RFS(bbcurrent)
    FUsorted = SORT_FU (bbcurrent, desc)
    za fucurrent ∈ FUsorted
        Op= GET_OPERATION(fucurrent)
        ako FUinstancesop < FUmaxop
            ACCOMPfucurrent = ACCOMP_LOGIC(fucurrent)
            OccACCOMPfucurrent = ESTIMATE_OCC(ACCOMPfucurrent)
            ako (Occtotal + Occfucurrent + OccACCOMPfucurrent) < Resplatform
                DP += fucurrent + ACCOMPfucurrent
                Occtotal += Occfucurrent + OccACCOMPfucurrent
    ináče
        UPDATE_DP(bbcurrent)
    ináče
        UPDATE_DP(bbcurrent)
    ináče
        UPDATE_DP(bbcurrent)
OPTIMIZE(DP)
UPDATE_CTRL_CONNS(DP, CTRL)
```

Terms:

- Total allocation
- Accompanying logic
- Final architecture



Integration Algorithm

■ Key Questions:

- What is “minimal architecture”?
- Tradeoff between performance and allocation
 - If there are free resources on platform is it worth to spend it?



DCT - Results

#instances	ADD	MUL	COMP	CONV	RFs	Mux2	Mux4	Mux8	Mux16	#cycles
unlimited	5	2	3	4	17	10	7	2	0	10830
<=3	3	2	3	2	16	10	9	2	0	10830
<=2	2	2	2	1	17	7	8	5	0	10830
<=1	1	1	1	0	15	3	8	2	2	10830



DCT - Results

#instances	Occupation Estimation		Real Occupation		Min. Period /ns	Max. Frequency/ MHz
	SLICES	DSP SLICES	SLICES	DSP SLICES		
unlimited	860	6	351	5	14.678	68.129
<-3	874	6	379	5	14.103	70.907
<-2	852	6	429	5	14.635	68.329
<-1	808	3	429	3	17.541	57.009



Processor Modeling vs. HLS

- High-Level Synthesis (C-to-Hardware):
 - Lexical Analysis
 - Algorithm Optimizations → Architectural Optimizations
 - Control and Data Dependencies Analysis
 - Technology Library Processing → Platform Description File
 - Resource Allocation
 - Operation Scheduling
 - Functional Units & Register Binding
→ Processor Architecture Level
 - Output: RTL (Register Transfer Level) Code



Conclusion

- Hardware Design through Processor Architecture abstraction
- Relation of Processor Architecture to HW: platform descriptions of architectural components



References

- [Eck09] W. Ecker, W. Muller, R. Dömer, **Hardware-dependent Software: Principles and Practice**, Springer, 2009.
- [Gaj92] D. D. Gajski, N. Dutt, A. Wu, S. Lin, **High-Level Synthesis – Introduction to Chip and System Design**, Kluwer Academic Publishers, 1992.
- [NISC] **NISC Online Demo**, <http://cecs02.cecs.uci.edu/nisc/demo.v9>
- [CECS] Center for Embedded Computer Systems,
<http://cecs.uci.edu>



Questions?
