On Formal Verification of Pipelined Processors with Arrays of Reconfigurable Functional Units

Miroslav N. Velev and Ping Gao

ARIES DESIGN AUTOMATION®

Advantages of Reconfigurable DSPs

Increased performance

Reduced power consumption

Adaptability to future applications that are yet unknown

Outline

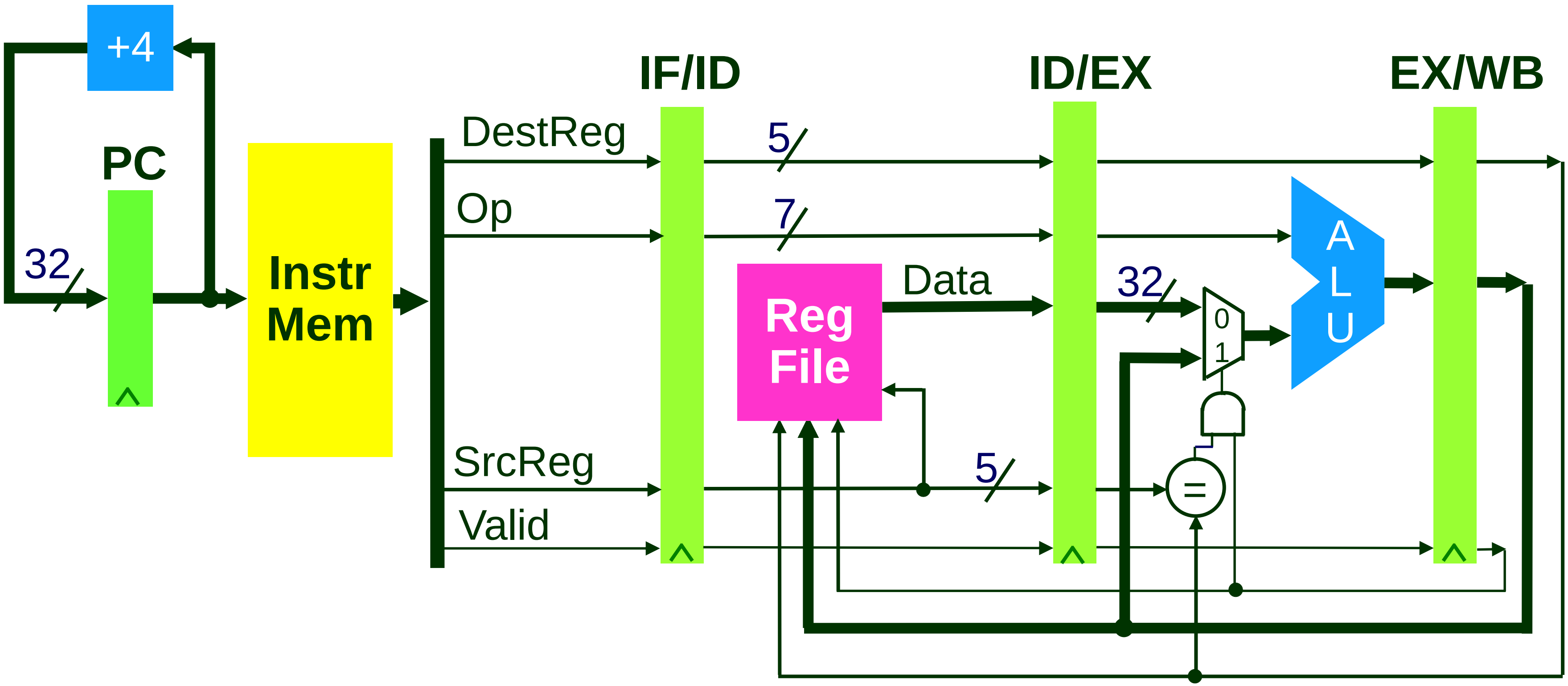
Background: Positive Equality & ADRES

Abstracting Arrays of Reconfigurable Functional Units

Results

Conclusions

Gate-Level Microprocessor



- **Data: vectors of wires**
- **ALUs and memories: gates**

Formal verification complexity is exponential

- **Velev & Bryant [*FMCAD '98*]**

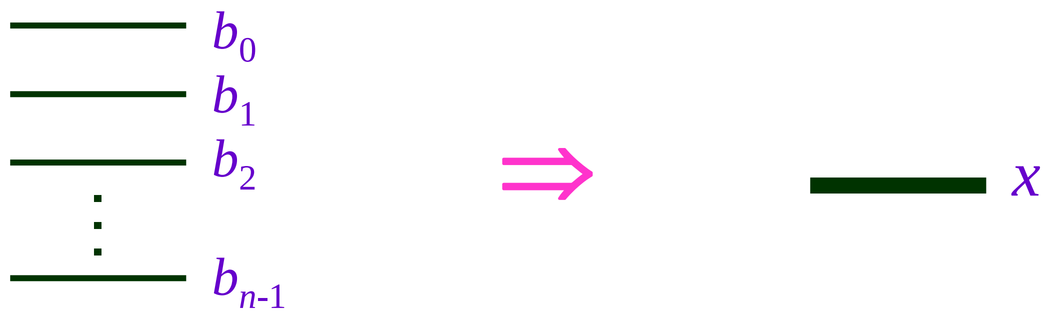
Two-Step Formal Methodology

1) **Formally verify the Functional Units (FUs) and Memories in isolation from the rest of the design**

2) **Formally verify the pipelined/superscalar/VLIW processor after abstracting the FUs and memories, but keeping the fully implemented control logic, data flow, placement of FUs and memories in pipeline stages**
   - **using our tool, HighCheck**
   - **applying suitable modeling techniques**

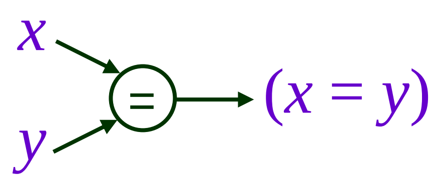
Abstracting Data

Terms abstract data values

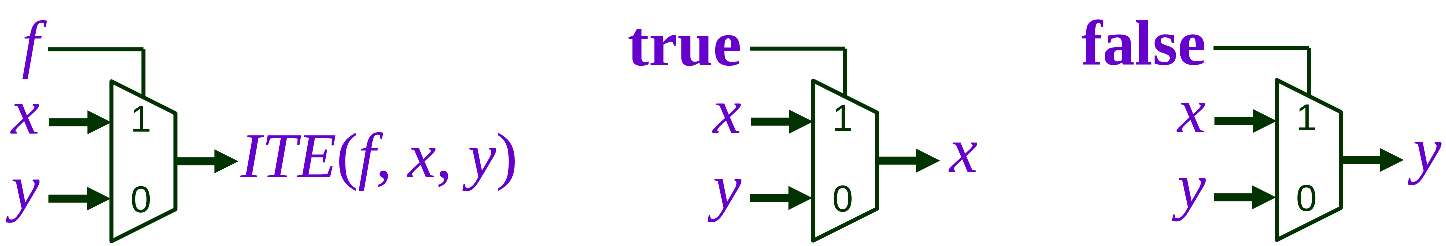


Properties:

- Equality comparison:



- Can be stored in memories
- Can be selected with *ITE* operators:

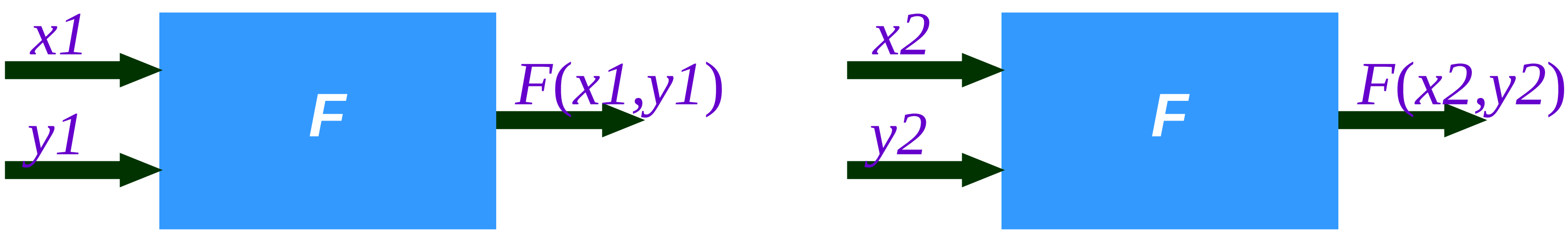


Abstracting ALUs

Uninterpreted Functions abstract computations

- internal implementation details removed

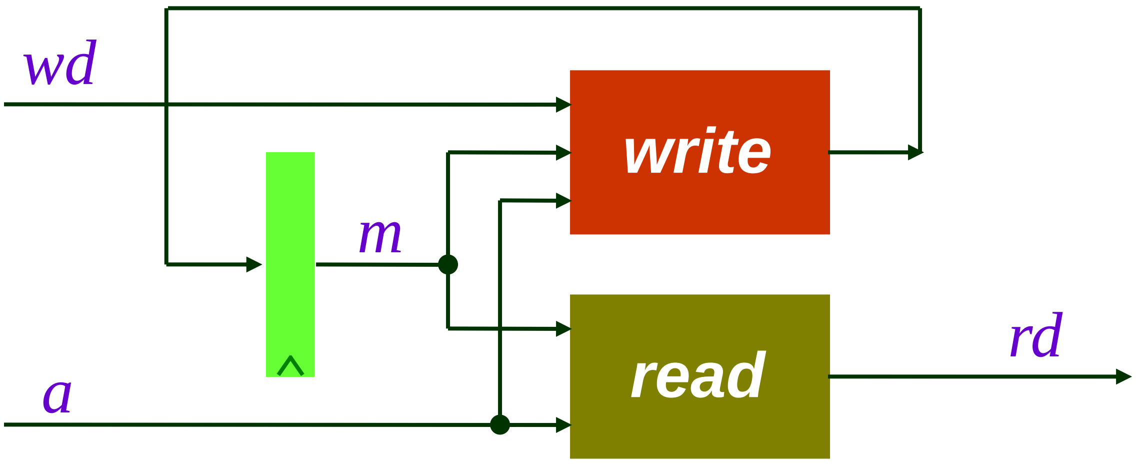
- functional consistency:



$$(x1 = x2) \wedge (y1 = y2) \Rightarrow F(x1,y1) = F(x2,y2)$$

Abstracting Memories

FSM model:

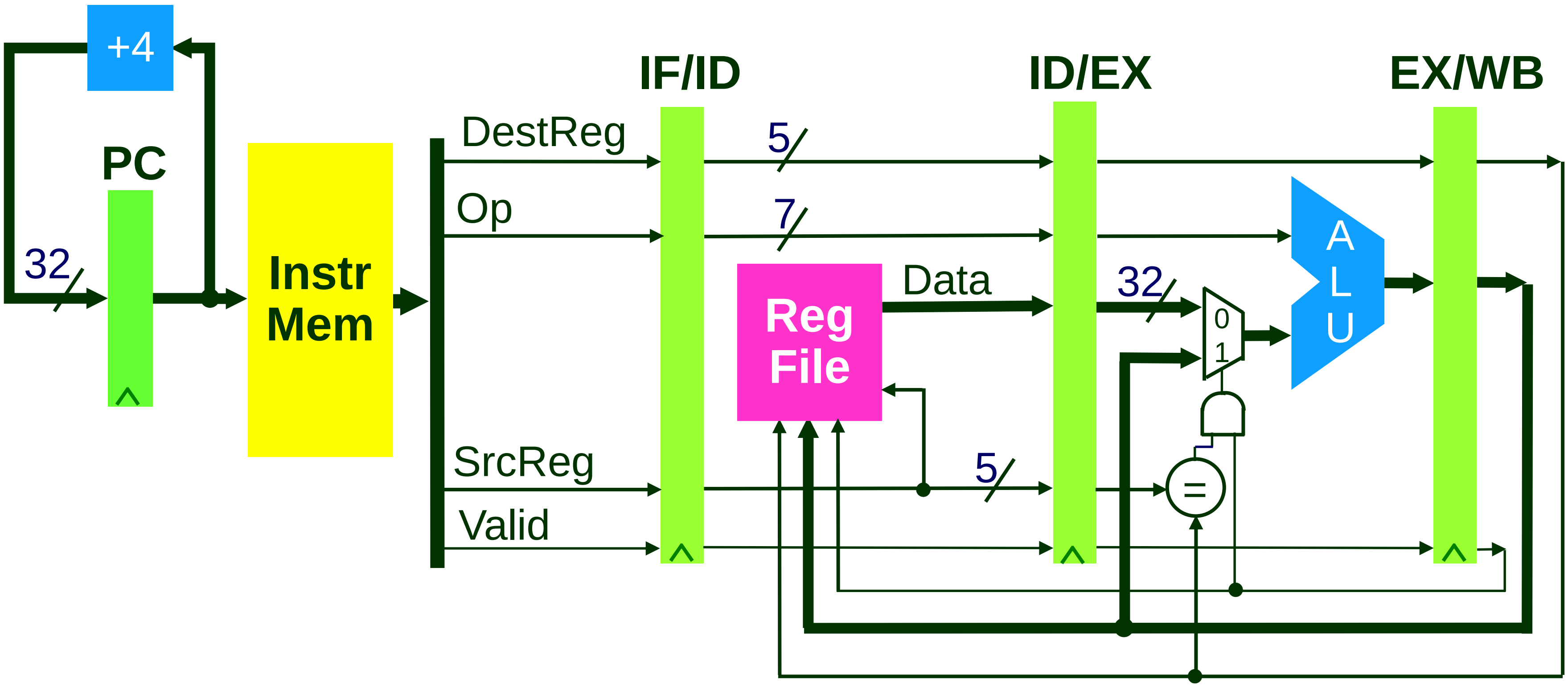


Functions *write* and *read* abstract memory operations

Forwarding property:

$$\mathbf{read}(\mathbf{write}(m_1, a_1, wd), a_2) \;=\; ITE(a_2 = a_1,\; wd,\; \mathbf{read}(m_1, a_2))$$

Gate-Level Microprocessor

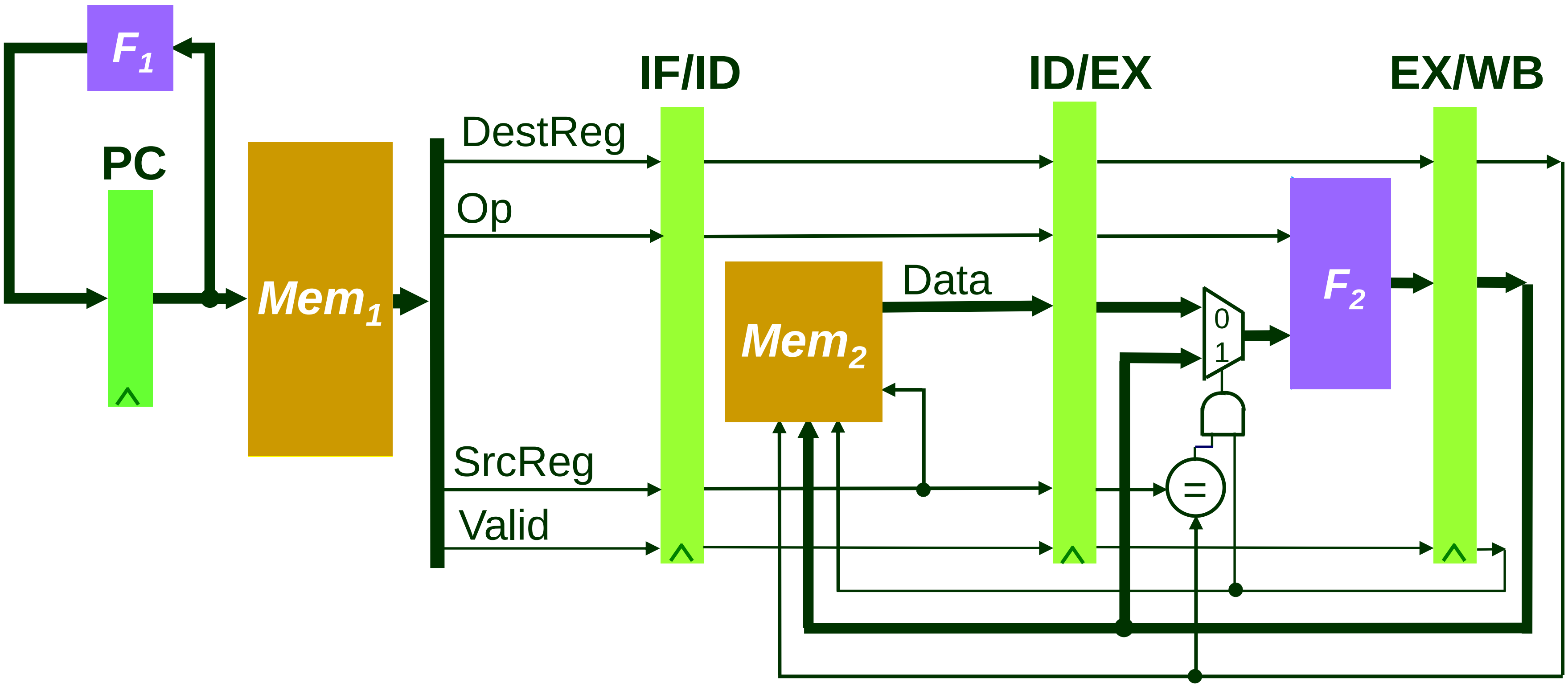


- Data: vectors of wires
- ALUs and memories: gates

Formal verification complexity is exponential

- Velev & Bryant [*FMCAD ’98*]

Application of Abstractions

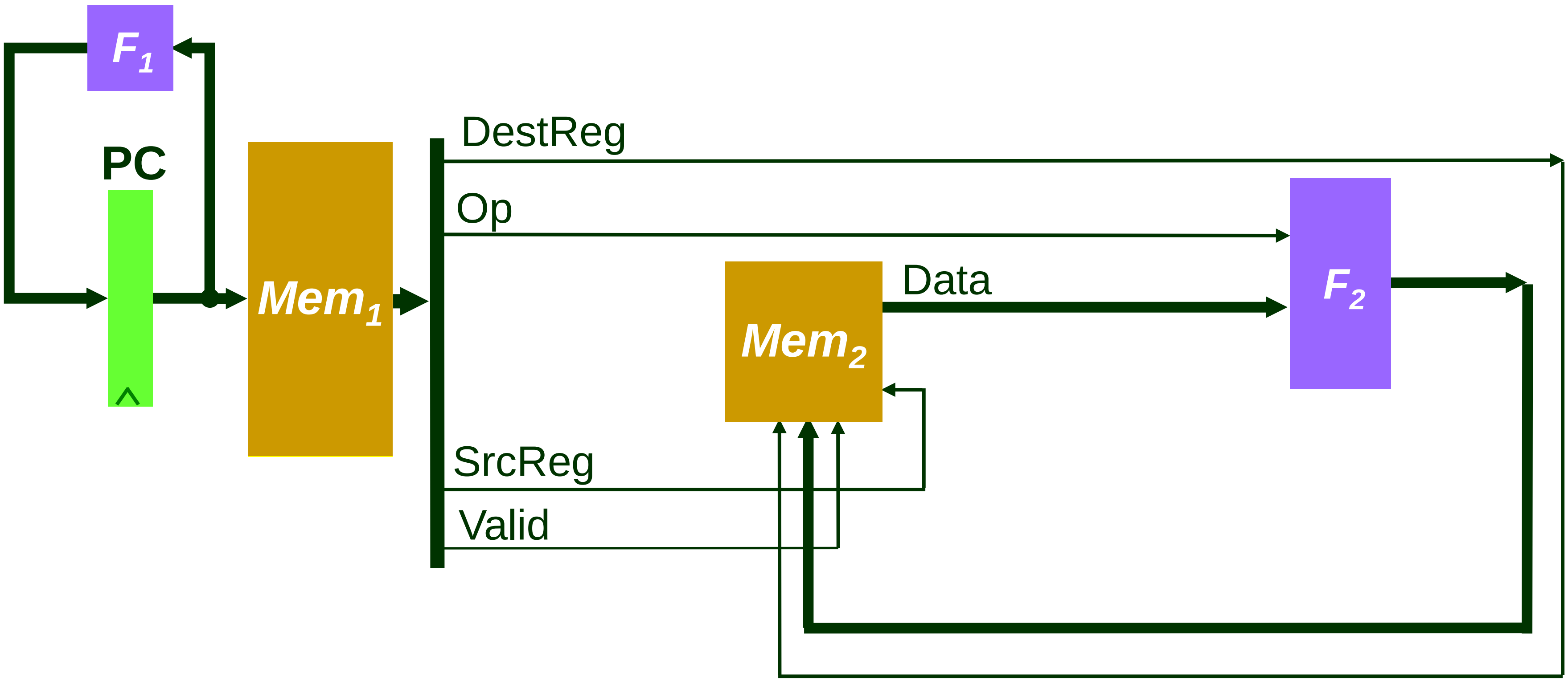


$\Rightarrow$ **More general processor**

- **easier to prove correct**

Functional units & memories formally verified separately

Specification Processor



- single-cycle execution
- only user-visible state
- much simpler control logic

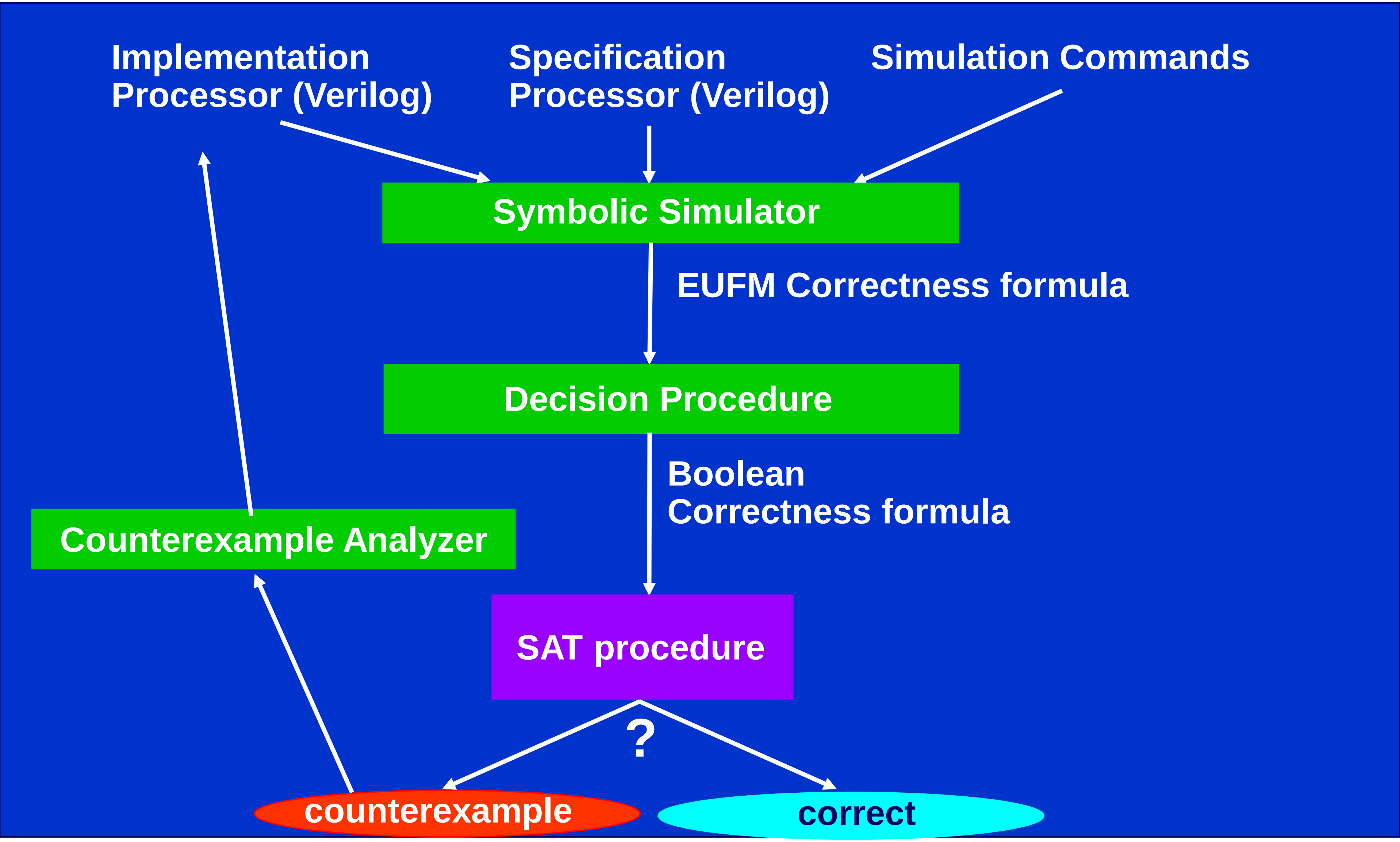
Safety Correctness Criterion

$$\begin{tikzcd} Q^0_{spec} \arrow[rr, "F^k_{spec}"] & & Q^1_{spec} \\ Q^0_{impl} \arrow[u, Rightarrow, "Flush"] \arrow[rr, "F_{impl}"] & & Q^1_{impl} \arrow[u, Rightarrow, "Flush"] \end{tikzcd}$$

Term-level symbolic simulation
of Implementation for 1 clock cycle

symbolic initial state
(represents
ANY initial state)

Our Tool: HighCheck



Positive Equality

By imposing some simple restrictions on the processor modeling style, we obtain a special structure of the correctness formula, where:

P-terms are compared only in positive equations

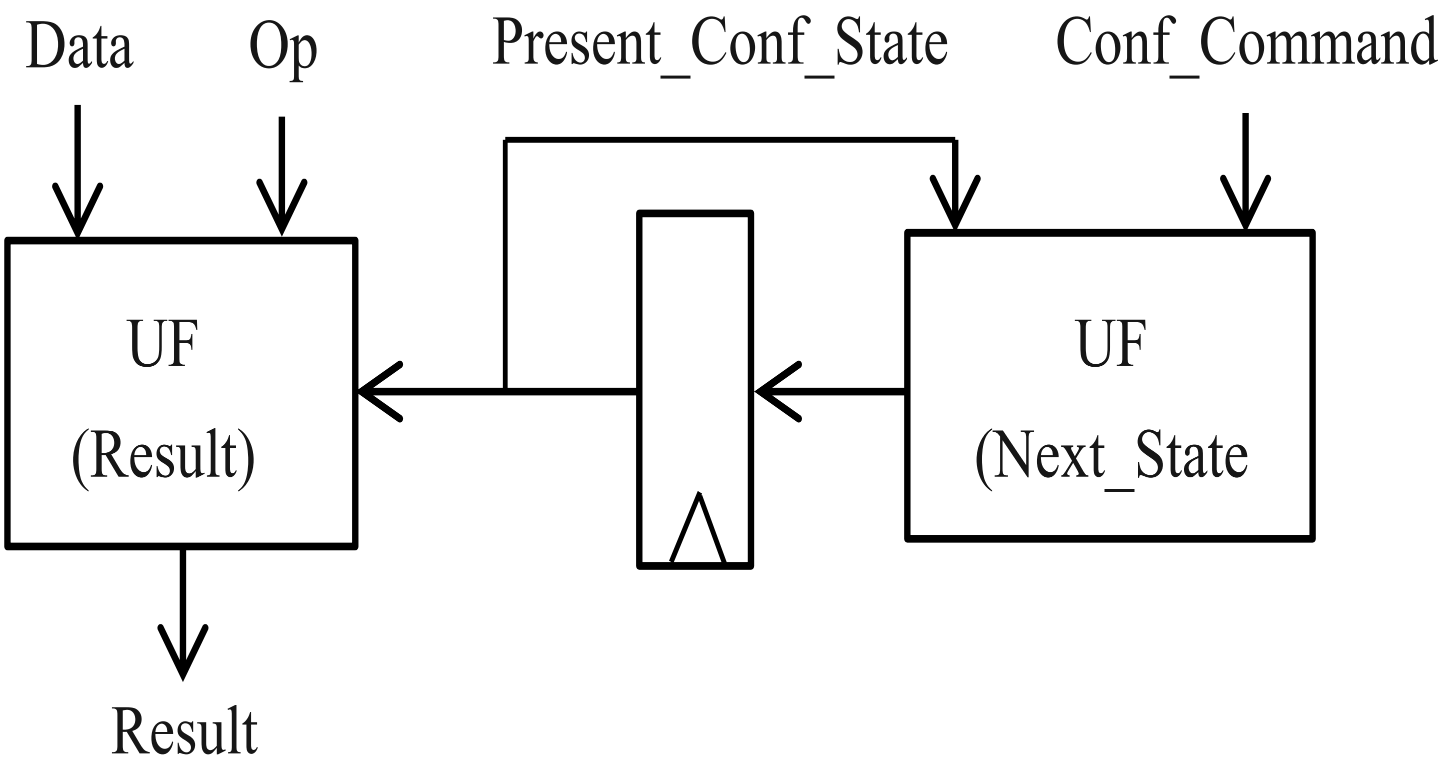
- Connected only with monotonically positive operators AND, OR

G-terms are compared in both positive and negated equations

As a result of the restrictions, most of the terms become p-terms and can be treated as DISTINCT CONSTANTS

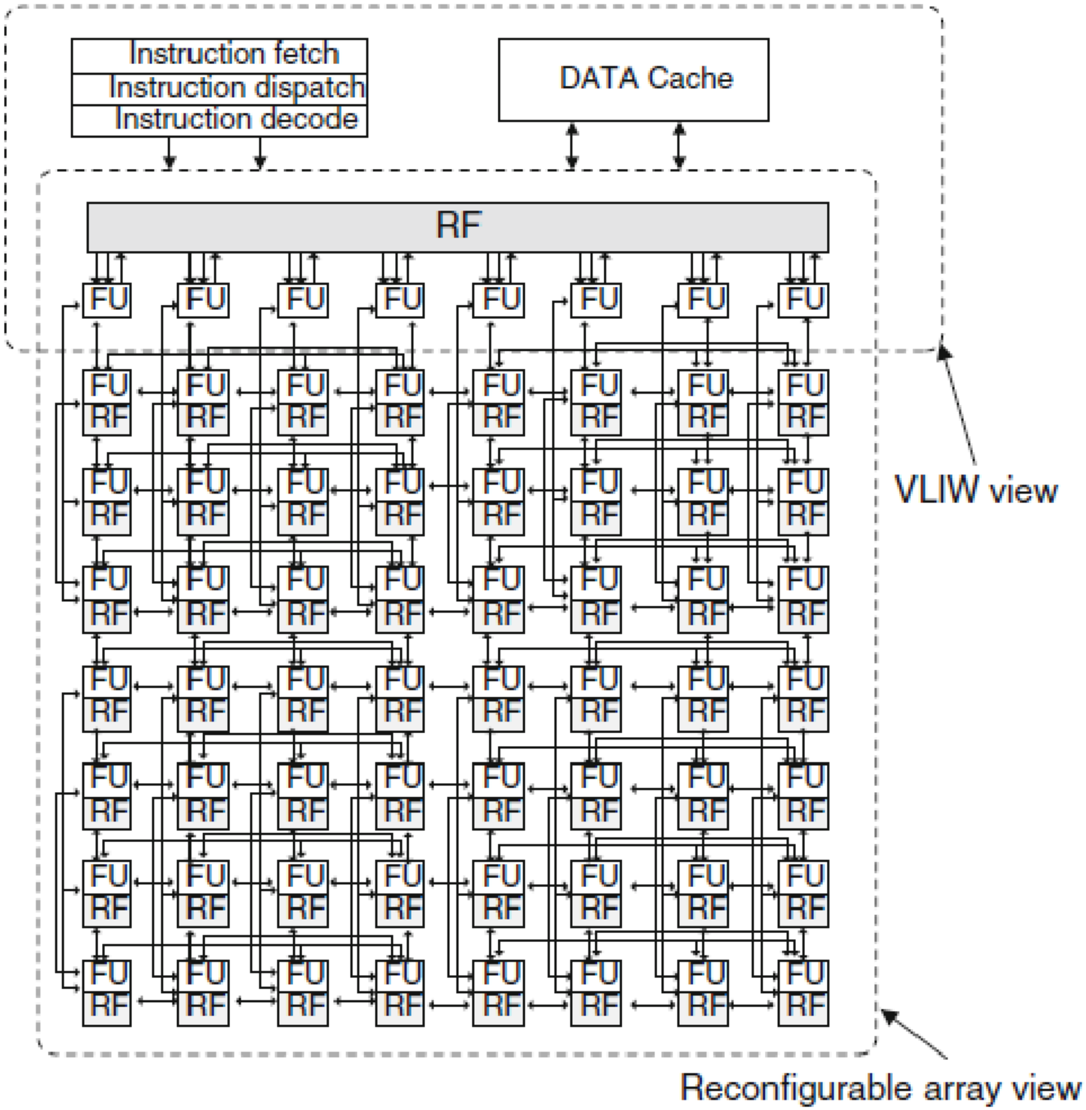
G-terms are assigned small domains of values that have to be indexed with fresh Boolean variables

Abstracting a Single Reconfigurable Functional Unit



[Velev & Gao, ASP-DAC'11]

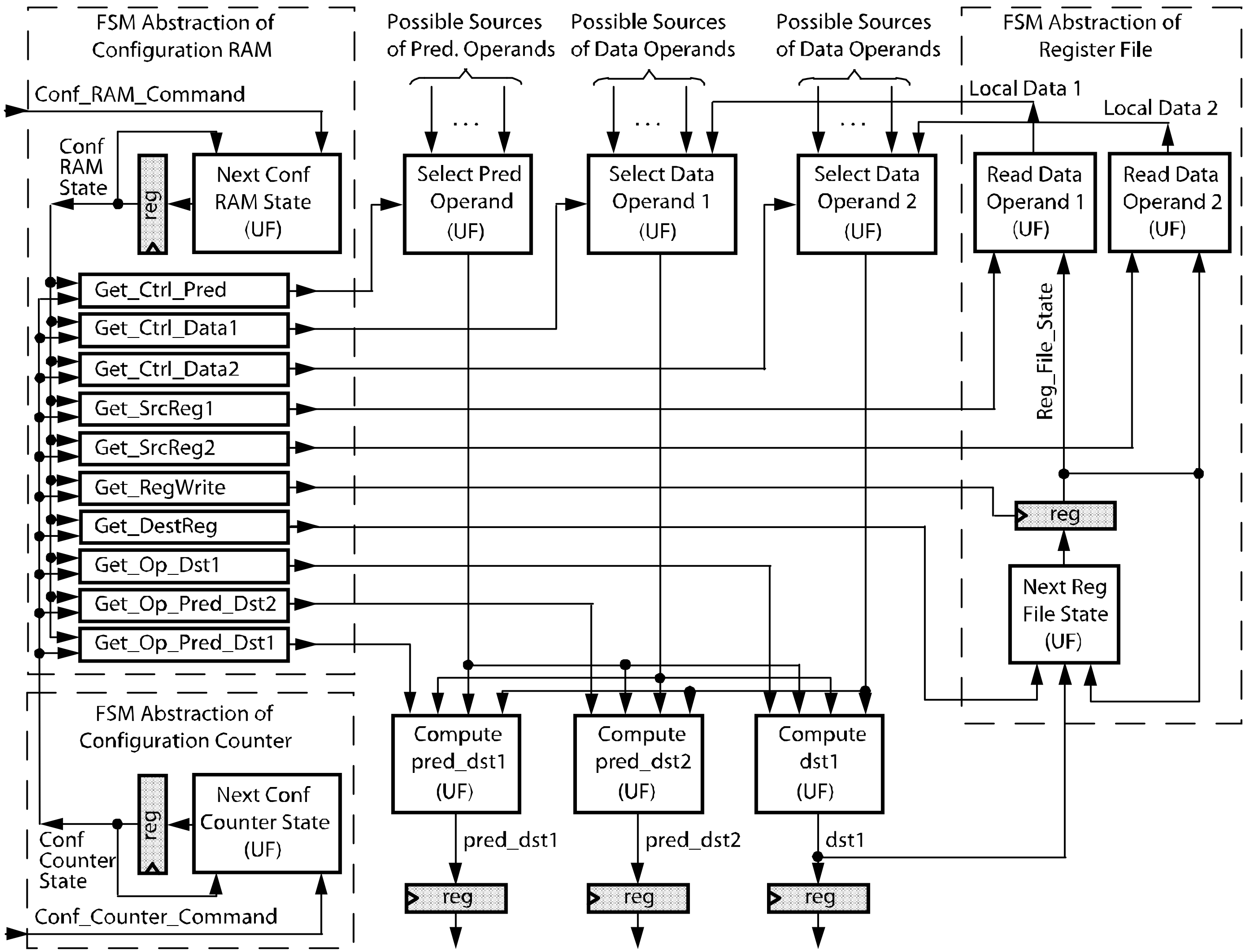
ADRES Reconfigurable Architecture



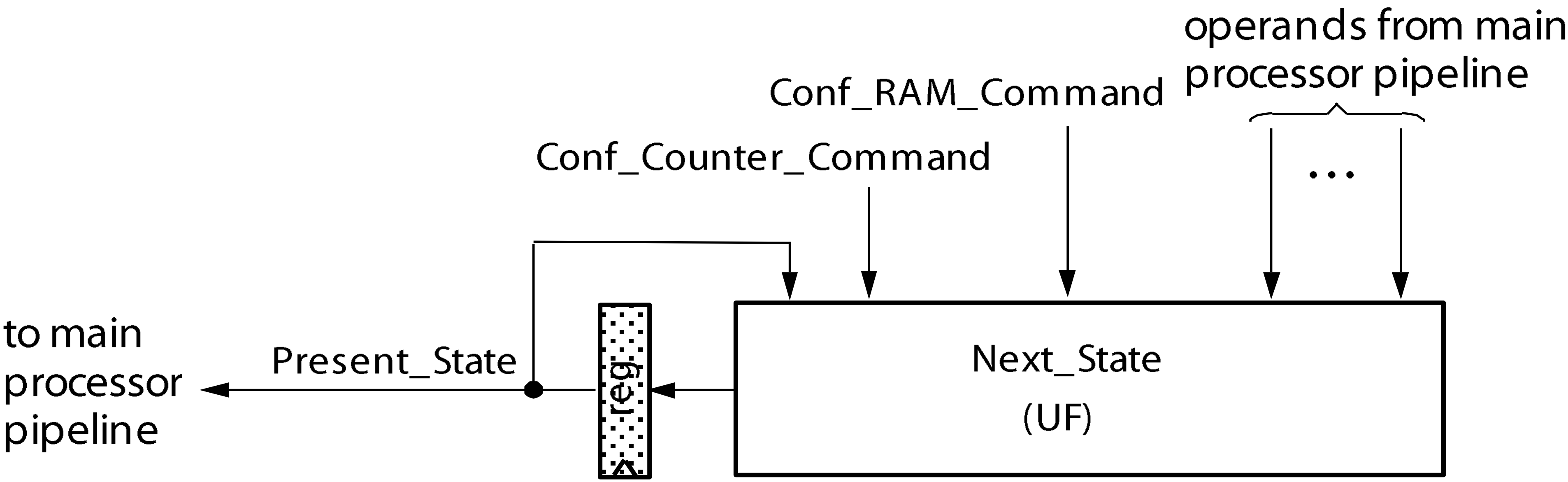
Outline

- ✓ Background: Positive Equality & ADRES
- Abstracting Arrays of Reconfigurable Functional Units
- Results
- Conclusions

Detailed Abstract Model of One Reconfig. Functional Unit in ADRES



Abstracting the Entire Array of Reconfigurable FUs in ADRES



Outline

- ✓ Background: Positive Equality & ADRES
- ✓ Abstracting Arrays of Reconfigurable Functional Units
- Results
- Conclusions

Reconfigurable DSP for NASA

- each VLIW instruction consists of 3 RISC instructions from the PowerPC 750 Instruction Set Architecture (ISA)
- each RISC instruction is predicated based on a predicate register identifier that points to a 1-bit register location in a Predicate Register File that was added to the PowerPC 750 ISA
- branch prediction
- register remapping
- mechanism to self-detect and self-heal from timing errors [Velev & Gao, ICFEM'10]

Results

Processor	CNF Vars	CNF Clauses	Time [s]
DSP_base	14,540	214,842	4.4
DSP_array_16	15,697	229,970	4.9
DSP_array_32	16,656	244,688	6.8
DSP_array_64	18,625	290,306	9.3
DSP_array_128	22,662	416,415	21
DSP_array_256	30,337	808,130	55
DSP_array_512	46,071	2,192,512	306
DSP_array_1028	77,280	7,316,482	1,790
DSP_array_2048	——	——	>10,236
DSP_array_fsm	14,706	216,120	5.2

Outline

- ✓ Background: Positive Equality & ADRES
- ✓ Abstracting Arrays of Reconfigurable Functional Units
- ✓ Results
- Conclusions

Conclusions

We presented abstraction techniques to formally verify integration of pipelined processors with arrays of reconfigurable FUs in style of ADRES

The arrays of reconfigurable FUs can be of any size, where the reconfigurable FUs have any design, and are connected with a network of any topology

These abstraction techniques result in at least 3 orders of magnitude speedup relative to formal verification without them, and the speedup is increasing with the number of reconfigurable FUs in the array

This is the first work on automatic formal verification of pipelined processors with arrays of reconfigurable functional units