Integrating ACL2 with SMT Solvers

Panagiotis Manolios

College of Computer and Information Science

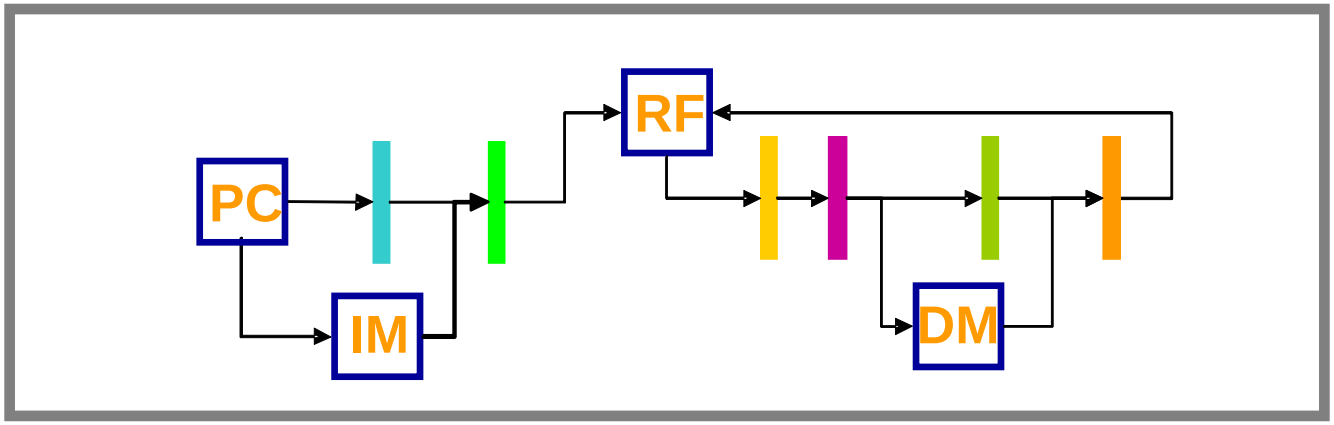
Northeastern University

Sudarshan Srinivasan

Department of Electrical & Computer Engineering

North Dakota State University

Motivation



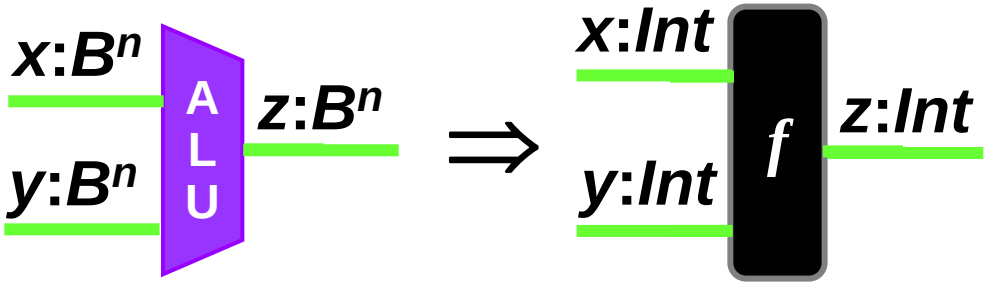
- Pipelined machine verification
- Deductive reasoning (ACL2)
  - Applicable to bit-level designs
  - Prior work: Sawada&Hunt
- Decision Procedures (Yices)

Decision Procedures

- Positives:
  - Highly automatic
  - Generates counterexamples

Decision Procedures

- Positives:
  - Highly automatic
  - Generates counterexamples
- Drawbacks:
  - Applicable only to term-level models
  - Term-level models not executable
  - Hard to analyze counter examples



Decision Procedures

- Positives:
  - Highly automatic
  - Generates counterexamples
- Drawbacks:
  - Applicable only to term-level models
  - Term-level models not executable
  - Hard to analyze counter examples
- Examples: EUF, UCLID
- State-of-the-art: SMT Solvers (Yices, Barcelogic)

Verification Approach

- Use ACL2 to reduce Bit-level verification problem to a term-level problem
- Hand off term-level problem to an SMT solver (Yices)
  - Reason about pipeline at the term-level
- Requires seamless integration of an SMT solver with ACL2: **ACL2-SMT**
- Result: We can verify executable machines with bit-level interfaces with a high degree of automation and efficiency

Outline

- ACL2-SMT Integration Strategy
- Translation Mechanism
- Application: Pipelined Machine Verification
- Conclusions and Future Work

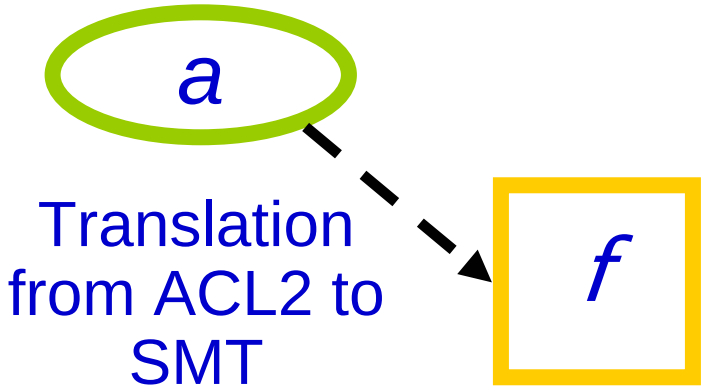
Outline

- ACL2-SMT Integration Strategy
- Translation Mechanism
- Application: Pipelined Machine Verification
- Conclusions and Future Work

Integration Strategy

a : ACL2 clause

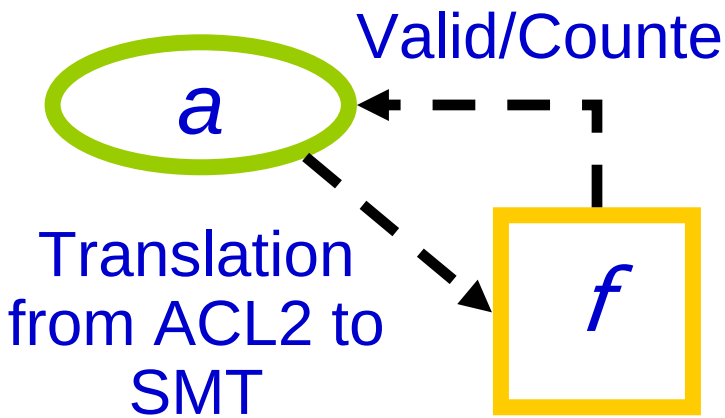
Integration Strategy



a : ACL2 clause

f : SMT formula

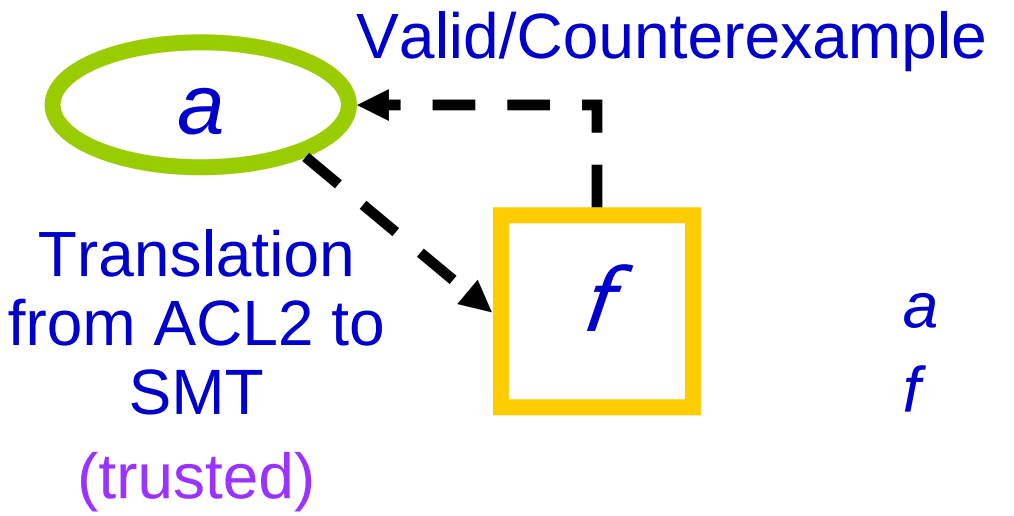
Integration Strategy



a : ACL2 clause

f : SMT formula

Integration Strategy



a : ACL2 clause

f : SMT formula

- Decidable logic
  - Uninterpreted functions, arrays, linear integer arithmetic
- ACL2 subset: *ALU*

Outline

- ACL2-SMT Integration Strategy
- Translation Mechanism
- Application: Pipelined Machine Verification
- Conclusions and Future Work

ISA Example

```
(defun step-isa (isa)
  (let ((pc (g 'pc isa))
        (rf (g 'rf isa))
        (imem (g 'imem isa)))
    (let ((inst (g pc imem)))
      (let ((arg1 (select (src1 inst) rf))
            (arg2 (select (src2 inst) rf)))
        (let ((result (alu arg1 arg2)))
          (let ((isa-new (seq nil
                              'pc (pcadd pc)
                              'rf (nextsrf inst rf result)
                              'imem imem)))
            isa-new))))))
```

Uninterpreted Functions

```
(encapsulate ((alu (x y) t))
  (local (defun alu (x y)
           (declare (ignore x)
                    (ignore y))
           1))
  (defthm alu-type
    (implies (and (integerp a)
                  (integerp b))
             (integerp (alu a b c)))))
```

Property

```
(defthm isa-pc
  (implies
   (and
    (integerp (g 'pc isa))
    (integer-arrayp (g 'rf isa))
    (integer-arrayp (g 'imem isa)))
   (equal
    (g 'pc (step-isa isa))
    (pcadd (g 'pc isa))))
  :hints (("Goal"
           :clause-processor
           (smt-clause-processor clause nil state))))
```

Top-level type hypothesis

Property

Function Expansion

```
(if (equal
   (g 'pc
      ((lambda (pc rf imem)
         ((lambda (inst imem pc rf)
            ((lambda (arg1 arg2 pc inst rf imem)
               ((lambda (result imem rf inst pc)
                  ((lambda (isa-new) isa-new)
                   (s 'pc (pcadd pc)
                      (s 'rf
                         (store (dest inst) result rf))
                         (s 'imem imem 'nil)))))
                (alu arg1 arg2)
                imem rf inst pc)) . . .]
```

Environment

```
(nil                                    :Boolean variables
 (smt1_isa_pc)                          :Integer variables
 nil                                    :Uninterpreted predicates
 ((pcadd . 1)                           :Uninterpreted functions
  (src2 . 1)
  (dest . 1)
  (src1 . 1)
  (alu . 2))
 (smt1_isa_imem smt1_isa_rf)            :integer array variables
 ((isa ('pc int)                        :record variables
       ('rf int-array)
       ('imem int-array))))
```

Translation Mechanism

```
(if_then_else
 (= (let int (pc smt1_isa_pc)
      (let int-array (rf smt1_isa_rf)
        (let int-array (imem smt1_isa_imem)
          (let int (inst (select imem pc))
            (let int (arg1 (select rf (src1 inst)))
              (let int (arg2 (select rf (src2 inst)))
                (let int (result (alu arg1 arg2))
                  (let (('pc int)
                        ('rf int-array)
                        ('imem int-array))
                    (isa-new (s 'pc
                                (pcadd pc) . . .]
```

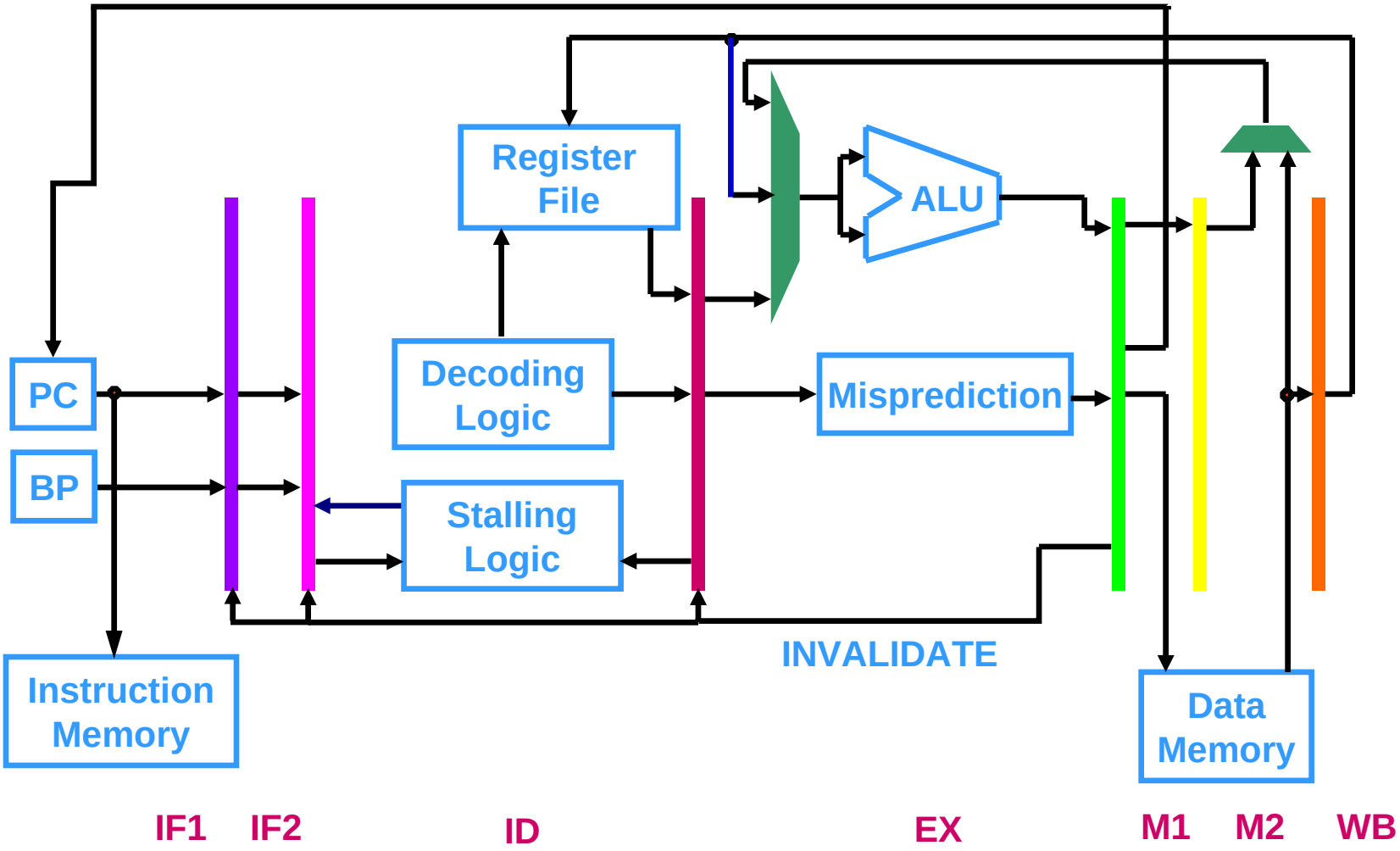
Translation Mechanism

```
(benchmark acl2_smt
 :extrafuns ((smt1_isa_pc Int))
 :extrafuns ((smt1_isa_rf Array)) . . .
 :extrafuns ((alu Int Int Int)) . . .
 :formula
  (if_then_else
   (= (let (pc smt1_isa_pc)
       (let (rf smt1_isa_rf)
        (let (imem smt1_isa_imem)
         (let (inst (select imem pc))
          (let (arg1 (select rf (src1 inst)))
           (let (arg2 (select rf (src2 inst)))
            (let (result (alu arg1 arg2))
             (let (smt1_isa-new_pc (pcadd pc))
                    . . .
      (pcadd smt1_isa_pc))
 false true))
```

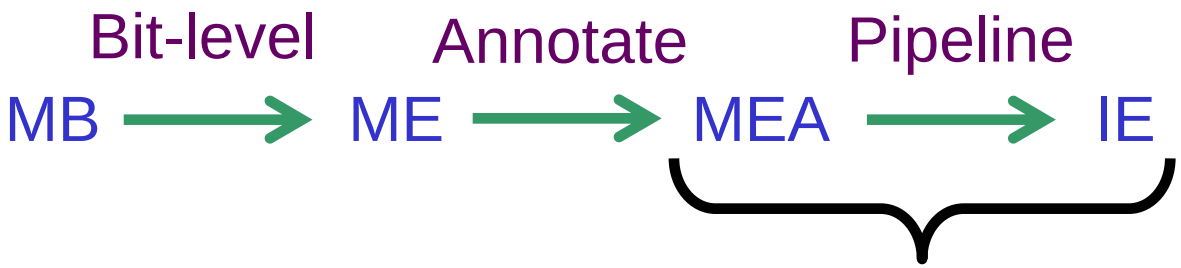
Outline

- ACL2-SMT Integration Strategy
- Translation Mechanism
- Application: Pipelined Machine Verification
- Conclusions and Future Work

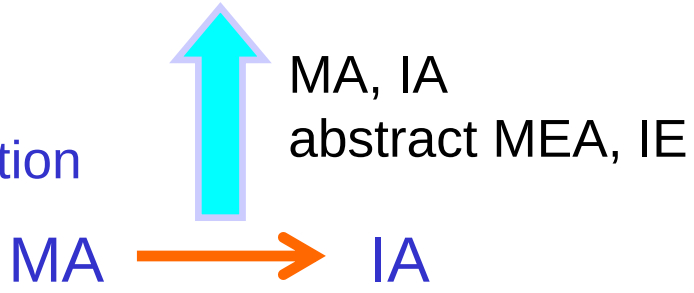
Processor Model

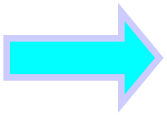


Proof Methodology



MB: Pipeline, bit-level, executable
ME: Pipeline, integer, executable
MEA: ME annotated with history information
IE: ISA version of ME




Functional Instantiation

Verification Statistics

Proof Step	Proof Time (sec)	User Effort (person-days)
MB $\rightarrow$ ME	22.40	7
ME $\rightarrow$ MEA	16.37	1
MA $\rightarrow$ IA	3.47	7
MEA $\rightarrow$ IE	1.61	10
Total	43.85	25

- Effort required for term-level verification using UCLID: 30 days
- We only require about 90% of UCLID effort

Outline

- ACL2-SMT Integration Strategy
- Translation Mechanism
- Application: Pipelined Machine Verification
- Conclusions and Future Work

Conclusions

- Developed ACL2-SMT by combining ACL2 with Yices
- Allows us to relate term-level models with RTL-level designs
- Showed how to verify bit-level pipelined machines in a highly automated and efficient manner
- Future work: Verify Plasma CPU: IP Core used in various applications

Integrating ACL2 with SMT Solvers

Panagiotis Manolios

College of Computer and Information Science

Northeastern University

Sudarshan Srinivasan

Department of Electrical & Computer Engineering

North Dakota State University