Controller Synthesis for Pipelined Circuits Using Uninterpreted Functions

A Glance at Ongoing Research

Georg Hofferek
IAIK – Graz University of Technology
georg.hofferek@iaik.tugraz.at
Reference

- Georg Hofferek and Roderick Bloem

"Controller Synthesis for Pipelined Circuits Using Uninterpreted Functions"

Ninth ACM/IEEE International Conference on Formal Methods and Models for Codesign (MemoCODE 2011)

http://dx.doi.org/10.1109/MEMCOD.2011.5970508
https://online.tugraz.at/tug_online/voe_main2.getvolltext?pCurrPk=57673
Motivation: Pipelined Microprocessor

Non-pipelined processor:

Pipelined processor, using the same combinational datapath elements:
Abstraction by Uninterpreted Functions

- **Datapath**
  - Bit-wise description prohibitively large
    - e.g. multipliers

- Abstraction through uninterpreted functions
  - neither know nor care about “internals”

- **functional consistency**
  - \( (a = b) \implies f(a) = f(b) \)
(Very) Simple Example

Non-pipelined Architecture (=reference):

Pipelined Architecture:
Equivalence for Pipelines

Burch-Dill paradigm:

Non-Pipelined Architecture

Instr. Set Arch. (ISA)

complete

Pipelined Architecture

complete

step

Instruction Set Architecture

Pipelined Architecture
Example: Equivalence Criterion

Non-pipelined Architecture (reference):

- Registers REG
- Read
- ALU
- Write
- Source
- Destination

Pipelined Architecture:

- Registers REG
- Read
- ALU
- Write
- Source
- Destination
- Control

Complete – ISA:

\[ \varphi_{cI} = [REG'_{cI} \leftarrow REG\{w \leftarrow ALU(v)\}] \land [REG''_{cI} \leftarrow REG''_{cI}\{d \leftarrow ALU(REG'_{cI}[s])\}] \]

Step – Complete:

Analogous

Equivalence criterion:

\[ \varphi_{AUE} := (\varphi_{cI} \land \varphi_{sc}) \rightarrow (REG''_{cI} = REG''_{sc}) \]
Synthesis Approach

- Define equivalence criterion: \( \varphi_{AUE} \)

- Claim: \( \forall \overline{R} . \forall \overline{f} . \forall i, s . \exists \overline{c} . \forall \overline{R}' . \forall \overline{s}' . \varphi_{AUE} \)

- If the claim is valid, extract \( \overline{c} := \overline{c}(\overline{R}, \overline{f}, \overline{i}, \overline{s}) \)
Reductions

- (closed) second-order formula, using theories of
  - Arrays (A)
  - Uninterpreted Functions (U)
  - Equality (E)

- with limited quantification.

- Three (validity-preserving) reductions:
  - AUE $\rightarrow$ UE (via index set)
  - UE $\rightarrow$ E (Ackermann’s reduction)
  - E $\rightarrow$ Propositional Logic (graph-based reduction)
Proof Structure

- Construct $\phi'$ from $\phi$ (standard procedures)

- Show that if $\forall a \cdot \exists b \cdot \forall c \cdot \phi$ is valid, then also $\forall x \cdot \exists y \cdot \forall z \cdot \phi'$ is valid.

1. Choose $\bar{x}$ arbitrarily
2. Map $\bar{x}$ to $\bar{a}$ according to $\alpha$
3. Find some $\bar{b}$ by using validity of $\phi$
4. Map $\bar{b}$ to $\bar{y}$ according to $\beta$
5. Choose $\bar{z}$ arbitrarily
6. Map $\bar{z}$ to $\bar{c}$ according to $\gamma$
7. Show that $\phi[\bar{a}/\bar{a}, \bar{b}/\bar{b}, \bar{c}/\bar{c}]$ implies $\phi'[\bar{x}/\bar{x}, \bar{y}/\bar{y}, \bar{z}/\bar{z}]$ (by using structural similarities)
Extract Function for Control Logic

- We started from: \( \forall \bar{R} \cdot \forall \bar{f} \cdot \forall \bar{i}, \bar{s} \cdot \exists \bar{c} \cdot \forall \bar{R}' \cdot \forall \bar{s}' \cdot \varphi \)

- Apply transformations, obtain \( \varphi_{prop} \)

- Universally quantify “next states” \( \bar{s}', \bar{R}' \)
  - i.e., quantify all variables which “come from” one of the next state variables. E.g.: \( E_{REG''}, REG' \)

- Expand existential quantification of \( \bar{c} \)
  - Example: \( \exists \bar{c} \cdot \varphi_{prop} \iff (c \land \varphi_{prop}) \lor (\neg c \land \varphi_{prop}) \)

- Find cofactors of \( \bar{c} \)
  - Positive Cofactor: ON-Set + DC-Set
  - Negative Cofactor: OFF-Set + DC-Set

- Find function in this interval
Summary of Synthesis algorithm

- We started from
  - a **datapath** of the target system
  - a **reference implementation**
  - an **equivalence criterion**

- We obtained
  - Boolean function(s) for the control logic
  - in terms of
    - (dis-)equalities between inputs and states
    - Example: \( c := e_{s,w} \) or \( c := (s = w) \)
Experimental Results

- **Equivalence Criterion for Simple Example:**
  - Manually reduced from AUE to UE and from UE to E
  - Semi-manually reduced from E to propositional logic

- **BDD-based computation of c:**
  - ~14 hours for simple example
    - Most time for reordering during creation of transitivity constraints
  - ~10 minutes with variable order determined by the 14 hour run
  - Resulting interval had two non-trivial boundaries and contained the expected result
Ongoing Work

- **Complete Liveness**
  - Deal with stalling

- **Quantitative Aspects**
  - e.g., minimize amount of „stall“ signals

- **Interpolation**
  - Directly in E or UE, using SMT solver or theorem prover
  - Avoid costly reductions to propositional logic
  - Multiple signals in one step

- **Prototype Tool**