A Value Propagation Based Equivalence Checking Method for Verification of Code Motion Techniques

Kunal Banerjee, Chandan Karfa*, Dipankar Sarkar, Chittaranjan Mandal

Department of Computer Sc & Engg, Indian Institute of Technology Kharagpur
Synopsys (India) Pvt Ltd, Bangalore*

ISED 2012
BESU, Kolkata, India
Outline

1. Motivation
2. Background
3. Code Motions Across Loops
4. The Method of Value Propagation
5. The Overall Verification Method
6. Experimental Results
7. Conclusion and Future Work
Motivation

A typical synthesis flow of complex systems like VLSI circuits or embedded systems comprises several phases. Each phase transforms/refines the input behavioural specification (of the systems to be designed) with a view to optimizing time and physical resources. Behavioural verification involves demonstrating the equivalence between the input behaviour and the final design which is the output of the last phase.

Objective

Validation of transformations of the input behaviours taking place at various phases of synthesis of embedded systems.
Finite State Machines with Datapaths (FSMDs)

FSMDs effectively capture both the control flow and the associated data processing of a behaviour.

The FSMD model is a seven tuple $F = \langle Q, q_0, I, V, O, f, h \rangle$:

- $Q$: Finite set of control states
- $q_0$: Reset state, i.e. $q_0 \in Q$
- $I$: Set of primary input signals
- $V$: Set of storage variables
- $O$: Set of primary output signals
- $f$: State transition function, i.e. $Q \times 2^S \rightarrow Q$
- $h$: Update function of the output and the storage variables, i.e. $Q \times 2^S \rightarrow U$

- $U$ represents a set of storage or output assignments
- $S$ is a set of arithmetic relations between arithmetic expressions
Equivalence checking of FSMDs

An FSMD $M_0$ is said to be contained in another FSMD $M_1$, represented by $M_0 \sqsubseteq M_1$, if for every computation $\mu_0$ of $M_0$ there is an equivalent computation $\mu_1$ of $M_0$.

Two FSMDs are equivalent, $M_0 \equiv M_1$, if $M_0 \sqsubseteq M_1$ and $M_1 \sqsubseteq M_0$.

Length and number of computations of an FSMD can both be infinite.
Identification of suitable cutpoints leads to a finite path cover.

The reset state and all states with multiple incoming/outgoing transitions can be considered as the cutpoints.

Since any computation corresponds to a concatenation of paths, it is enough to establish path equivalences.

Code transformations can make this job difficult.

Equivalence checking of FSMDs

\[ \begin{align*}
q_0,0 & \xleftarrow{\neg a \iff b + c} q_0,1 \\
q_0,1 & \xleftarrow{\neg d \iff a - e} q_0,2 \\
q_0,2 & \xleftarrow{x < y} q_0,3 \\
q_0,3 & \xleftarrow{x \iff x + y, \neg x \iff x} q_0,4 \\
q_0,4 & \xleftarrow{\neg t \iff x + f} q_0,5 \\
q_0,5 & \xleftarrow{\neg \neg} q_0,6 \\
q_0,6 & \xleftarrow{\neg \neg m \iff t - d} q_0,7 \\
q_0,7 & \xleftarrow{\neg h \iff r + m} q_0,8 \\
q_0,8 & \xrightarrow{\neg \neg} (a) M_0
\end{align*} \]

\[ \begin{align*}
q_1,0 & \xleftarrow{\neg a \iff b + c} q_1,1 \\
q_1,1 & \xleftarrow{\neg d \iff a - e} q_1,2 \\
q_1,2 & \xleftarrow{x < y} q_1,3 \\
q_1,3 & \xleftarrow{x \iff x + y, \neg x \iff x} q_1,4 \\
q_1,4 & \xleftarrow{\neg t \iff x + f} q_1,5 \\
q_1,5 & \xleftarrow{\neg \neg} q_1,6 \\
q_1,6 & \xleftarrow{\neg \neg m \iff t - d} q_1,7 \\
q_1,7 & \xleftarrow{\neg \neg h \iff r - d} q_1,8 \\
q_1,8 & \xrightarrow{\neg \neg} (b) M_1
\end{align*} \]
Equivalence checking of FSMDs

- Identification of suitable cutpoints leads to a finite path cover
- The reset state and all states with multiple incoming/outgoing transitions can be considered as the cutpoints
- Since any computation corresponds to a concatenation of paths, it is enough to establish path equivalences
- Code transformations can make this job difficult
Code motions across loops

- \( t \leftarrow a + b \)  

[Tristan et al, PLDI 2009] – assumes that there exists an injective function from the nodes of the original code to the nodes of the transformed code.
Code motions across loops

[Tristan et al, PLDI 2009] – assumes that there exists an injective function from the nodes of the original code to the nodes of the transformed code.
The method of value propagation

\[ q_0, s, \{ \ldots, v, \ldots \} \]

\[ v \leftarrow f(x) \]

\[ q_0, t, \{ \ldots, f(x), \ldots \} \]

\[ (a) M_0 \]

\[ q_1, s, \{ \ldots, v, \ldots \} \]

\[ v \leftarrow g(y) \]

\[ q_1, t, \{ \ldots, g(y), \ldots \} \]

\[ (b) M_1 \]

An example of value propagation
The method of value propagation

An example of value propagation with dependency between propagated values
The method of value propagation

\[ q_0,a \leftarrow \langle \ldots, v_i, \ldots, v_j, \ldots \rangle \]
\[ -/v_i \leftarrow f(v_n, v_j) \]
\[ \beta \]
\[ q_0,b \]
\[ -/v_j \leftarrow h(v_k, v_l) \]
\[ q_0,c \leftarrow \langle \ldots, f(v_n, v_j), \ldots, v_j, \ldots \rangle \]
\[ \beta' \]
\[ c_1/v_i \leftarrow v_i + g(v_m) \]
\[ q_0,z \leftarrow \langle \ldots, g(v_m) + f(v_n, v_j), \ldots, v_j, \ldots \rangle \]

(a) \( M_0 \)

\[ q_1,a \leftarrow \langle \ldots, v_i, \ldots, v_j, \ldots \rangle \]
\[ -/v_i \leftarrow g(v_m) \]
\[ \alpha \]
\[ q_1,b \]
\[ -/v_j \leftarrow h(v_k, v_l) \]
\[ q_1,c \leftarrow \langle \ldots, g(v_m), \ldots, v_j, \ldots \rangle \]
\[ \alpha' \]
\[ c_1/v_i \leftarrow v_i + f(v_n, v_j) \]
\[ q_1,z \leftarrow \langle \ldots, g(v_m) + f(v_n, v_j), \ldots, v_j, \ldots \rangle \]

(b) \( M_1 \)

An erroneous decision taken
The method of value propagation

- \( q_0, a \leftarrow \langle \ldots, v_i, \ldots, v_j, \ldots \rangle \)
  \[-/v_i \leftarrow f(v_n, v_j) \]

- \( q_0, b \leftarrow \langle \ldots, v_i, \ldots, v_j, \ldots \rangle \)
  \[-/v_j \leftarrow h(v_k, v_l) \]

- \( q_0, c \leftarrow \langle \ldots, f(v_n, v_j), \ldots, h(v_k, v_l), \ldots \rangle \)

- \( q_1, a \leftarrow \langle \ldots, v_i, \ldots, v_j, \ldots \rangle \)
  \[-/v_i \leftarrow g(v_m) \]

- \( q_1, b \leftarrow \langle \ldots, v_i, \ldots, v_j, \ldots \rangle \)
  \[-/v_j \leftarrow h(v_k, v_l) \]

- \( q_1, c \leftarrow \langle \ldots, g(v_m), \ldots, h(v_k, v_l), \ldots \rangle \)

- \( q_0, z \leftarrow \langle \ldots, g(v_m) + f(v_n, v_j), \ldots, h(v_k, v_l), \ldots \rangle \)

- \( q_1, z \leftarrow \langle \ldots, g(v_m) + f(v_n, h(v_k, v_l)), \ldots, h(v_k, v_l), \ldots \rangle \)

**Correct** decision taken
Value propagation: Algorithm

Algorithm: valuePropagation ($\beta$, $\langle C^i_\beta, \bar{v}^i_\beta \rangle$, $\alpha$, $\langle C^i_\alpha, \bar{v}^i_\alpha \rangle$)

1. $C^f_\beta \leftarrow C^i_\beta \land R_\beta(\bar{v})\{\bar{v}^i_\beta / \bar{v}\};$  
   $C^f_\alpha \leftarrow C^i_\alpha \land R_\alpha(\bar{v})\{\bar{v}^i_\alpha / \bar{v}\};$

2. $\forall$ variable $v_j \in |V_0 \cup V_1|$

   \[
   \Pi_j(\bar{v}_\beta^f) \leftarrow \begin{cases} 
   s_\beta(\bar{v})\{\bar{v}^i_\beta / \bar{v}\}\big|_{v_j}, & \text{if } s_\beta(\bar{v})\{\bar{v}^i_\beta / \bar{v}\}\big|_{v_j} \neq s_\alpha(\bar{v})\{\bar{v}^i_\alpha / \bar{v}\}\big|_{v_j} \text{ or } \\
   \exists v_k, s_\beta(\bar{v})\{\bar{v}^i_\beta / \bar{v}\}\big|_{v_k} \neq s_\alpha(\bar{v})\{\bar{v}^i_\alpha / \bar{v}\}\big|_{v_k} \quad \land \\
   \bigvee_{\gamma \in \{\beta, \alpha\}} v_j \text{ occurs in } s_\gamma(\bar{v})\{\bar{v}^i_\gamma / \bar{v}\}\big|_{v_k} \\
   v_j, & \text{otherwise}
   \end{cases}
   \]

3. $\forall$ variable $v_j \in |V_0 \cup V_1|$

   \[
   \Pi_j(\bar{v}_\alpha^f) \leftarrow \begin{cases} 
   s_\alpha(\bar{v})\{\bar{v}^i_\alpha / \bar{v}\}\big|_{v_j}, & \text{if } s_\alpha(\bar{v})\{\bar{v}^i_\alpha / \bar{v}\}\big|_{v_j} \neq s_\beta(\bar{v})\{\bar{v}^i_\beta / \bar{v}\}\big|_{v_j} \text{ or } \\
   \exists v_k, s_\alpha(\bar{v})\{\bar{v}^i_\alpha / \bar{v}\}\big|_{v_k} \neq s_\beta(\bar{v})\{\bar{v}^i_\beta / \bar{v}\}\big|_{v_k} \quad \land \\
   \bigvee_{\gamma \in \{\beta, \alpha\}} v_j \text{ occurs in } s_\gamma(\bar{v})\{\bar{v}^i_\gamma / \bar{v}\}\big|_{v_k} \\
   v_j, & \text{otherwise}
   \end{cases}
   \]

4. return $\langle(C^f_\beta, \bar{v}_\beta^f), (C^f_\alpha, \bar{v}_\alpha^f)\rangle$;
Equivalence checking of FSMDs using value propagation

At the reset states
Equivalence checking of FSMDs using value propagation

\[ q_0, a \]
\[ \rightarrow \neg i \leftarrow 1 \]
\[ q_0, b \]
\[ \langle T, \langle x, y, i, N, t_1, t_2, h \rangle \rangle \]
\[ i < N/ \]
\[ x \leftarrow t_1 + t_2 + x \times i, \]
\[ y \leftarrow t_1 - t_2 \]
\[ i \leftarrow i + 1 \]
\[ q_0, c \]

\[ q_1, a \]
\[ \rightarrow \neg i \leftarrow 1, h \leftarrow t_1 + t_2, \]
\[ y \leftarrow t_1 - t_2 \]
\[ q_1, b \]
\[ \langle T, \langle x, t_1 - t_2, i, N, t_1, t_2, \rangle \rangle \]
\[ i < N/ \]
\[ x \leftarrow h + x \times i, \]
\[ i \leftarrow i + 1 \]
\[ q_1, c \]

At the beginning of the loops

(a) \( M_0 \)

(b) \( M_1 \)
Equivalence checking of FSMDs using value propagation

At the end of the loops

(a) $M_0$

- $i \leftarrow 1$
- $i \leftarrow i + 1$
- $x \leftarrow t_1 + t_2 + x \times i$
- $y \leftarrow t_1 - t_2$

(b) $M_1$

- $-i \leftarrow 1$
- $h \leftarrow t_1 + t_2$
- $y \leftarrow t_1 - t_2$
- $i \leftarrow i + 1$
- $x \leftarrow h + x \times i$
- $i \leftarrow i + 1$
- $y \leftarrow t_1 - t_2$

Kunal Banerjee (IIT Kharagpur)
Equivalence checking of FSMDs using value propagation

At the end states
Verification algorithm

Algorithm: equivalenceChecker (FSMD $M_0$, FSMD $M_1$)

1. Introduce cutpoints in the FSMDs $M_0$ and $M_1$, and compute the path covers;
2. Let $\zeta$, the set of corresponding state pairs, be $\langle q_{0,0}, q_{1,0} \rangle$;
3. for (each $\langle q_{0,i}, q_{1,j} \rangle \in \zeta$) {
   4. for (each path $\beta : (q_{0,i} \Rightarrow q_{0,m}) \in P_0$ originating from $q_{0,i}$) {
      5. if (path $\alpha : (q_{1,j} \Rightarrow q_{1,n}) \in P_1$ can be found such that $\beta \simeq c \alpha$)
         $\zeta \leftarrow \zeta \cup \{\langle q_{0,m}, q_{1,n} \rangle\}$;
      6. elsif (path $\alpha : (q_{1,j} \Rightarrow q_{1,n}) \in P_1$ can be found such that $\beta \simeq \alpha$)
         employ value propagation until $\beta \simeq \alpha$ can be established without violating loop invariance;
      7. else
         print “failure”;
   8. }
   9. }
10. print “success”;
### Experimental Results

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>Original FSMD</th>
<th>Transformed FSMD</th>
<th>#Variables</th>
<th>#across loops</th>
<th>Maximum mismatch</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#states</td>
<td>#paths</td>
<td>#states</td>
<td>#paths</td>
<td>com</td>
<td>unconm</td>
</tr>
<tr>
<td>BARCODE</td>
<td>33</td>
<td>54</td>
<td>25</td>
<td>56</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>DCT</td>
<td>16</td>
<td>1</td>
<td>8</td>
<td>1</td>
<td>41</td>
<td>6</td>
</tr>
<tr>
<td>DIFFEQ</td>
<td>15</td>
<td>3</td>
<td>9</td>
<td>3</td>
<td>19</td>
<td>3</td>
</tr>
<tr>
<td>EWF</td>
<td>34</td>
<td>1</td>
<td>26</td>
<td>1</td>
<td>40</td>
<td>1</td>
</tr>
<tr>
<td>GCD</td>
<td>8</td>
<td>11</td>
<td>4</td>
<td>8</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>IEEE754</td>
<td>55</td>
<td>59</td>
<td>44</td>
<td>50</td>
<td>32</td>
<td>3</td>
</tr>
<tr>
<td>LRU</td>
<td>33</td>
<td>39</td>
<td>32</td>
<td>38</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>MODN</td>
<td>8</td>
<td>9</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>PERFECT</td>
<td>6</td>
<td>7</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>QRS</td>
<td>53</td>
<td>35</td>
<td>24</td>
<td>35</td>
<td>25</td>
<td>15</td>
</tr>
<tr>
<td>TLC</td>
<td>13</td>
<td>20</td>
<td>7</td>
<td>16</td>
<td>8</td>
<td>1</td>
</tr>
</tbody>
</table>

Value propagation based equivalence checking transcends path based technique by being able to verify code motions across loops.

Unlike [Tristan et al, PLDI 2009], the presented method does not require additional inputs.

Experimentation involving 2-step translation process exhibits effectiveness of the method.
Future Work

- Complexity analysis
- Formal proof of correctness
- Handle transformations that change the control flow graph of a program
- Combine this technique with the existing framework that handles non-uniform code motion techniques [Karfa et al, TODAES 2012]
- Incorporate array variables, and subsequently address loop transformations
Selected References

Thank You !!!

kunalb@cse.iitkgp.ernet.in