From clock driven to Data-driven models

Y. Bai and K. Schneider and N. Bhardwaj and B. Katti and T. Shazadi

Formal Methods and Models for Co-design

(MEMOCODE’14)
Motivation

- Synchronous systems:
  - bound to a clock – i.e. clock dependent \(\rightarrow\) clock constraint
  - expected to do 0 time computation but NOT
  - they’re time consuming – read - compute – write - takes time \(\rightarrow\) timing constraint

- These two constraints are not good for distributed systems
Outline

• Quartz – the synchronous language

• Synchronous systems
  – Clock and timing constraint
  – Example – Seq ITE

• Desynchronization
  – Asynchronous systems
  – Box values □
  – Example – Seq ITE

• Endochrony
  – Endochronous systems
  – No clock yet deterministic
  – Symbolic representation
  – Proof: Examples – Seq ITE is endo!

• Verification and SAT
Core Idea

- independent of clock yet deterministic
- transformations on the synchronous systems
- verify whether or not it is Endochronous !!
- For verification, we have reduced the problem to SAT!
- Quartz – clocked guarded actions we describe the system such that it is reduced to a boolean expression whose SATisfiability check defines the presence of Endochrony!
Quartz – The programming Language

- asynchronous parallel execution of threads
- explicit implementation of non determinism
- delayed data assignments (next(\(\lambda\)) = T)
- guarded actions
- boolean expression \(\rightarrow\) SAT solvability!

module site(clocked bool ?x1,?x2,?x3,!y) {
  loop{
    if(clk(x1) & clk(x2) & x1) {
      y = (x2,true);
    }
    if(clk(x1) & clk(x3) & !x1) {
      y = (x3,true);
    }
    pause;
  }
}

clocked guarded actions

Seq ITE program in Quartz
Synchronous systems

- Clock driven
  - +ve $\rightarrow$ Deterministic!
  - -ve $\rightarrow$ dependent & inefficient
- Slow clock
- Solution: Elastic systems $\Rightarrow$ the communication wires are replaced by buffers
  - +ve $\rightarrow$ improves worst case execution time
  - -ve $\rightarrow$ still clock driven
Synchronous systems

• Example Seq ITE

<table>
<thead>
<tr>
<th>x₁</th>
<th>x₂</th>
<th>x₃</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1 :: A)</td>
<td>(b :: B)</td>
<td>(c :: C)</td>
<td>[b]</td>
</tr>
<tr>
<td>(0 :: A)</td>
<td>(b :: B)</td>
<td>(c :: C)</td>
<td>[c]</td>
</tr>
</tbody>
</table>

| ξ(x₁) : | 1 0 0 1 1 ... |
| ξ(x₂) : | 1 3 5 7 9 ... |
| ξ(x₃) : | 0 2 4 6 8 ... |
| ξ(y)  : | 1 2 4 7 9 ... |
Introducing

- In Seq ITE there are inputs that consume values even when they’re not needed → waste of energy and time!
- Replace these values by □
- □ are the values which are not required for computation but for alignment of input streams
- System is still working on a clock!
Introducing

- Seq ITE – aligned input stream BUT with

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$b$</td>
<td>$\Box$</td>
<td>$c$</td>
</tr>
<tr>
<td>0</td>
<td>$b$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Column $\xi(x_1)$:

- 1 0 0 1 1 ...

Column $\xi(x_2)$:

- 1 $\Box$ $\Box$ 7 9 ...

Column $\xi(x_3)$:

- $\Box$ 2 4 $\Box$ $\Box$ ...

Column $\xi(y)$:

- 1 2 4 7 9 ...

Input $x_1$:

0 0 1

Input $x_2$:

- $\Box$ $\Box$ 1

Input $x_3$:

- 4 2 $\Box$

Output $Y$:

.. 4 2 1
Desynchronization

- Finally, we remove
- Streams are desynchronized
- Nodes now have to *resynchronize* their inputs

**Synchronous Systems**
- Clock driven
- Deterministic

**Replacing unwanted inputs with**
- Clock driven
- Inputs aligned with

**Asynchronous Systems**
- No clock
- No
- Data driven
Desynchronization

- Seq ITE – no more

<table>
<thead>
<tr>
<th>x₁</th>
<th>x₂</th>
<th>x₃</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td></td>
<td>[b]</td>
</tr>
<tr>
<td>0</td>
<td>A</td>
<td></td>
<td>[c]</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\xi(x_1) : & \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad \ldots \\
\xi(x_2) : & \quad 1 \quad 7 \quad 9 \quad \ldots \\
\xi(x_3) : & \quad 2 \quad 4 \quad \ldots \\
\xi(y) : & \quad 1 \quad 2 \quad 4 \quad 7 \quad 9 \quad \ldots 
\end{align*}
\]

1 1 0 0 1

9 7 1

8 6 4 2

Input x₁ → 0 0 1

Input x₂ → 1

Input x₃ → 4 2

Output Y → ..4 2 1
Endochronous systems

• If the asynchronous system obtained after desynchronization is able to:
  – Resynchronize its input stream
  – Generate same output as parent – synchronous – system
  – Exhibit constructiveness
  – Maintains determinism

→ System possesses endochrony!
Endochronous systems

- But not all systems are endochronous!
- Example – Par ITE

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(1 :: A)$</td>
<td>$(b :: B)$</td>
<td>$(c :: C)$</td>
<td>$[b]$</td>
</tr>
<tr>
<td>$(0 :: A)$</td>
<td>$(b :: B)$</td>
<td>$(c :: C)$</td>
<td>$[c]$</td>
</tr>
<tr>
<td>$(a :: A)$</td>
<td>$(b :: B)$</td>
<td>$(b :: C)$</td>
<td>$[b]$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(1 :: A)$</td>
<td>$(b :: B)$</td>
<td>$(\square :: C)$</td>
<td>$[b]$</td>
</tr>
<tr>
<td>$(0 :: A)$</td>
<td>$(\square :: B)$</td>
<td>$(c :: C)$</td>
<td>$[c]$</td>
</tr>
<tr>
<td>$(\square :: A)$</td>
<td>$(b :: B)$</td>
<td>$(b :: C)$</td>
<td>$[b]$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(1 :: A)$</td>
<td>$(b :: B)$</td>
<td>$C$</td>
<td>$[b]$</td>
</tr>
<tr>
<td>$(0 :: A)$</td>
<td>$B$</td>
<td>$(c :: C)$</td>
<td>$[c]$</td>
</tr>
<tr>
<td>$A$</td>
<td>$(b :: B)$</td>
<td>$(b :: C)$</td>
<td>$[b]$</td>
</tr>
</tbody>
</table>
Endochronous systems

For input stream – 0 0 1 1 1 0...

Output stream – 1 2 1 2 4 3 ...

≠

Output stream – 1 2 3 2 4 6 5 ...

<table>
<thead>
<tr>
<th>$\varrho(x_1)$</th>
<th>0 0 1 1 1 0 ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varrho(x_2)$</td>
<td>1 2 4 6 8 10 ...</td>
</tr>
<tr>
<td>$\varrho(x_3)$</td>
<td>1 2 3 5 7 9 ...</td>
</tr>
<tr>
<td>$\varrho(y)$</td>
<td>1 2 1 2 4 3 ...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\varrho(x_1)$</th>
<th>0 0 1 1 1 0 ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varrho(x_2)$</td>
<td>1 2 4 6 8 10 ...</td>
</tr>
<tr>
<td>$\varrho(x_3)$</td>
<td>1 2 3 5 7 9 ...</td>
</tr>
<tr>
<td>$\varrho(y)$</td>
<td>1 2 3 2 4 6 5 ...</td>
</tr>
</tbody>
</table>

- Not *Latency-insensitive*
- Not even a function
- Not deterministic
Endochronous systems

- **Synchronous Par ITE**
  - Clock driven
  - Deterministic

- **Par ITE with boxes in input stream**
  - Clock driven
  - Inputs aligned with box

- **Asynchronous Par ITE**
  - No clock
  - No boxes
  - Data driven

- **Non-Endochronous Par ITE**
  - No clock - data driven
  - No boxes
  - Non-Deterministic

→ **Not all systems** are Endochronous
→ **Some need clock** for deterministic output!
• **Step 1:** Describing synchronous systems

Synchronous System

\[\begin{align*}
(V, G) & \\
(V_{in} \cup V_{loc} \cup V_{out}) & \\
(\gamma, \alpha) & \\
next (x) = \tau
\end{align*}\]

• For every \(x\), clock \(\text{clk}(x)\) - such that for all points of time \(\text{clk}(x) = \text{true}\) iff \(x \neq \square\)

\[\text{if}(\text{clk}(x_1) \& \text{clk}(x_2) \& x_1) \quad \Rightarrow \quad \text{clk}(x_1) \& \text{clk}(x_2) \& x_1 \quad \Rightarrow \quad y = (x_2, \text{true})\]
Verification

- Transition relation of the system:

\[ R :\iff \left( \bigwedge_{x \in V_{\text{loc}} \cup V_{\text{out}}} (T_x \land C_x) \land \bigvee_{x \in V_{\text{loc}} \cup V_{\text{out}}} B_x \right) \]
\[ R_{\parallel} :\iff (R \land \text{next}(R)). \]

- **Step 2:** Achieving stateless components with \( V_{\text{loc}} = \{\} \)
  - \( \text{next}(x) = \tau \iff x' = \tau \)
Verification

• **Step 3:**  
  - Clock is a *schedule instruction*
  - No computation is lost
  - Transition relation of the system:

    \[ R_{\text{buf}} \iff R_{||} \land BC \]
    \[ BC \iff \forall x \in V_{in}. (\neg \text{clk}(x) \rightarrow \text{next}(x) = x) \]

• **Step 4:** Data Driven components
  - Transition relation of the system:

    \[ R_{\text{DPN}} \iff \exists \text{clk}(x), \text{clk}(x'). R_{\text{buf}} \iff \exists \text{clk}(x), \text{clk}(x'). R_{||} \land BC \]
Verification

• Final *stateless component* with $R_{buf}$:

\[
\forall s_1, s_2 \in R_{buf}. \forall x \in V_{in}. \\
\quad s_1(x) = s_2(x) \rightarrow s_1(\text{clk}(x)) = s_2(\text{clk}(x))
\]

– Quantify over states
– SAT checks the validity – generates number of clauses!
– BDD based solver – records the size of the BDDs!
– If BDD = true node, system is **Endochronous!**
Thank you!