Combining Process Algebras and Petri Nets for the Specification and Synthesis of Asynchronous Circuits

Marco A. Peña and Jordi Cortadella

Department of Computer Architecture
Universitat Politècnica de Catalunya
Barcelona, Spain
Motivation

inline void def_color(const int i){
    if (i==0) color=white;
    else if (i<3) color=red;
    else if (i<7) color=blue;
    else color=black;
}

car1.def_color(6);
car2.def_color(2);

Syntax-directed translation

    if (6==0) car1.color=white;
    else if (6<3) car1.color=red;
    else if (6<7) car1.color=blue;
    else car1.color=black;

    if (2==0) car1.color=white;
    else if (2<3) car1.color=red;
    else if (2<7) car1.color=blue;
    else car1.color=black;

Optimizing compiler

car1.color=blue;
car2.color=red;
Motivation

High-level language
(C++, Pascal, Lisp,...)  
<table>
<thead>
<tr>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intermediate language</td>
</tr>
<tr>
<td>Code generation and optimization</td>
</tr>
<tr>
<td>Assembly language</td>
</tr>
<tr>
<td>Assembler</td>
</tr>
<tr>
<td>Binary code</td>
</tr>
</tbody>
</table>

Process algebras
(CSP, Tangram, Occam,...)  
| Syntax-directed translation |
| Network of synchronizing elements (Handshake circuit) |
| Event-based translation and optimization |
| STG, Burst-mode automata,... |
| Logic synthesis |
| Gate netlist |
Outline

- Synthesis framework
- Petri net composition
- A complete synthesis example
- Experimental results
- Conclusions
A framework for specification and synthesis

CSP  Tangram  Occam

Network of synchronizing elements

Syntax–directed translation

PN composition

CSC + optimizations

Optimized and synthesizable STG

Logic synthesis

Circuit
How powerful can the method be?

```
begin
  [R~; sel U~; skip |
  L~; R~]
  #[
  R~;
  sel U~; skip |
  L~; R~]
end
```
How powerful can the method be?
How powerful can the method be?
How powerful can the method be?

petrify
(ICCAD’95, ASYNC’96)
How powerful can the method be?

![Diagram with labels Aa, Ra, Ab, Rb, DME, ME, and arrows connecting states.]

ASYNC’96 – Aizu-Wakamatsu, Japan
Petri net composition — Assumptions

• Petri nets correctly composable by syntax-directed construction

Wrong compositions:
Petri net composition — Composition theorem

**Process** $P = (\alpha P, \beta P)$:
- $\alpha P$ is an alphabet of events
- $\beta P \subseteq \alpha P^*$ is a set of traces

**Parallel composition** of $P$ and $Q$ ($P \parallel Q$):
- $\alpha(P \parallel Q) = \alpha P \cup \alpha Q$
- $\beta(P \parallel Q) = \{s \in \alpha(P \parallel Q)^* \mid (s[\alpha P] \in \beta P \land (s[\alpha Q] \in \beta Q)\}$

**Composition theorem.**
Let $N_1$ and $N_2$ be two labeled Petri nets. Then:

$$P(N_1 \parallel N_2) = P(N_1) \parallel P(N_2)$$
A complete synthesis example

```
(A:? bool act & B:! bool act)
beg
    block
        x: var bool
        action
            #[ A?x ; B!x ]
    fblock
end
```
A complete synthesis example
A complete synthesis example
A complete synthesis example
## Experimental results

<table>
<thead>
<tr>
<th>Example</th>
<th>SDT HSKs</th>
<th>Area</th>
<th>final STG States</th>
<th>Area</th>
<th>Red. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUF1</td>
<td>5</td>
<td>402</td>
<td>18</td>
<td>192</td>
<td>52</td>
</tr>
<tr>
<td>RIP_BUF</td>
<td>12</td>
<td>1076</td>
<td>9+247</td>
<td>786</td>
<td>27</td>
</tr>
<tr>
<td>WAG_BUF</td>
<td>16</td>
<td>1992</td>
<td>16+564+112</td>
<td>1906</td>
<td>4</td>
</tr>
<tr>
<td>SRC</td>
<td>8</td>
<td>914</td>
<td>43+19080</td>
<td>768</td>
<td>16</td>
</tr>
<tr>
<td>SRD</td>
<td>11</td>
<td>1414</td>
<td>95+4460+612</td>
<td>1328</td>
<td>6</td>
</tr>
<tr>
<td>3-FIFO</td>
<td>14</td>
<td>1112</td>
<td>53</td>
<td>870</td>
<td>22</td>
</tr>
<tr>
<td>DME</td>
<td>13</td>
<td>550</td>
<td>52+392+332</td>
<td>500</td>
<td>10</td>
</tr>
<tr>
<td>XYZ</td>
<td>12</td>
<td>1686</td>
<td>8</td>
<td>72</td>
<td>95</td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td>9146</td>
<td></td>
<td>6422</td>
<td>30</td>
</tr>
</tbody>
</table>
Conclusions

- Use Process Algebras for specification and event/state-based models (STGs, Burst-mode automata, FSMs) for synthesis

- Synthesis framework: Syntax-directed translation + PN composition + Logic synthesis

- Linear-cost PN composition

- Other approaches: Utah, IMEC, ...