Abstract Clock-based Approaches in the Programming, Design and Analysis of Embedded Systems

Abdoulaye GAMATIÉ
LIRMM/CNRS, Montpellier
abdoulaye.gamatie@lirmm.fr

Multi-Processor System-on-Chip (MPSoC)

- Parallel and distributed implementations of applications
- Execution correctness, performance an energy-efficiency

Embedded applications

- smart & data-intensive: high #functions, data amounts
- real-time: time-bounded reactivity w.r.t. environment

Rational for high-level design approach

Fast, easy, costless and relevant design for multi-clock and data-intensive applications with high-level concepts
Outline

Polychrony and Signal programming

Clock Constraint Specification Language (CCSL)

Abstract clock-based design analysis for MPSoCs

Multi-clocked systems

Globally Asynchronous Locally Synchronous

Polychronous design

Non monolithic vision of system design
- No need to design a priori a global common clock: different activation clocks
- Partially ordered events
- Component-oriented (incremental) design
- Non determinism: beyond the system, its environment
Polychronous formalisms

Signal (around 1981)
- IRISA/Inria Rennes (France)
- Polychrony
  (http://www.irisa.fr/espresso/Polychrony)

Clock Constraint Specification Language – CCSL
(around 2006)
- I3S/Inria Sophia-Antipolis (France)
- TimeSquare (http://timesquare.inria.fr)

Multi-Rate Instantaneous Channel connected Data Flow – MRICDF (around 2008)
- Virginia Tech (Blacksburg, USA)
- EmCodeSyn (http://www.fermat.ece.vt.edu)

PROGRAMMING CONCEPTS OF SIGNAL
Polychrony at a glance

- A formal design model for multi-clocked systems such as GALS [Le Guernic et al.’03]: polychronous\(^1\) model
- Logical time, as in other synchronous languages
- Signal language
- Polychrony (academic) environment (http://www.irisa.fr/espresso/Polychrony)
  - GUI for modeling/specifying/programming in SIGNAL
  - Compiler
  - Connection to the Sigali tool
  - RT-Builder (commercial) environment of the Geensoft spin-off of Dassault Syst. (http://www.geensoft.com)

\(^1\)From the Greek "poly chronos", meaning multiple clocks.

The SIGNAL language

Basic notions
- signal \(x\): infinite series of typed values \((x_t)_{t \in \mathbb{N}}\)

\[
\begin{array}{ccccccc}
  t_0 & t_1 & t_2 & t_3 & t_4 & t_5 & \ldots \\
  x : & 1 & 5 & \bot & 6 & \bot & 0 & \ldots \\
\end{array}
\]

- absence of events: \(\bot\)
- constant signals: series of identical values
- usual data types: boolean, integer, real, complex, char, string, etc.
- pure events: event (sub-type of boolean)

The SIGNAL language (cont’d)

Basic notions
- signal \(x\): infinite series of typed values \((x_t)_{t \in \mathbb{N}}\)

\[
\begin{array}{ccccccc}
  t_0 & t_1 & t_2 & t_3 & t_4 & t_5 & \ldots \\
  x : & 1 & 5 & \bot & 6 & \bot & 0 & \ldots \\
\end{array}
\]

- Abstract clock of a signal: \(\sim x\) (instants of presence)
### The Signal language (cont’d)

#### Basic notions

- **signal** $x$: infinite series of typed values $(x_t)_{t \in \mathbb{N}}$

<table>
<thead>
<tr>
<th>$t_0$</th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
<th>$t_5$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
</tbody>
</table>

- Abstract clock of a signal: $\sim x$ (instants of presence)
- Synchronous signals: $x \sim y$ (same clock)

#### Exercise...

Which scenarios denote synchronous signals?

- **Instantaneous functions/relations**: $z := x + y$

  $(\forall t \in \mathbb{N}) z_t = \begin{cases} \bot & \text{if } x_t = y_t = \bot \\ x_t + y_t & \text{else} \end{cases}$

  $x$: $\bot$ 5 $\bot$ 4 2 0 $\bot$ ...
  $y$: $\bot$ 3 $\bot$ 1 9 8 $\bot$ ...
  $z$: $\bot$ 8 $\bot$ 5 11 8 $\bot$ ...

- **Basic operators on signals**

<table>
<thead>
<tr>
<th>$t_0$</th>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
<th>$t_5$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
</tbody>
</table>

  $Abstract clock$ of a signal: $\sim x$ (instants of presence)

  $Synchronous$ signals: $x \sim y$ (same clock)

  $Process$: system of equations expressing functional and temporal relations between signals.
Basic operators on signals

- **Instantaneous functions/relations:** \( z := x + y \)

\[
(\forall t \in \mathbb{N}) \ z_t = \begin{cases} 
\bot & \text{if } x_t = y_t = \bot \\
 x_t + y_t & \text{else}
\end{cases}
\]

\[
\begin{array}{cccccccccc}
x: & \bot & 5 & \bot & \bot & 4 & 2 & 0 & \bot & \ldots \\
y: & \bot & 3 & \bot & \bot & 1 & 9 & 8 & \bot & \ldots \\
z: & \bot & 8 & \bot & \bot & 5 & 11 & 8 & \bot & \ldots \\
\end{array}
\]

- **Delay:** \( y := x \) $\uparrow$ 1 init 3.14

\[
\begin{array}{cccccccccc}
x_t = \bot & \iff & y_t = \bot \\
(\exists t; \in \mathbb{N}) \ x_t \neq \bot & \Rightarrow & y_{t_0} = 3.14, \ (\forall t; > 0) \ y_{t+1} = x_t
\end{array}
\]

\[
\begin{array}{cccccccccc}
x: & 1.7 & \bot & 2.5 & \bot & \bot & 6.5 & 2.4 & \ldots \\
y: & 3.14 & \bot & 1.7 & \bot & \bot & 2.5 & 6.5 & \ldots \\
\end{array}
\]

- **Undersampling:** \( y := \ x \) when \( b \)

- Extracts elements from \( (x_t)_{t \in \mathbb{N}} \) at instants where \( b \) is true

\[
\begin{array}{cccccccccc}
x: & 5 & \bot & 4 & 8 & 7 & 3 & \bot & \ldots \\
b: & tt & ff & \bot & ff & tt & \bot & tt & \ldots \\
y: & 5 & \bot & \bot & 7 & \bot & \bot & \ldots \\
\end{array}
\]

- **Merging:** \( y := u \) default \( v \)

- Deterministic functional merging of series \( (u_t)_{t \in \mathbb{N}} \) and \( (v_t)_{t \in \mathbb{N}} \)

\[
\begin{array}{cccccccccc}
u: & 5 & \bot & 4 & 8 & \bot & \bot & 3 & \ldots \\
v: & 1 & 7 & \bot & 2 & \bot & 0 & 1 & \ldots \\
y: & 5 & 7 & 4 & 8 & \bot & 0 & 3 & \ldots \\
\end{array}
\]

- **Exercises...**

Discuss the correctness of the following statements:

1. A signal is necessarily present whenever it holds a value different from \( \bot \).
2. A constant signal is always present.
3. When a signal becomes absent, it implicitly keeps its previously carried value.
4. A signal of event type and a signal of boolean type are exactly the same.
5. The abstract clock of a signal defines the set of instants at which the signal occurs.
6. `Signal` assumes a reference clock that enables to always decide the presence/absence of any defined signal.
Exercises...

1. In the expression $s_n := R(s_1, \ldots, s_{n-1})$ where $R$ is an instantaneous relation, if the signal $s_1$ is absent while all other arguments of $R$ are present, $s_n$ is calculated by considering some default value depending on the type of $s_1$.

2. In the expression $s_2 := s_1 \ \text{init} \ c$, the signal $s_2$ may occur with the latest value of $s_1$ while $s_1$ is absent.

3. In the expression $s_3 := s_1 \ \text{default} \ s_2$, the signals $s_1$ and $s_2$ must have exclusive clocks.

4. In the expression $s_3 := s_1 \ \text{or} \ s_2$, $s_3$ is true when $s_1$ is true and $s_2$ is absent; $s_3$ is true when $s_1$ is true and $s_2$ is false; $s_3$ is false when $s_1$ is absent and $s_2$ is false; $s_3$ is absent when $s_1$ is absent and $s_2$ is absent.

Basic operators on processes

- Composition: $u := x + y \mid z := u \ \text{when} \ b$
  
  - Union of systems of equations (static single assignment)
  
  - $P$ and $Q$ communicate via their common signals

<table>
<thead>
<tr>
<th>$x$</th>
<th>$5$</th>
<th>$4$</th>
<th>$2$</th>
<th>$0$</th>
<th>$\ldots$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y$</td>
<td>$3$</td>
<td>$1$</td>
<td>$9$</td>
<td>$9$</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>$u$</td>
<td>$8$</td>
<td>$5$</td>
<td>$11$</td>
<td>$9$</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>$b$</td>
<td>$tt$</td>
<td>$ff$</td>
<td>$tt$</td>
<td>$tt$</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>$z$</td>
<td>$5$</td>
<td>$9$</td>
<td>$\ldots$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Hiding (local declaration): $P \ \text{where} \ u$
  
  - Restricts the visibility scope of signal $u$ to $P$

Syntactic elements: program/process

```plaintext
process <IDENTIFIER> =
  %<SOME_COMMENTS>%
  {<STATIC_PARAMETERS>;}
  (? <INPUT_PARAMETERS>;
  ! <OUTPUT_PARAMETERS>; )
  (| <EQUATION_1|
  | ...|
  | <EQUATION_K>|
  | {<EQUATION_K1>|
  | ...|
  | <EQUATION_Kp>|
  |})
  | <EQUATION_N>|
  | ...|
  |})
where
  <LOCAL_DECLARATIONS>;
end;
```
Example: a counter modulo N

process COUNTER_N =
{integer N;} (? event tick;
! integer cnt; )
(| cnt ^= tick
| cnt := (0 when reset) default cnt_pre + 1
| cnt_pre := cnt $ 1 init 0
| reset := true when (cnt_pre = N-1)
|)
where
  integer cnt_pre; event reset;
end;%COUNTER_N%

<table>
<thead>
<tr>
<th>tick</th>
<th>tt</th>
<th>tt</th>
<th>tt</th>
<th>tt</th>
<th>tt</th>
<th>tt</th>
<th>tt</th>
<th>tt</th>
</tr>
</thead>
<tbody>
<tr>
<td>cnt</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>cnt_pre</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>...</td>
</tr>
</tbody>
</table>

Exercises...

1. Define a process $\text{Sum}$ in which on each occurrence of its unique input $i$ of real type, the sum of occurred values of $i$ until now is computed in the output signal $s$.

2. Define a process $\text{Average}$ in which on each occurrence of its unique input $i$ of real type, the average of occurred values of $i$ until now is computed in the output signal $\text{avg}$.

Exercises... (cont'd)

1. Given a constant parameter $N$ of integer type, define a process $\text{AverageN}$ in which on each occurrence of its unique input $i$ of real type, the average of the $N$ previous values of $i$ from now is computed in the output signal $\text{avg}$.

2. Let $i$ be a positive integer signal. Define a signal $\text{max}$ in a process $\text{Maximum}$, which takes at the current logical instant the most recent maximum value held by $i$.

Note: $i$ may be initialized to 0.0 by using the following expression: $\text{init}[\text{to } N:0.0]$.

Clock-oriented specifications

Extended constructs for clock manipulation

- Set operations on clocks
  - Empty/null clock: $\sim 0$
  - Union (upper bound): $x1 \sim + x2$
  - Intersection (lower bound): $x1 \sim -$ $x2$
  - Relative complement (difference): $x1 \sim - x2$
Clock-oriented specifications

Extended constructs for clock manipulation

- Set operations on clocks
  - Empty/null clock: \( \sim 0 \)
  - Union (upper bound): \( x_1 \sim + x_2 \)
  - Intersection (lower bound): \( x_1 \sim * x_2 \)
  - Relative complement (difference): \( x_1 \sim - x_2 \)

- Clock comparison
  - Inferiority: \( x_1 \sim < x_2 \)
  - Superiority: \( x_1 \sim > x_2 \)
  - Exclusion: \( x_1 \sim # x_2 \)
  - Equality: \( x_1 \sim = x_2 \)

- Set of instants at which a condition holds: \( \text{when } c \)
Clock analysis

What happens with the following program?

```plaintext
process P1 =
    (? integer s1;
     ! integer s2, s3, s4; )
    (| s2 := s1 when (s1 > 0)
        | s3 := s1 when not (s1 > 0)
        | s4 := s2 + s3
    )
```

Clock analysis

What happens with the following program3?

```plaintext
process P2 =
    (? dreal s2, s4;
     ! dreal s1, s3; )
    (| s3 := sin(s1) + s2
        | s1 := s4 default s3
    )
```

Automatic code generation

Clock inclusion hierarchy for efficient control-flow (beyond data-dependency analysis)

```
if (clk_i)
    { ...;
        if (clk_b1)
            { ... };
        if (clk_b2)
            { ... };
    }
```

Automatic code generation

Clock inclusion hierarchy for efficient control-flow (beyond data-dependency analysis)

```
if (clk_i)
    { ...;
        if (clk_b1)
            { ... };
        if (clk_b2)
            { ... };
    }
```

Endochronous versus exochronous programs

3dreal type denotes real double precision
Clock inclusion hierarchy for efficient control-flow (beyond data-dependency analysis)

```
if (clk_i)
  { ...
    if (clk_b1)
      { ...;
        if (clk_b2)
          { ...);
    }
  }
```

Data dependency analysis

What happens with the following program?

```
process P3 =
  ( ? dreal s2, s4;
    ! dreal s1, s3;
    |
      s3 := sin(s1) + cos(s2)
    | tmp := s3 / 3.14
    | s1 := tmp * tmp
  |
where
dreal tmp;
end;
```

A demo on C code generation?
Design libraries and model-checking

**Sigali tool**: verification (model-checking) of reactive systems and discrete controller synthesis ([http://www.irisa.fr/vertecs/Softwares/sigali.html](http://www.irisa.fr/vertecs/Softwares/sigali.html))

**APEX-ARINC 653 library**: for integrated modular avionics

Avionic application design, real-time java code re-engineering

---

**Bibliographic notes**


---

**Other facilities of Polychrony**

- Environment front-end

---
Polychrony and Signal programming

Clock Constraint Specification Language (CCSL)

Abstract clock-based design analysis for MPSoCs

Clock Constraint Specification Language

UML/Marte standard modeling profile: Modeling and Analysis of Real-Time and Embedded systems

Semantic issues related to time: CCSL brings a formal foundation, with suitable expressivity

Temporal behavior modeling and reasoning
- causal relations between events
- polychronous
- partially ordered events

CCSL

Basic notions: logical clocks and instant relations
CCSL: clock relations

**Coincidence relation:** c1 equals to c2

**Clock containment:** c2 is a sub-clock of c1

**Infinitely many precedence relations:** c2 precedes c1

**Instant alternation**
CCSL: clock relations

Periodicity: \( c_2 \) is periodic on \( c_1 \)

Example of specification

Downscaling transformation

Intuitive description

A downscaling application model

Functional specification
Mapped downscaling application

Allocation specification on a platform

> Image Producer
> Downscaler
> Image Consumer

<<allocate>>

<<Allocate>>

<<HwSensor>>

CMOS sensor

<<HwProcessor>>

Processor

<<HwActuator>>

TFT

Display

Consumer

Producer

Image

Processor

downscale

Image

Display

TFT

Mapped downscaling application

Frequency and rate constraint specification

> Image Producer
> Downscaler
> Image Consumer

<<allocate>>

<<Allocate>>

<<HwSensor>>

frequency, ...

<<HwProcessor>>

frequency, ...

<<HwActuator>>

frequency, ...

CMOS sensor

Processor

TFT

Display

Consumer

Producer

Image

Processor

production rate, ...

activation rate, ...

consumption rate, ...

Mapped downscaling application

Capture by abstract clock relations

> Image Producer
> Downscaler
> Image Consumer

<<allocate>>

<<Allocate>>

<<HwSensor>>

frequency, ...

<<HwProcessor>>

frequency, ...

<<HwActuator>>

frequency, ...

CMOS sensor

Processor

TFT

Display

Basic clock specification

Definition of a clock type, named "ActivationClock"

<<clockType>>

{nature = discrete, unitType = activationTick isLogical = true}

ActivationClock

currentTime():Real
Clock constraints for downscaling

Bibliographic notes


- Charles André, Frédéric Mallet, Robert de Simone. Modeling Time(s). In 10th Int. Conf on Model Driven Engineering Languages and Systems (MODELS ’07), LNCS, Pages 559-573, Nashville, TN, USA, September 2007.


Outline

- Polychrony and Signal programming
- Clock Constraint Specification Language (CCSL)
- Abstract clock-based design analysis for MPSoCs
MPSoC design challenges

Cost-effective and safe design for adaptive MPSoCs

Design Issue for adaptive MPSoCs
Identifying efficient application-architecture mappings, w.r.t. adaptive behaviors: frequencies, task migration...
MPSoC design challenges

Cost-effective and safe design for adaptive MPSoCs

Design Issue for adaptive MPSoCs
Identifying efficient application-architecture mappings, w.r.t. adaptive behaviors: frequencies, task migration...

Starting point: Y-chart

Starting point: Y-chart (focus)
**Classy design analysis framework**

**Application behavior**

**Application specification**

![Application Diagram](image)

Clock modeling: events: $e_0, e_1, e_2$

**Architecture behavior**

**Architecture specification**

![Architecture Diagram](image)

- $f_0 = 300, f_1 = 200, f_2 = 150$;
- Frequency of reference clock $\mathcal{K}$: $\text{LCM}(f_0, f_1, f_2) = 600$.

Clock modeling

| $\mathcal{K}$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | ...
|---------------|---|---|---|---|---|---|---|---|---|---
| $\rho_0$      |   |   |   |   |   |   |   |   |   |   
| $\rho_1$      |   |   |   |   |   |   |   |   |   |   
| $\rho_2$      |   |   |   |   |   |   |   |   |   |   

"Starting point: Y-chart (focus)"
If $f_0 = 300$, $f_1 = 200$, $f_2 = 150$;
frequency of reference clock $K$: $LCM(f_0, f_1, f_2) = 600$.

**Clock modeling**

<table>
<thead>
<tr>
<th>$K$</th>
<th>0 1 2 3 4 5 6 7 8 ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_0$</td>
<td>● ● ● ● ● ● ● ...</td>
</tr>
<tr>
<td>$p_1$</td>
<td>● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>$p_2$</td>
<td>● ● ● ● ● ● ● ...</td>
</tr>
</tbody>
</table>

If $f_0 = 300$, $f_1 = 200$, $f_2 = 150$;
frequency of reference clock $K$: $LCM(f_0, f_1, f_2) = 600$.

**Clock modeling**

<table>
<thead>
<tr>
<th>$K$</th>
<th>0 1 2 3 4 5 6 7 8 ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_0$</td>
<td>● ● ● ● ● ● ● ...</td>
</tr>
<tr>
<td>$p_1$</td>
<td>● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>$p_2$</td>
<td>● ● ● ● ● ● ● ...</td>
</tr>
</tbody>
</table>

If $f_0 = 300$, $f_1 = 200$, $f_2 = 150$;
frequency of reference clock $K$: $LCM(f_0, f_1, f_2) = 600$.

**Clock modeling**

<table>
<thead>
<tr>
<th>$K$</th>
<th>0 1 2 3 4 5 6 7 8 ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_0$</td>
<td>● ● ● ● ● ● ● ...</td>
</tr>
<tr>
<td>$p_1$</td>
<td>● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>$p_2$</td>
<td>● ● ● ● ● ● ● ...</td>
</tr>
</tbody>
</table>

If $f_0 = 300$, $f_1 = 200$, $f_2 = 150$;
frequency of reference clock $K$: $LCM(f_0, f_1, f_2) = 600$.

**Clock modeling**

<table>
<thead>
<tr>
<th>$K$</th>
<th>0 1 2 3 4 5 6 7 8 ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_0$</td>
<td>● ● ● ● ● ● ● ...</td>
</tr>
<tr>
<td>$p_1$</td>
<td>● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>$p_2$</td>
<td>● ● ● ● ● ● ● ...</td>
</tr>
</tbody>
</table>

**Adaptive architecture behavior**

| instant 4: $p_0 : f_0 \rightarrow 2f_0$, adaptation penalty: one cycle. |

**Clock modeling**

<table>
<thead>
<tr>
<th>$K$</th>
<th>0 1 2 3 4 5 6 7 8 ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_0$</td>
<td>● ● ● ● ● ● ● ...</td>
</tr>
<tr>
<td>$p_1$</td>
<td>● ● ● ● ● ● ●</td>
</tr>
<tr>
<td>$p_2$</td>
<td>● ● ● ● ● ● ● ...</td>
</tr>
</tbody>
</table>
Adaptive architecture behavior

Architecture specification

- instant 4: $p_0 : f_0 \rightarrow 2f_0$, adaptation penalty: one cycle.

Clock modeling

<table>
<thead>
<tr>
<th>$K$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_0$</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>...</td>
</tr>
<tr>
<td>$p_1$</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>...</td>
</tr>
</tbody>
</table>

Adaptive architecture behavior

Architecture specification

- instant 4: $p_0 : f_0 \rightarrow 2f_0$, adaptation penalty: one cycle.

Clock modeling

<table>
<thead>
<tr>
<th>$K$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_0$</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>...</td>
</tr>
<tr>
<td>$p_1$</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>...</td>
</tr>
</tbody>
</table>

Mapping of application on platform

- Function $T \rightarrow P$;
- Elementary costs $\alpha$, i.e., time and energy costs;
- Best and Worst Case costs $[\alpha_\perp, \alpha_\top]$. 
Abstract clock modeling of scheduling

- $A = \{e_0, e_1\}$ with costs $[2, 2]$ and $[1, 1]$ cycles;

<table>
<thead>
<tr>
<th>$K$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\rho_0$</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\text{clk}(A/p_0)$</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- event executions: 1 followed by $-1$;
Abstract clock modeling of scheduling

- $A = \{e_0, e_1\}$ with costs $[2, 2]$ and $[1, 1]$ cycles;
- $\mathcal{K}$
  \[
  \begin{array}{cccccccccc}
  0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
  \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
  \end{array}
  \]
- $\rho_0$
  \[
  \begin{array}{cccccccccc}
  1 & -1 & -1 & -1 & 0 & -1 & 1 & -1 \\
  \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
  \end{array}
  \]

- **event executions**: 1 followed by −1;
- **non-execution**: 0 followed by −1.
- **As Soon As Possible scheduling** preserving precedence relations.

Performance analysis

- $\mathcal{K}$
  \[
  \begin{array}{cccccccccc}
  0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
  \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
  \end{array}
  \]
- $\rho_0$
  \[
  \begin{array}{cccccccccc}
  1 & -1 & -1 & -1 & 0 & -1 & 1 & -1 \\
  \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
  \end{array}
  \]
- $\rho_1$
  \[
  \begin{array}{cccccccccc}
  0 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\
  \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
  \end{array}
  \]

- Execution time: the longest clock, e.g., $9/\mathcal{K}$;
- Usage ratio of PEs: busy cycles/overall cycles, e.g., $\rho_0 : 2/3$;
- Energy consumption of PEs: energy costs for (running tasks + being idle)

Design space exploration: CLASSY Tool

http://www.lirmm.fr/~gamatie/pages/Tool/Classy.html

- Exhaustive and heuristic-based **exploration methods**
- Pareto-optimal mapping and platform config. solutions w.r.t. **time** and **energy**

Experimental results: CLASSY vs SoCLib

Performance analysis\(^4\) on JPEG encoder

- less precise but similar observation tendency.

\(^4\)Assuming deadlock-free communications.
Experimental results: CLASSY vs SoCLib

Energy consumption analysis\(^5\) on JPEG encoder

\[\text{SystemC simulation} \quad \text{Abstract clock analysis}\]

\begin{center}
\begin{tabular}{|c|c|c|c|c|c|}
\hline
Configurations & 1 & 6 & 7 & 8 & 9 \\
\hline
\hline
Energy [J] & 350 & 300 & 300 & 250 & 150 \\
\hline
\end{tabular}
\end{center}

\(^5\)Assuming deadlock-free communications.

Summary

Abstract clock-based approaches for
- programming with Signal
- reasoning about temporal properties
- design space exploration for MPSoCs

Bibliographic notes


THE END...