Resource-Constrained Optimal Scheduling of SDF Graphs via Timed Automata

Waheed Ahmad, Robert de Groote, Philip Hölzenspies, Mariëlle Stoelinga, Jaco van de Pol
University of Twente, Netherlands

June 26, 2014

Supported by EU FP7 Projects SENSATION and POLCA
Modern multimedia applications impose high demands on system performance.
Motivation

- Modern multimedia applications impose high demands on system performance.
- Resource usage must be minimal.
Motivation

- Modern multimedia applications impose high demands on system performance.
- Resource usage must be minimal.
- Hence, trade-off between resource usage and performance is needed.
Modern multimedia applications impose high demands on system performance.

Resource usage must be minimal.

Hence, trade-off between resource usage and performance is needed.

Current scheduling schemes require max-plus algebraic semantics or transformation to HSDF graphs.
Motivation

- Modern multimedia applications impose high demands on system performance.
- Resource usage must be minimal.
- Hence, trade-off between resource usage and performance is needed.
- Current scheduling schemes require max-plus algebraic semantics or transformation to HSDF graphs.
- Infinite sequence of data and Overlapping iterations.
Motivation

- Modern multimedia applications impose high demands on system performance.
- Resource usage must be minimal.
- Hence, trade-off between resource usage and performance is needed.
- Current scheduling schemes require max-plus algebraic semantics or transformation to HSDF graphs.
- Infinite sequence of data and Overlapping iterations.
- An alternative, novel analysis of SDF graphs using Timed Automata (TA).
Motivation

- Modern multimedia applications impose high demands on system performance.
- Resource usage must be minimal.
- Hence, trade-off between resource usage and performance is needed.
- Current scheduling schemes require max-plus algebraic semantics or transformation to HSDF graphs.
- Infinite sequence of data and Overlapping iterations.
- An alternative, novel analysis of SDF graphs using Timed Automata (TA).
- Automatically derives a schedule that fits on available processors:
  - maximises throughput
  - handles heterogeneous systems
  - Model checking
Overview

- Recapitulation of Synchronous Dataflow Graphs
- Timed Automata
- Analysis of SDF Graphs using Timed Automata
- Results
- Conclusions
An SDF Graph is a tuple $G = (A, D, \text{Tok}_0, \tau)$ where:

- $A$ is a finite set of actors,

An SDF Graph is a tuple $G = (A, D, Tok_0, \tau)$ where:

- $A$ is a finite set of actors,
- $D$ is a finite set of dependency edges $D \subseteq A^2 \times \mathbb{N}^2$, 

An SDF Graph is a tuple $G = (A, D, \text{Tok}_0, \tau)$ where:

- $A$ is a finite set of actors,
- $D$ is a finite set of dependency edges $D \subseteq A^2 \times \mathbb{N}^2$,
- $\text{Tok}_0 : D \rightarrow \mathbb{N}$ denotes initial tokens in each edge and
An SDF Graph is a tuple $G = (A, D, Tok_0, \tau)$ where:

- $A$ is a finite set of actors,
- $D$ is a finite set of dependency edges $D \subseteq A^2 \times \mathbb{N}^2$,
- $Tok_0 : D \rightarrow \mathbb{N}$ denotes initial tokens in each edge and
- $\tau : A \rightarrow \mathbb{N}_{\geq 1}$ assigns an execution time to each actor.
An SDF Graph is a tuple $G = (A, D, Tok_0, \tau)$ where:

- $A$ is a finite set of actors,
- $D$ is a finite set of dependency edges $D \subseteq A^2 \times \mathbb{N}^2$,
- $Tok_0 : D \rightarrow \mathbb{N}$ denotes initial tokens in each edge and
- $\tau : A \rightarrow \mathbb{N}_{\geq 1}$ assigns an execution time to each actor.
Methodology

Application SDF Graph

Translation to TA

Application Model

Architecture Model

Mapping

Performance Analysis

Measures of Interest

Architecture

Translation to TA
Not all actors can be mapped onto all processors.

**Definition**

A *processor application model* is a tuple \((P, \zeta)\) consisting of,

- a finite set \(P\) of processors and,
- a function \(\zeta : P \to 2^A\).
Maximum Throughput via Self-Timed Execution

Presented by Ghamarian, A.H. et. al (ACSD, 2006) and implemented in SDF3.

\[ (\rho_0, \upsilon_0) = ((0, 0, 6, 2, 1), (\emptyset, \emptyset, \emptyset)) \]

\[ (\rho_r, \upsilon_r) = ((0, 0, 3, 0, 0), (\emptyset, \{2\}, \emptyset)) \]

\[ ((2, 0, 2, 0, 1), (\emptyset, \emptyset, \{\emptyset, \emptyset\})) \]
Periodic Schedule

- Consistency

A repetition vector $\gamma$ is a function such that, for all edges: production rate (edge) $\times \gamma(producer) = consumption rate (edge) \times \gamma(consumer)$

$\Gamma \gamma = 0,$ \hspace{1cm} (1)

$$\Gamma((a,b,p,q)),x) = \begin{cases} p, & \text{if } x = a - q, \\ b, & \text{if } x = b, \\ 0, & \text{otherwise} \end{cases}$$ \hspace{1cm} (2)

An SDF graph with $n$ actors has a periodic schedule if and only if its topology matrix $\Gamma$ has a rank $n - 1.$ [Lee and Messerschmitt, 1987]

If $\Gamma \gamma = 0$ then $\Gamma(K \gamma) = 0$ for any integer constant $K.$
Periodic Schedule

- Consistency
- A repetition vector $\gamma$ is function such that, for all edges: production rate (edge) * $\gamma$ (producer) = consumption rate (edge) * $\gamma$ (consumer)
Periodic Schedule

- Consistency
- A repetition vector $\gamma$ is function such that, for all edges: production rate (edge) $\times \gamma$ (producer) = consumption rate (edge) $\times \gamma$ (consumer)

$$\Gamma \gamma = 0,$$  \hspace{1cm} (1)
Periodic Schedule

- Consistency
- A repetition vector $\gamma$ is a function such that, for all edges: production rate (edge) * $\gamma$ (producer) = consumption rate (edge) * $\gamma$ (consumer)

$$\Gamma\gamma = 0,$$  \hspace{1cm} (1)

$$\Gamma((a, b, p, q), x) = \begin{cases} 
    p, & \text{if } x = a \\
    -q, & \text{if } x = b \\
    0, & \text{otherwise}
\end{cases}$$ \hspace{1cm} (2)
Periodic Schedule

- Consistency
- A repetition vector $\gamma$ is function such that, for all edges: production rate (edge) * $\gamma$ (producer) = consumption rate (edge) * $\gamma$ (consumer)

$$\Gamma\gamma = 0,$$  \hspace{1cm} (1)

- An SDF graph with $n$ actors has a periodic schedule if and only if its topology matrix $\Gamma$ has a rank $n-1$. [Lee and Messerschmitt, 1987]
Consistency

A repetition vector $\gamma$ is function such that, for all edges: production rate (edge) \* $\gamma$ (producer) = consumption rate (edge) \* $\gamma$ (consumer)

$$\Gamma \gamma = 0,$$  \hspace{1cm} (1)

$$\Gamma((a, b, p, q), x) = \begin{cases} p, & \text{if } x = a \\ -q, & \text{if } x = b \\ 0, & \text{otherwise.} \end{cases}$$  \hspace{1cm} (2)

An SDF graph with $n$ actors has a periodic schedule if and only if its topology matrix $\Gamma$ has a rank $n - 1$. [Lee and Messerschmitt, 1987]

If $\Gamma \gamma = 0$ then $\Gamma(K \gamma) = 0$ for any integer constant $K$. 


Maximum Throughput via Self-Timed Execution

\[ (\rho_0, v_0) \]

\[ (\rho_r, v_r) \]

\[ f_{at} \]

\[ f_{ap} \]
\((\rho_0, \nu_0) = \text{initial state,}\)
\begin{itemize}
\item $(\rho_0, \nu_0) = \text{initial state,}$
\item $(\rho_r, \nu_r) = \text{recurrent state}$
\end{itemize}
(\(\rho_0, \nu_0\)) = initial state,

(\(\rho_r, \nu_r\)) = recurrent state

For each actor \(a \in A\), let \(f_{at}\) = number of firings in the transient phase and
(ρ₀, ν₀) = initial state,
(ρᵣ, νᵣ) = recurrent state
For each actor a ∈ A, let \( f_{at} \) = number of firings in the transient phase and
\( f_{ap} = \) number of firings in the periodic phase.
Fastest Execution

- \((\rho_0, \nu_0) = \text{initial state,}\)
- \((\rho_r, \nu_r) = \text{recurrent state}\)
- For each actor \(a \in A\), let \(f_{at} = \text{number of firings in the transient phase and}\)
- \(f_{ap} = \text{number of firings in the periodic phase.}\)
- \(m = \text{number of iterations per period.}\)
(ρ₀, υ₀) = initial state,
(ρᵣ, υᵣ) = recurrent state
For each actor a ∈ A, let fₜₐ = number of firings in the transient phase and
fₚₐ = number of firings in the periodic phase.
m = number of iterations per period.
If a periodic phase in a self-timed execution is repeated for n times, then fₚₐ is equal to nmγ(a).
(\(\rho_0, \nu_0\)) = initial state,
(\(\rho_r, \nu_r\)) = recurrent state
For each actor \(a \in A\), let \(f_{at}\) = number of firings in the transient phase and
\(f_{ap}\) = number of firings in the periodic phase.
\(m\) = number of iterations per period.
If a periodic phase in a self-timed execution is repeated for \(n\) times, then \(f_{ap}\) is equal to \(nm\gamma(a)\).
Self-timed execution = fastest execution
Maximum Throughput via Fastest Execution

\[ \rho_0, \upsilon_0 \]

\[ \rho_r, \upsilon_r \]

\[ f_{ap} \]

\[ f_{ar} \]
Maximum Throughput via Fastest Execution

\[ f_{at} = k \gamma(a) - f_{at} \]

\[ (\rho_0, \nu_0) \]

\[ (\rho_r, \nu_r) \]

\[ f_{ap} \]
Lemma

From the state \((\rho_r, \nu_r)\), if the SDF graph is executed in such a way that each actor \(a \in A\) fires equal to \(f'_a = k \gamma(a) - f_a\) for some constant \(k\), the SDF graph reaches the initial state \((\rho_0, \nu_0)\).

Proof.

Total number of firings for each actor \(a \in A\) in this case are:

\[
= f_a + f_{ap} + f'_a \\
= f_a + nm\gamma(a) + k\gamma(a) - f_a \\
= (nm + k)\gamma(a)
\]

As we know, \(\Gamma(n\gamma) = 0\) for any constant \(n\). □
Lemma

The fastest execution of every consistent and strongly connected SDF graph

- repeats the periodic phase \( n \) times
- if each actor \( a \in A \) fires \((nm + k_{min})\gamma(a)\) times for \( n, k_{min} \in \mathbb{N} \).
Lemma

The fastest execution of every consistent and strongly connected SDF graph

- repeats the periodic phase $n$ times
- if each actor $a \in A$ fires $(nm + k_{min})\gamma(a)$ times for $n, k_{min} \in \mathbb{N}$.

**Uppaal** has an option of generating a *Fastest Trace*. 
Self-timed execution assumes there is an *unbounded* number of processors.

Let $P_{\text{min}}$ be the minimum number of processors required to allow self-timed execution.

**Lemma**

*For every consistent and strongly connected SDF graph mapped on a processor application model $(P, \zeta)$ in such a way that,*

- $\bigcup \zeta(p) = A$ and $\emptyset \subset P \subseteq P_{\text{min}}$,

*the maximal throughput of the SDF graph is determined from the periodic phase of the fastest execution to the $i^{\text{th}}$ multiple of the repetition vector for some constant $i$.*

(University of Twente)
Timed Automata

- Timed automata are finite state machines with clocks.

A timed automaton $A$ is a tuple $(L, \text{Act}, \mathcal{C}, E, \text{Inv}, l_0)$, where

- $L$ is a set of locations;
- $\text{Act}$ is a finite set of actions, co-actions and internal $\lambda$-actions;
- $\mathcal{C}$ is a finite set of clocks;
- $E \subseteq L \times \text{Act} \times B(\mathcal{C}) \times 2^\mathcal{C} \times L$ is a set of edges;
- $\text{Inv} : L \rightarrow B(\mathcal{C})$ assigns an invariant to each location;
- and $l_0 \in L$ is the initial location.
Timed automata are finite state machines with clocks.

Clocks are variables which can evaluate to a real number.
Timed Automata

- Timed automata are finite state machines with clocks.
- Clocks are variables which can evaluate to a real number.
- Clocks can be defined in each automaton to measure the time progress.
Timed Automata

- Timed automata are finite state machines with clocks.
- Clocks are variables which can evaluate to a real number.
- Clocks can be defined in each automaton to measure the time progress.
- All clocks evolve at the same pace to represent the global progress of time.
Timed Automata

- Timed automata are finite state machines with clocks.
- Clocks are variables which can evaluate to a real number.
- Clocks can be defined in each automaton to measure the time progress.
- All clocks evolve at the same pace to represent the global progress of time.
- The actual value of a clock can be either tested or reset (not assigned).
Timed Automata

- Timed automata are finite state machines with clocks.
- Clocks are variables which can evaluate to a real number.
- Clocks can be defined in each automaton to measure the time progress.
- All clocks evolve at the same pace to represent the global progress of time.
- The actual value of a clock can be either tested or reset (not assigned).

A timed automaton $\mathcal{A}$ is a tuple $(L, Act, C, E, Inv, l^0)$, where $L$ is a set of locations; $Act$ is a finite set of actions, co-actions and internal $\lambda$-actions; $C$ is a finite set of clocks; $E \subseteq L \times Act \times B(C) \times 2^C \times L$ is a set of edges; $Inv : L \rightarrow B(C)$ assigns an invariant to each location; and $l^0 \in L$ is the initial location.
Timed Automata Models

(a) UPPAAL model $A_G$ for actors $u, v, w$

(b) UPPAAL model Processor for actors $u, v, w$
Results

Figure: Schedule using four processors

Figure: Schedule using three processors
<table>
<thead>
<tr>
<th>Proc.</th>
<th>Thr</th>
<th>Throughput</th>
<th>Deadlock</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Memory</td>
<td>Time</td>
</tr>
<tr>
<td>Bipartite graph</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1/42</td>
<td>38036</td>
<td>0.41</td>
</tr>
<tr>
<td>3</td>
<td>1/44</td>
<td>37880</td>
<td>0.31</td>
</tr>
<tr>
<td>2</td>
<td>1/51</td>
<td>37884</td>
<td>0.21</td>
</tr>
<tr>
<td>1</td>
<td>1/73</td>
<td>2008</td>
<td>0.21</td>
</tr>
<tr>
<td>MPEG-4 Decoder</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1/4</td>
<td>99460</td>
<td>259.18</td>
</tr>
<tr>
<td>5</td>
<td>1/5</td>
<td>48960</td>
<td>12.04</td>
</tr>
<tr>
<td>4</td>
<td>1/5</td>
<td>39628</td>
<td>0.71</td>
</tr>
<tr>
<td>3</td>
<td>1/6</td>
<td>2008</td>
<td>0.1</td>
</tr>
<tr>
<td>2</td>
<td>1/8</td>
<td>2008</td>
<td>0.1</td>
</tr>
<tr>
<td>1</td>
<td>1/13</td>
<td>2008</td>
<td>0.1</td>
</tr>
<tr>
<td>MP3 Playback Application</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1/1880</td>
<td>99176</td>
<td>7.25</td>
</tr>
<tr>
<td>1</td>
<td>1/2118</td>
<td>59472</td>
<td>1.41</td>
</tr>
<tr>
<td>MP3 Decoder</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1/9</td>
<td>38172</td>
<td>0.22</td>
</tr>
<tr>
<td>1</td>
<td>1/15</td>
<td>2088</td>
<td>0.1</td>
</tr>
<tr>
<td>Audio Echo Canceller</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1/23</td>
<td>2874728</td>
<td>302.97</td>
</tr>
<tr>
<td>3</td>
<td>1/24</td>
<td>484736</td>
<td>133.65</td>
</tr>
<tr>
<td>2</td>
<td>1/25</td>
<td>149264</td>
<td>18.29</td>
</tr>
<tr>
<td>1</td>
<td>1/70</td>
<td>55572</td>
<td>1.41</td>
</tr>
</tbody>
</table>
Tool Support

![Diagram of a tool support system with nodes and edges labeled with channels and process identifiers.]

-chan: 1 --> (0) --> 1
-chan: 1 --> (0) --> 1
-a2b: 3 --> (5) --> 5
-proc:1
-proc:2

(University of Twente) SDFG via TA June 26, 2014 20 / 21
Conclusions and Future Directions

- Combination of the flexibility of automata with the efficiency of SDF graphs to derive optimum schedules.
Conclusions and Future Directions

- Combination of the flexibility of automata with the efficiency of SDF graphs to derive optimum schedules.
- Analysis of the properties such as the absence of deadlocks and unboundedness, safety, liveness and reachability.
Conclusions and Future Directions

- Combination of the flexibility of automata with the efficiency of SDF graphs to derive optimum schedules.
- Analysis of the properties such as the absence of deadlocks and unboundedness, safety, liveness and reachability.
- Multi-core LTL model checking using opaal+LTS\text{MIN} to tackle state-space explosion.
Conclusions and Future Directions

- Combination of the flexibility of automata with the efficiency of SDF graphs to derive optimum schedules.
- Analysis of the properties such as the absence of deadlocks and unboundedness, safety, liveness and reachability.
- Multi-core LTL model checking using opaal+LTSmin to tackle state-space explosion.
- Energy optimal reachability analysis with the help of Priced Timed Automata.