Response Time Analysis of Synchronous Data Flow Programs on a Many-Core Processor

Hamza Rihani, Matthieu Moy, Claire Maiza, Robert I. Davis, Sebastian Altmeyer

RTNS’16, October 19, 2016
Execution of Synchronous Data Flow Programs

High level representation

Single-core code generation

Static non-preemptive scheduling

```
int main_app(i1, i2)
{
    na = NA(i1);
    ne = NE(i2);
    nb = NB(na);
    nd = ND(na);
    nf = NF(ne);
    o = NC(nb, nd, nf);
    return o;
}
```
Execution of Synchronous Data Flow Programs

Multi/Many-core code generation

static non-preemptive scheduling
Execution of Synchronous Data Flow Programs

Multi/Many-core

code generation

static non-preemptive scheduling

Respect the dependency constraints
Execution of Synchronous Data Flow Programs

Multi/Many-core code generation

static non-preemptive scheduling

High level representation

\[ \begin{array}{c}
\text{NA} \quad \text{NB} \quad \text{NC} \\
\tau_1 \quad \tau_2 \quad \tau_3 \quad \tau_4 \\
\text{ND} \quad \text{NE} \quad \text{NF} \\
\tau_5 \quad \tau_6
\end{array} \]

\[ \begin{array}{c}
\text{Respect the dependency constraints} \\
\text{Set the release dates to get precise upper bounds on the interference}
\end{array} \]

\[ \begin{align*}
\text{int NF (\ldots)} & \{ \\
\text{\quad \quad \quad \quad \quad // task } \tau_6 \text{ return (\ldots);}
\}
\end{align*} \]

\[ \begin{align*}
\text{int NE (\ldots)} & \{ \\
\text{\quad \quad \quad \quad \quad // task } \tau_5 \text{ return (\ldots);}
\}
\end{align*} \]

\[ \begin{align*}
\text{int ND (\ldots)} & \{ \\
\text{\quad \quad \quad \quad \quad // task } \tau_4 \text{ return (\ldots);}
\}
\end{align*} \]

\[ \begin{align*}
\text{int NC (\ldots)} & \{ \\
\text{\quad \quad \quad \quad \quad // task } \tau_3 \text{ return (\ldots);}
\}
\end{align*} \]

\[ \begin{align*}
\text{int NB (\ldots)} & \{ \\
\text{\quad \quad \quad \quad \quad // task } \tau_2 \text{ return (\ldots);}
\}
\end{align*} \]

\[ \begin{align*}
\text{int NA (\ldots)} & \{ \\
\text{\quad \quad \quad \quad \quad // task } \tau_1 \text{ return (\ldots);}
\}
\end{align*} \]
Contributions

1. Precise accounting for interference on shared resources in a many-core processor

![Diagram of a multi-level arbiter to the shared memory]
Contributions

1. Precise accounting for interference on shared resources in a many-core processor

2. Model of a multi-level arbiter to the shared memory
Contributions

1. Precise accounting for interference on shared resources in a many-core processor

2. Model of a multi-level arbiter to the shared memory

3. Response time and release dates analysis respecting dependencies.
Outline

1 Motivation and Context

2 Models Definition
   - Architecture Model
   - Execution Model
   - Application Model

3 Multicore Response Time Analysis of SDF Programs

4 Evaluation

5 Conclusion and Future Work
Outline

1 Motivation and Context

2 Models Definition
   - Architecture Model
   - Execution Model
   - Application Model

3 Multicore Response Time Analysis of SDF Programs

4 Evaluation

5 Conclusion and Future Work
Kalray MPPA 256 Bostan
16 compute clusters + 4 I/O clusters
Dual NoC
Per cluster:
- 16 cores + 1 Resource Manager
- NoC Tx, NoC Rx, Debug Unit
- 16 shared memory banks per cluster (total 2 MB)
Per cluster:
- 16 cores + 1 Resource Manager
- NoC Tx, NoC Rx, Debug Unit
- 16 shared memory banks per cluster (total 2 MB)
- Multi-level bus arbiter per memory bank
Per cluster:
- 16 cores + 1 Resource Manager
- NoC Tx, NoC Rx, Debug Unit
- 16 shared memory banks per cluster (total 2 MB)
- Multi-level bus arbiter per memory bank
Tasks mapping on cores

Static non-preemptive scheduling

Spatial Isolation
different tasks go to different memory banks

Interference from communications

Execution model:
execute in a “local” bank
write to a “remote” bank

Single phase: execute and write data.

Two phases: execute then write data.

memory access pattern

8 shared memory banks

8 shared memory banks
○ Tasks mapping on cores
○ Static non-preemptive scheduling
Tasks mapping on cores
- Static non-preemptive scheduling
- Spatial Isolation
  - different tasks go to different memory banks
Tasks mapping on cores
Static non-preemptive scheduling
Spatial Isolation
different tasks go to different memory banks
Interference from communications
Tasks mapping on cores
- Static non-preemptive scheduling
- Spatial Isolation
  - different tasks go to different memory banks
- Interference from communications
- Execution model:
  - execute in a “local” bank
  - write to a “remote” bank

Single phase: execute and write data.
Tasks mapping on cores
Static non-preemptive scheduling
Spatial Isolation
  different tasks go to different memory banks
Interference from communications
Execution model:
  execute in a “local” bank
  write to a “remote” bank

Single phase: execute \textit{and} write data.

Two phases: execute \textit{then} write data.
Application Model

- Direct Acyclic Task Graph
- Mono-rate (or at least harmonic rates)
- Fixed mapping and execution order

![Diagram of Application Model]
Application Model

- Direct Acyclic Task Graph
- Mono-rate (or at least harmonic rates)
- Fixed mapping and execution order

Each task $\tau_i$:
Application Model

- Direct Acyclic Task Graph
- Mono-rate (or at least harmonic rates)
- Fixed mapping and execution order

Each task $\tau_i$:
- Processor Demand, Memory Demand
Application Model

- Direct Acyclic Task Graph
- Mono-rate (or at least harmonic rates)
- Fixed mapping and execution order

Each task $\tau_i$:
- Processor Demand, Memory Demand
- Release date ($rel_i$), response time ($R_i$)

Diagram:
- Nodes represent tasks $\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6$
- Edges indicate dependencies
- Labeling of tasks and variables
- Timeline with release and response times
- Isolation and interference concepts
Application Model

- Direct Acyclic Task Graph
- Mono-rate (or at least harmonic rates)
- Fixed mapping and execution order

Each task $\tau_i$:
- Processor Demand, Memory Demand
- Release date ($rel_i$), response time ($R_i$)
Application Model

- Direct Acyclic Task Graph
- Mono-rate (or at least harmonic rates)
- Fixed mapping and execution order

Each task $\tau_i$:
- Processor Demand, Memory Demand
- Release date ($rel_i$), response time ($R_i$)

Find $R_i$ (including the interference)
Find $rel_i$ respecting precedence constraints
Outline

1 Motivation and Context

2 Models Definition
   ■ Architecture Model
   ■ Execution Model
   ■ Application Model

3 Multicore Response Time Analysis of SDF Programs

4 Evaluation

5 Conclusion and Future Work
Response Time Analysis

\[ R = PD + I^\text{BUS}(R) \]

- Response Time

- Processor Demand
- Bus Interference (given a model of the bus arbiter)
- Interference from preempting tasks (no preemption: \( I^\text{PROC} = 0 \))
- Interference from DRAM refreshes (out of scope. \( I^\text{DRAM} = 0 \))

Recursive formula \( \Rightarrow \) fixed-point algorithm.

\[ I^\text{BUS}(R) = \sum_{b \in B} I^\text{BUS}_b(R) \]
where \( B \): a set of memory banks
Response Time Analysis

\[ R = PD + I^{BUS}(R) \]

- Response Time
  - Processor Demand

- Bus Interference (given a model of the bus arbiter)
- Interference from preempting tasks (no preemption: \( I_{PROC} = 0 \))
- Interference from DRAM refreshes (out of scope: \( I_{DRAM} = 0 \))

\[ I^{BUS}(R) = \sum_{b \in B} I^{BUS}_b(R) \]

where \( B \): a set of memory banks

\( R \): Recursive formula ⇒ fixed-point algorithm.

\( I^{BUS}(R) \): Multiple shared resources (memory banks)
Response Time Analysis

\[ R = PD + I_{BUS}(R) \]

- Response Time
  - Processor Demand
    - Bus Interference
      - \((given \ a \ model \ of \ the \ bus \ arbiter)\)
Response Time Analysis

\[ R = PD + I_{BUS}^R + I_{PROC}^R + I_{DRAM}^R \]

- Response Time
- Processor Demand
  - Bus Interference 
    (*given a model of the bus arbiter*)
  - Interference from preempting tasks 
    (*no preemption: \( I_{PROC} = 0 \)*)
  - Interference from DRAM refreshes 
    (*out of scope. \( I_{DRAM} = 0 \)*)
Response Time Analysis

\[ R = PD + I_{BUS}(R) + I_{PROC}(R) + I_{DRAM}(R) \]

- Response Time
- Processor Demand
  - Bus Interference
    *(given a model of the bus arbiter)*
  - Interference from preempting tasks
    *(no preemption: \( I_{PROC} = 0 \))*
  - Interference from DRAM refreshes
    *(out of scope. \( I_{DRAM} = 0 \))*

- Recursive formula \( \Rightarrow \) fixed-point algorithm.
Response Time Analysis

\[ R = PD + I_{BUS}(R) + I_{PROC}(R) + I_{DRAM}(R) \]

- Response Time
  - Processor Demand
    - Bus Interference
      \((\text{given a model of the bus arbiter})\)
    - Interference from preempting tasks
      \((\text{no preemption: } I_{PROC} = 0)\)
    - Interference from DRAM refreshes
      \((\text{out of scope. } I_{DRAM} = 0)\)
  - Recursive formula \(\Rightarrow\) fixed-point algorithm.
  - Multiple shared resources (memory banks)
Response Time Analysis

\[ R = PD + I_{BUS}^R + I_{PROC}^R + I_{DRAM}^R \]

- Response Time
  - Processor Demand
    - Bus Interference
      (given a model of the bus arbiter)
    - Interference from preempting tasks
      (no preemption: \( I_{PROC}^R = 0 \))
    - Interference from DRAM refreshes
      (out of scope: \( I_{DRAM}^R = 0 \))
  - Recursive formula \( \Rightarrow \) fixed-point algorithm.
  - Multiple shared resources (memory banks)

\[ I_{BUS}^R = \sum_{b \in B} I_{BUS}^b \]

where \( B \): a set of memory banks
Response Time Analysis

\[ R = PD + I_{BUS}(R) + I_{PROC}(R) + I_{DRAM}(R) \]

- Response Time
  - Processor Demand
    - Bus Interference
      (given a model of the bus arbiter)
    - Interference from preempting tasks
      (no preemption: \( I_{PROC} = 0 \))
    - Interference from DRAM refreshes
      (out of scope. \( I_{DRAM} = 0 \))

- Recursive formula \( \Rightarrow \) fixed-point algorithm.
- Multiple shared resources (memory banks)

\[ I_{BUS}(R) = \sum_{b \in B} I_{BUS}^b(R) \]

where \( B \): a set of memory banks

Requires a model of the bus arbiter
Model of the MPPA Bus

\[ J_{BUS}^{i}: \text{delay from all accesses + concurrent ones} \]
Model of the MPPA Bus

\[ \tau_i \in \text{task of interest} \]

- \( P_0 \): delay from all accesses + concurrent ones
- \( S^b_i \): number of accesses of task \( \tau_i \) to bank \( b \)

\[ S^b_i = \text{Memory Demand to bank } b \]

\[ \begin{align*}
L_{v1} & = S^b_i \\
Lv_2 & = \min(A_{G2}, b_i, Lv_1) \\
Lv_3 & = \min(A_{G3}, b_i, Lv_2) \\
Lv_4 & = \sum_{y=1}^{15} \min(A_{y}, b_i, Lv_1)
\end{align*} \]
Model of the MPPA Bus

\( \text{BUS} \): delay from all accesses + concurrent ones

\( S^b_i \): number of accesses of task \( \tau_i \) to bank \( b \)

\( S^b_i = \text{Memory Demand to bank } b \)

\( A_{y,b}^i \): number of concurrent accesses from core \( y \) to bank \( b \)

\( L_{v1} = S^b_i \)

\( L_{v2} = L_{v1} + \sum_{y=1}^{15} \min(A_{y,b}^i, L_{v1}) \)
Model of the MPPA Bus

\[ \begin{align*}
Lv_1 &= S_i^b \\
Lv_2 &= Lv_1 + \sum_{y=1}^{15} \min(A_{i,y}^{y,b}, Lv_1) \\
Lv_3 &= Lv_2 + \min(A_{i}^{G2,b}, Lv_2)
\end{align*} \]

- \( P_0 \): task of interest
- \( y \): core
- \( t \): time
- \( A_{i}^{y,b} \): number of concurrent accesses from core \( y \) to bank \( b \)
- \( S_i^b \): number of accesses of task \( \tau_i \) to bank \( b \)
- \( S_i^b = \text{Memory Demand to bank } b \)
- \( P_{00,40,80} \): region of interest
- \( G_3 \): high priority
- \( Rx \), \( Tx \), \( DSU \), \( RM \)
- \( RR_{3-1} \), \( RR_{2-1} \), \( FP \)
- \( Shared Memory Bank \)
- \( I_{BUS} \): delay from all accesses + concurrent ones

\( I_{BUS} = \) delay from all accesses + concurrent ones

\( S_i^b = \) Memory Demand to bank \( b \)

\( A_{i}^{y,b} = \) number of concurrent accesses from core \( y \) to bank \( b \)
Model of the MPPA Bus

\[ Lv_1 = S_i^b \]
\[ Lv_2 = Lv_1 + \sum_{y=1}^{15} \min( A_{yi}^{y,b}, Lv_1 ) \]
\[ Lv_3 = Lv_2 + \min( A_{yG2,i}^{y,b}, Lv_2 ) \]
\[ Lv_4 = Lv_4 + A_{iG3}^{G3,b} \]

\( i_{BUS} \): delay from all accesses + concurrent ones
\( S_i^b \): number of accesses of task \( \tau_i \) to bank \( b \)
\[ S_i^b = \text{Memory Demand to bank } b \]
\( A_{iG2}^{y,b} \): number of concurrent accesses from core \( y \) to bank \( b \)
Model of the MPPA Bus

\[ \begin{align*}
    L_{v1} &= S^b_i \\
    L_{v2} &= L_{v1} + \sum_{y=1}^{15} \min(A_{i}^{y,b}, L_{v1}) \\
    L_{v3} &= L_{v2} + \min(A_{i}^{G2,b}, L_{v2}) \\
    L_{v4} &= L_{v4} + A_{i}^{G3,b}
\end{align*} \]

\[ I_{bus}^{B} = L_{v4} \times \text{Bus Delay} \]
Model of the MPPA Bus

\[ I_{b}^{BUS} = L_{V4} \times \text{Bus Delay} \]

\[ L_{V1} = S_{i}^{b} \]

\[ L_{V2} = L_{V1} + \sum_{y=1}^{15} \min(A_{i}^{y,b}, L_{V1}) \]

\[ L_{V3} = L_{V2} + \min(A_{i}^{G2,b}, L_{V2}) \]

\[ L_{V4} = L_{V4} + A_{i}^{G3,b} \]

\[ I_{b}^{BUS} : \text{delay from all accesses + concurrent ones} \]

\[ S_{i}^{b} : \text{number of accesses of task } \tau_{i} \text{ to bank } b \]

\[ S_{i}^{b} = \text{Memory Demand to bank } b \]

\[ A_{i}^{y,b} : \text{number of concurrent accesses from core } y \text{ to bank } b \]

\[ A_{i}^{y,b} = \sum \text{overlapping concurrent accesses} \]
Model of the MPPA Bus

- $I_{b}^{BUS}$: delay from all accesses + concurrent ones
- $S_{i}^{b}$: number of accesses of task $\tau_i$ to bank $b$
  \[ S_{i}^{b} = \text{Memory Demand to bank } b \]
- $A_{i}^{y,b}$: number of concurrent accesses from core $y$ to bank $b$
  \[ A_{i}^{y,b} = \sum \text{overlapping concurrent accesses} \]

\[
L_{v_1} = S_{i}^{b}
\]
\[
L_{v_2} = L_{v_1} + \sum_{y=1}^{15} \min(A_{i}^{y,b}, L_{v_1})
\]
\[
L_{v_3} = L_{v_2} + \min(A_{i}^{G_2,b}, L_{v_2})
\]
\[
L_{v_4} = L_{v_4} + A_{i}^{G_3,b}
\]

$I_{b}^{BUS} = L_{v_4} \times \text{Bus Delay}$
Response Time Analysis with Dependencies

1. Start with initial release dates.

WCRT analysis

for all $i$ do
  $R_{i+1}^{l+1} \leftarrow PD_i + BUS(R_i^l, rel_i)$
end for
Response Time Analysis with Dependencies

1. Start with initial release dates.
2. Compute response times
   
   \[
   \begin{align*}
   & \text{WCRT analysis} \\
   & \text{for all } i \text{ do} \\
   & \quad R_i^{l+1} \leftarrow \text{PD}_i + \text{BUS}(R_i^l, \text{rel}_i) \\
   & \text{end for}
   \end{align*}
   \]

   initial \( \text{rel}_i^0 \)

   \( R_i^{l+1} \neq R_i^l \)
Start with initial release dates.

2 Compute response times

... ...
Response Time Analysis with Dependencies

1. Start with initial release dates.
2. Compute response times
   ... ... ... a fixed-point is reached!

---

WCRT analysis
for all $i$
$$R_{i+1}^{l+1} \leftarrow PD_i + BUS(R_i^l, rel_i)$$
end for
Response Time Analysis with Dependencies

1. Start with initial release dates.
2. Compute response times...
   ... … a fixed-point is reached!
3. Update the release dates.

WCRT analysis

for all $i$ do
    $R_i^{l+1} \leftarrow PD_i + I\text{BUS}(R_i^l, rel_i)$
end for

initial $rel_i^0$

$R_i^{l+1} \neq R_i^l$

Update release dates

for all $i$ do
    $rel_i \leftarrow$ latest finish time of all the dependencies
end for
1. Start with initial release dates.
2. Compute response times
   ... ... ... a fixed-point is reached!
3. Update the release dates.
4. Repeat until no release date changes
   (another fixed-point iteration).

WCRT analysis

\[
\text{for all } i \text{ do } R_{i+1}^{l+1} \leftarrow PD_i + \text{BUS}(R_i, rel_i) \\
\text{end for}
\]

Update release dates

\[
\text{for all } i \text{ do } rel_i \leftarrow \text{latest finish time of all the dependencies} \\
\text{end for}
\]
Response Time Analysis with Dependencies

1. Start with initial release dates.
2. Compute response times
   ... ... ... a fixed-point is reached!
3. Update the release dates.
4. Repeat until no release date changes
   (another fixed-point iteration).

WCRT analysis

\[ \text{for all } i \text{ do} \]
\[ R_{i}^{l+1} \leftarrow PD_{i} + I_{BUS}(R_{i}^{l}, rel_{i}) \]
\[ \text{end for} \]

Update release dates

\[ \text{for all } i \text{ do} \]
\[ rel_{i} \leftarrow \text{latest finish time of all the dependencies} \]
\[ \text{end for} \]

Return: \((rel_{i}, R_{i})\)
Convergence Toward a Fixed-point

- Convergence of the 1st fixed-point iteration:

```
WCRT analysis
for all i do
R_{i+1}^{l+1} ← PD_{i} + \sum_{j=1}^{l} BUS(R_{j}^{l}, rel_{j})
end for
```

Update release dates
for all i do
rel_{i} ← latest finish time of all the dependencies
end for

Return: (rel_{i}, R_{i})
Convergence Toward a Fixed-point

- Convergence of the 1st fixed-point iteration:
  - Monotonic and bounded ✔

---

**Convergence of the 2nd fixed-point iteration:**
- No monotonicity: \( R_i \) and \( rel_i \) may grow or shrink at each iteration.

---

**Theorem**
- At each iteration, at least one task finds its final release date.

Full proof in our technical report: [Link](http://www-verimag.imag.fr/TR/TR-2016-1.pdf)

---

**WCRT analysis**

```
for all i do
    \( R_{i+1} \leftarrow PD_i + JBUS(R_i, rel_i) \)
end for
```

**Update release dates**

```
for all i do
    rel_i \leftarrow \text{latest finish time of all the dependencies}
end for
```

Return: \((rel_i, R_i)\)
Convergence Toward a Fixed-point

- Convergence of the 1\textsuperscript{st} fixed-point iteration:
  - Monotonic and bounded
- Convergence of the 2\textsuperscript{nd} fixed-point iteration:

WCRT analysis

\begin{align*}
\text{for all } i & \text{ do} \\
R_{i+1} & \leftarrow \text{PD}_i + \text{BUS}(R_i, \text{rel}_i) \\
\text{end for}
\end{align*}

Update release dates

\begin{align*}
\text{for all } i & \text{ do} \\
\text{rel}_i & \leftarrow \text{latest finish time of all the dependencies} \\
\text{end for}
\end{align*}

Return: \((\text{rel}_i, R_i)\)
Convergence Toward a Fixed-point

- Convergence of the 1

  - Monotonic and bounded

- Convergence of the 2

  - no monotonicity: \( R_i \) and \( rel_i \) may grow or shrink at each iteration.

Theorem
At each iteration, at least one task finds its final release date.
Full proof in our technical report: http://www-verimag.imag.fr/TR/TR-2016-1.pdf

WCRT analysis

\[
\text{for all } i \text{ do } \\
R_{i+1} \leftarrow PD_i + BUS(R_i, rel_i) \\
\text{end for}
\]

Update release dates

\[
\text{for all } i \text{ do } \\
rel_i \leftarrow \text{latest finish time of all the dependencies} \\
\text{end for}
\]

Return: \((rel_i, R_i)\)
Convergence Toward a Fixed-point

- Convergence of the 1\textsuperscript{st} fixed-point iteration:
  - Monotonic and bounded
- Convergence of the 2\textsuperscript{nd} fixed-point iteration:
  - no monotonicity: $R_i$ and $rel_i$ may grow or shrink at each iteration.

**Theorem**

At each iteration, at least one task finds its final release date.

Full proof in our technical report:

http://www-verimag.imag.fr/TR/TR-2016-1.pdf
Convergence Toward a Fixed-point

Theorem

At each iteration, at least one task finds its final release date.

Full proof in our technical report:
http://www-verimag.imag.fr/TR/TR-2016-1.pdf
Convergence Toward a Fixed-point

○ Convergence of the $1^{st}$ fixed-point iteration:
  ○ Monotonic and bounded ✓

○ Convergence of the $2^{nd}$ fixed-point iteration:
  ○ no monotonicity: $R_i$ and $rel_i$ may grow or shrink at each iteration.

Theorem

At each iteration, at least one task finds its final release date.

Full proof in our technical report:
http://www-verimag.imag.fr/TR/TR-2016-1.pdf
Convergence Toward a Fixed-point

- Convergence of the 1\textsuperscript{st} fixed-point iteration:
  - Monotonic and bounded \(\checkmark\)
- Convergence of the 2\textsuperscript{nd} fixed-point iteration:
  - no monotonicity: \(R_i\) and \(rel_i\) may grow or shrink at each iteration.

\textbf{Theorem}

\textit{At each iteration, at least one task finds its final release date.}

Full proof in our technical report:

http://www-verimag.imag.fr/TR/TR-2016-1.pdf
Convergence Toward a Fixed-point

- Convergence of the 1\textsuperscript{st} fixed-point iteration:
  - Monotonic and bounded \(\checkmark\)
- Convergence of the 2\textsuperscript{nd} fixed-point iteration:
  - no monotonicity: \(R_i\) and \(rel_i\) may grow or shrink at each iteration.

**Theorem**

At each iteration, at least one task finds its final release date.

Full proof in our technical report:

http://www-verimag.imag.fr/TR/TR-2016-1.pdf
Convergence Toward a Fixed-point

- Convergence of the 1^{st} fixed-point iteration:
  - Monotonic and bounded ✓

- Convergence of the 2^{nd} fixed-point iteration:
  - no monotonicity: $R_i$ and $r_i$ may grow or shrink at each iteration.

**Theorem**

*At each iteration, at least one task finds its final release date.*

Full proof in our technical report:

http://www-verimag.imag.fr/TR/TR-2016-1.pdf
Outline

1. Motivation and Context

2. Models Definition
   - Architecture Model
   - Execution Model
   - Application Model

3. Multicore Response Time Analysis of SDF Programs

4. Evaluation

5. Conclusion and Future Work
○ Flight management system controller

---

1 Pagetti et al., RTAS 2014
Evaluation: ROSACE Case Study

- Flight management system controller
- Receive from sensors and transmit to actuators

---

1 Pagetti et al., RTAS 2014
Flight management system controller

Receive from sensors and transmit to actuators

**Assumptions:**
- Tasks are mapped on 5 cores
- Debug Support Unit is disabled
- Context switches are over-approximated constants

---

1 Pagetti et al., RTAS 2014
Evaluation: ROSACE Case Study

- Flight management system controller
- Receive from sensors and transmit to actuators
- **Assumptions:**
  - Tasks are mapped on 5 cores
  - Debug Support Unit is disabled
  - Context switches are over-approximated constants

---

1 Pagetti et al., RTAS 2014
### Evaluation: ROSACE Case Study

#### Table: Task profiles of the FMS controller

<table>
<thead>
<tr>
<th>Task</th>
<th>Processor Demand (cycles)</th>
<th>Memory Demand (accesses)</th>
</tr>
</thead>
<tbody>
<tr>
<td>altitude</td>
<td>275</td>
<td>22</td>
</tr>
<tr>
<td>az_filter</td>
<td>274</td>
<td>22</td>
</tr>
<tr>
<td>h_filter</td>
<td>326</td>
<td>24</td>
</tr>
<tr>
<td>va_control</td>
<td>303</td>
<td>24</td>
</tr>
<tr>
<td>va_filter</td>
<td>301</td>
<td>23</td>
</tr>
<tr>
<td>vz_control</td>
<td>320</td>
<td>25</td>
</tr>
<tr>
<td>vz_filter</td>
<td>334</td>
<td>25</td>
</tr>
</tbody>
</table>

- Profile obtained from measurements
## Evaluation: ROSACE Case Study

<table>
<thead>
<tr>
<th>Task</th>
<th>Processor Demand (cycles)</th>
<th>Memory Demand (accesses)</th>
</tr>
</thead>
<tbody>
<tr>
<td>altitude</td>
<td>275</td>
<td>22</td>
</tr>
<tr>
<td>az_filter</td>
<td>274</td>
<td>22</td>
</tr>
<tr>
<td>h_filter</td>
<td>326</td>
<td>24</td>
</tr>
<tr>
<td>va_control</td>
<td>303</td>
<td>24</td>
</tr>
<tr>
<td>va_filter</td>
<td>301</td>
<td>23</td>
</tr>
<tr>
<td>vz_control</td>
<td>320</td>
<td>25</td>
</tr>
<tr>
<td>vz_filter</td>
<td>334</td>
<td>25</td>
</tr>
</tbody>
</table>

Table: Task profiles of the FMS controller

- Profile obtained from measurements
- Memory Demand: data and instruction cache misses + communications
### Evaluation: ROSACE Case Study

<table>
<thead>
<tr>
<th>Task</th>
<th>Processor Demand (cycles)</th>
<th>Memory Demand (accesses)</th>
</tr>
</thead>
<tbody>
<tr>
<td>altitude</td>
<td>275</td>
<td>22</td>
</tr>
<tr>
<td>az_filter</td>
<td>274</td>
<td>22</td>
</tr>
<tr>
<td>h_filter</td>
<td>326</td>
<td>24</td>
</tr>
<tr>
<td>va_control</td>
<td>303</td>
<td>24</td>
</tr>
<tr>
<td>va_filter</td>
<td>301</td>
<td>23</td>
</tr>
<tr>
<td>vz_control</td>
<td>320</td>
<td>25</td>
</tr>
<tr>
<td>vz_filter</td>
<td>334</td>
<td>25</td>
</tr>
</tbody>
</table>

Table: Task profiles of the FMS controller

- Profile obtained from measurements
- Memory Demand: data and instruction cache misses + communications
- Moreover:
  - NoC Rx: writes 5 words
  - NoC Tx: reads 2 words
### Evaluation: ROSACE Case Study

#### Table: Task profiles of the FMS controller

<table>
<thead>
<tr>
<th>Task</th>
<th>Processor Demand (cycles)</th>
<th>Memory Demand (accesses)</th>
</tr>
</thead>
<tbody>
<tr>
<td>altitude</td>
<td>275</td>
<td>22</td>
</tr>
<tr>
<td>az_filter</td>
<td>274</td>
<td>22</td>
</tr>
<tr>
<td>h_filter</td>
<td>326</td>
<td>24</td>
</tr>
<tr>
<td>va_control</td>
<td>303</td>
<td>24</td>
</tr>
<tr>
<td>va_filter</td>
<td>301</td>
<td>23</td>
</tr>
<tr>
<td>vz_control</td>
<td>320</td>
<td>25</td>
</tr>
<tr>
<td>vz_filter</td>
<td>334</td>
<td>25</td>
</tr>
</tbody>
</table>

- Profile obtained from measurements
- Memory Demand: data and instruction cache misses + communications
- Moreover:
  - *NoC Rx*: writes 5 words
  - *NoC Tx*: reads 2 words

⚠️ Experiments: Find the smallest schedulable hyper-period
Evaluation: Experiments

Smallest schedulable hyper-period

<table>
<thead>
<tr>
<th>Processor cycles</th>
<th>1 bank</th>
<th>5 banks</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPPA</td>
<td>E5: Pessimistic</td>
<td>E5: All accesses interfere</td>
</tr>
<tr>
<td>RR</td>
<td>E4: 1-Phase (w/o release)</td>
<td>E4, E3: We don't use the release dates</td>
</tr>
<tr>
<td>Bus Policy</td>
<td>E3: 2-Phase (w/o release)</td>
<td>E2, E1: Our approach. We use the release dates</td>
</tr>
</tbody>
</table>

- E5: Pessimistic
- E4: 1-Phase (w/o release)
- E3: 2-Phase (w/o release)
- E2: 1-Phase
- E1: 2-Phase
Evaluation: Experiments

<table>
<thead>
<tr>
<th>Processor cycles</th>
<th>Bus Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>E5: Pessimistic</td>
<td>MPPA</td>
</tr>
<tr>
<td>E4: 1-Phase (w/o release)</td>
<td>RR</td>
</tr>
<tr>
<td>E3: 2-Phase (w/o release)</td>
<td>MPPA</td>
</tr>
<tr>
<td>E2: 1-Phase</td>
<td>RR</td>
</tr>
<tr>
<td>E1: 2-Phase</td>
<td>RR</td>
</tr>
</tbody>
</table>

Pessimistic assumption: High priority tasks are bounded by 1 access per bank

Smallest schedulable hyper-period

E5: All accesses interfere
Evaluation: Experiments

Smallest schedulable hyper-period

- E5: All accesses interfere
- E4, E3: We don’t use the release dates

○ Pessimistic assumption:
  High priority tasks are bounded by 1 access per bank
Evaluation: Experiments

Smallest schedulable hyper-period

- **E5**: All accesses interfere
- **E4, E3**: We don’t use the release dates
- **E2, E1**: Our approach. We use the release dates

○ Pessimistic assumption:
High priority tasks are bounded by 1 access per bank

<table>
<thead>
<tr>
<th>Processor cycles</th>
<th>E5: Pessimistic</th>
<th>E4: 1-Phase (w/o release)</th>
<th>E3: 2-Phase (w/o release)</th>
<th>E2: 1-Phase</th>
<th>E1: 2-Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 bank</td>
<td>MPPA</td>
<td>RR</td>
<td>MPPA</td>
<td>RR</td>
<td></td>
</tr>
<tr>
<td>5 banks</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **1 bank**
- **5 banks**
Evaluation: Experiments

Smallest schedulable hyper-period

- **1-Phase model**
  - Memory access pattern

- **2-Phase model**
  - Memory access pattern

**Bus Policy**
- E5: Pessimistic
- E4: 1-Phase (w/o release)
- E3: 2-Phase (w/o release)
- E2: 1-Phase
- E1: 2-Phase

**Processor cycles**
- 1 bank
- 5 banks

- **Pessimistic assumption:** High priority tasks are bounded by 1 access per bank
- **Phases are modeled as sub-tasks**

- **E5: All accesses interfere**
- **E4, E3:** We don’t use the release dates
- **E2, E1:** Our approach. We use the release dates
Evaluation: Experiments

Taking into account the memory banks improves the analysis with a factor in [1.77, 2.52].
Evaluation: Experiments

Taking into account the memory banks improves the analysis with a factor in [1.77, 2.52]

<table>
<thead>
<tr>
<th>Bus Policy</th>
<th>Processor cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>E5: Pessimistic</td>
<td>E4: 1-Phase (w/o release)</td>
</tr>
<tr>
<td>E4: 1-Phase</td>
<td>E3: 2-Phase (w/o release)</td>
</tr>
<tr>
<td>E3: 2-Phase</td>
<td>E2: 1-Phase</td>
</tr>
<tr>
<td>E2: 1-Phase</td>
<td>E1: 2-Phase</td>
</tr>
</tbody>
</table>

Smallest schedulable hyper-period

<table>
<thead>
<tr>
<th>Artifact</th>
<th>Consistent</th>
<th>Complete</th>
<th>Well documented</th>
<th>Easy to reuse</th>
<th>Evaluated</th>
<th>RTNS</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MPPA</td>
<td>4.15</td>
<td>4.12</td>
<td>1.68</td>
<td>1.29</td>
<td>~1.01</td>
<td>0.77</td>
</tr>
<tr>
<td>RR</td>
<td>3.3</td>
<td>3.29</td>
<td>1.24</td>
<td>1.13</td>
<td>~1.01</td>
<td>0.91</td>
</tr>
</tbody>
</table>
Taking into account the memory banks improves the analysis with a factor in $[1.77, 2.52]$.

<table>
<thead>
<tr>
<th>Bus Policy</th>
<th>Processor cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>E5: Pessimistic</td>
<td>MPPA: 4.15</td>
</tr>
<tr>
<td>E4: 1−Phase (w/o release)</td>
<td>RR: 3.3</td>
</tr>
<tr>
<td>E3: 2−Phase (w/o release)</td>
<td>MPPA: 1.68</td>
</tr>
<tr>
<td>E2: 1−Phase</td>
<td>RR: 1.24</td>
</tr>
<tr>
<td>E1: 2−Phase</td>
<td>MPPA: 1.29</td>
</tr>
</tbody>
</table>

Smallest schedulable hyper-period

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MPPA</td>
<td>4.15</td>
<td>4.12</td>
<td>1.68</td>
<td>1.29</td>
<td>~1.01</td>
<td>0.77</td>
</tr>
<tr>
<td>RR</td>
<td>3.3</td>
<td>3.29</td>
<td>1.24</td>
<td>1.13</td>
<td>~1.01</td>
<td>0.91</td>
</tr>
</tbody>
</table>
Evaluation: Experiments

Taking into account the memory banks improves the analysis with a factor in \([1.77, 2.52]\).

### Processor cycles

<table>
<thead>
<tr>
<th>Bus Policy</th>
<th>1 bank</th>
<th>5 banks</th>
</tr>
</thead>
<tbody>
<tr>
<td>E5: Pessimistic</td>
<td>16000</td>
<td>16000</td>
</tr>
<tr>
<td>E4: 1-Phase (w/o release)</td>
<td>12000</td>
<td>12000</td>
</tr>
<tr>
<td>E3: 2-Phase (w/o release)</td>
<td>8000</td>
<td>8000</td>
</tr>
<tr>
<td>E2: 1-Phase</td>
<td>4000</td>
<td>4000</td>
</tr>
<tr>
<td>E1: 2-Phase</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Smallest schedulable hyper-period

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MPPA</td>
<td>4.15</td>
<td>4.12</td>
<td>1.68</td>
<td>1.29</td>
<td>~1.01</td>
<td>0.77</td>
</tr>
<tr>
<td>RR</td>
<td>3.3</td>
<td>3.29</td>
<td>1.24</td>
<td>1.13</td>
<td>~1.01</td>
<td>0.91</td>
</tr>
</tbody>
</table>

Speedup factors
Evaluation: Experiments

Taking into account the memory banks improves the analysis with a factor in $[1.77, 2.52]$.

<table>
<thead>
<tr>
<th>Bus Policy</th>
<th>Processor cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>E5: Pessimistic</td>
<td>5 banks</td>
</tr>
<tr>
<td>E4: 1-Phase (w/o release)</td>
<td></td>
</tr>
<tr>
<td>E3: 2-Phase (w/o release)</td>
<td></td>
</tr>
<tr>
<td>E2: 1-Phase</td>
<td></td>
</tr>
<tr>
<td>E1: 2-Phase</td>
<td></td>
</tr>
</tbody>
</table>

Smallest schedulable hyper-period

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MPPA</td>
<td>4.15</td>
<td>4.12</td>
<td>1.68</td>
<td>1.29</td>
<td>∼1.01</td>
<td>0.77</td>
</tr>
<tr>
<td>RR</td>
<td>3.3</td>
<td>3.29</td>
<td>1.24</td>
<td>1.13</td>
<td>∼1.01</td>
<td>0.91</td>
</tr>
</tbody>
</table>
Taking into account the memory banks improves the analysis with a factor in $[1.77, 2.52]$.
Outline

1 Motivation and Context

2 Models Definition
   • Architecture Model
   • Execution Model
   • Application Model

3 Multicore Response Time Analysis of SDF Programs

4 Evaluation

5 Conclusion and Future Work
Conclusion

- A response time analysis of SDF on the Kalray MPPA 256
Conclusion

- A response time analysis of SDF on the Kalray MPPA 256

- Given:
  - Task profile
  - Mapping of Tasks
  - Execution Order
Conclusion

- A response time analysis of SDF on the Kalray MPPA 256

- Given:
  - Task profile
  - Mapping of Tasks
  - Execution Order

- We compute:
  - Tight response times taking into account the interference.
  - Release dates respecting the dependency constraints.
Conclusion

- A response time analysis of SDF on the Kalray MPPA 256

- Given:
  - Task profile
  - Mapping of Tasks
  - Execution Order

- We compute:
  - Tight response times taking into account the interference.
  - Release dates respecting the dependency constraints.

model of the multi-level arbiter
Conclusion

- A response time analysis of SDF on the Kalray MPPA 256

- Given:
  - Task profile
  - Mapping of Tasks
  - Execution Order

- We compute:
  - Tight response times taking into account the interference
  - Release dates respecting the dependency constraints.
Conclusion

- A response time analysis of SDF on the Kalray MPPA 256

- Given:
  - Task profile
  - Mapping of Tasks
  - Execution Order

- We compute:
  - Tight response times taking into account the interference.
  - Release dates respecting the dependency constraints.

- Not restricted to SDF
Future Work

- Model of the Resource Manager.

Diagram:

- 8 shared memory banks
- NoC Rx and NoC Tx
- RM, DSU
- Processors: P0, P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12, P13, P14, P15

Questions?
Future Work

- Model of the Resource Manager.
- Model of the NoC accesses.
- Memory access pipelining.
- Model of blocking and non-blocking accesses.

Tighter estimation of context switches and other interrupts.

Questions?
Future Work

- Model of the Resource Manager.
- Model of the NoC accesses.

tighter estimation of context switches and other interrupts
Future Work

- Model of the Resource Manager.
- Model of the NoC accesses.

Tighter estimation of context switches and other interrupts.

Use the output of any NoC analysis.
Future Work

- Model of the Resource Manager.
- Model of the NoC accesses.
- Memory access pipelining.

tighter estimation of context switches and other interrupts

use the output of any NoC analysis
Future Work

- Model of the Resource Manager.
- Model of the NoC accesses.
- Memory access pipelining.

Tighter estimation of context switches and other interrupts:

Use the output of any NoC analysis:

Current assumption: bus delay is 10 cycles.

Questions?
Future Work

- Model of the Resource Manager.
- Model of the NoC accesses.
- Memory access pipelining.
- Model Blocking and non-blocking accesses.

Use the output of any NoC analysis.

Current assumption: bus delay is 10 cycles.

Tighter estimation of context switches and other interrupts.
Future Work

- Model of the Resource Manager.
- Model of the NoC accesses.
- Memory access pipelining.
- Model Blocking and non-blocking accesses.

- tighter estimation of context switches and other interrupts
- use the output of any NoC analysis
- current assumption: bus delay is 10 cycles
- reads are blocking
- writes are non-blocking
Future Work

- Model of the Resource Manager.
- Model of the NoC accesses.
- Memory access pipelining.
- Model Blocking and non-blocking accesses.

use the output of any NoC analysis

tighter estimation of context switches and other interrupts

current assumption: bus delay is 10 cycles

reads are blocking writes are non-blocking

Questions?
BACKUP
Multicore Response Time Analysis

Example: Fixed Priority bus arbiter, PE1 > PE0
Bus access delay = 10

\[ T_1 \]

\[ T_0 \]

\[ T_1 \]

\[ T_1 \]

1Altmeyer et al., RTNS 2015
Example: Fixed Priority bus arbiter, PE1 > PE0

Bus access delay = 10

○ Task of interest running on PE0:

\[ R_0 = 10 + 3 \times 10 \] (response time in isolation)

\[ ^1 \text{Altmeyer et al., RTNS 2015} \]
Multicore Response Time Analysis

Example: Fixed Priority bus arbiter, PE1 > PE0
Bus access delay = 10

- Task of interest running on PE0:
  \[ R_0 = 10 + 3 \times 10 \] (response time in isolation)
  \[ R_1 = 10 + 3 \times 10 + 2 \times 10 = 60 \]

---

Altmeyer et al., RTNS 2015
Multicore Response Time Analysis

Example: Fixed Priority bus arbiter, PE1 > PE0
Bus access delay = 10

- Task of interest running on PE0:
  \[ R_0 = 10 + 3 \times 10 \] (response time in isolation)
  \[ R_1 = 10 + 3 \times 10 + 2 \times 10 = 60 \]
  \[ R_2 = 10 + 3 \times 10 + 2 \times 10 + 2 \times 10 = 80 \]

---

\(^1\) Altmeyer et al., RTNS 2015
Example: Fixed Priority bus arbiter, PE1 > PE0
Bus access delay = 10

Task of interest running on PE0:

- \( R_0 = 10 + 3 \times 10 \) (response time in isolation)
- \( R_1 = 10 + 3 \times 10 + 2 \times 10 = 60 \)
- \( R_2 = 10 + 3 \times 10 + 2 \times 10 + 2 \times 10 = 80 \)
- \( R_3 = 10 + 3 \times 10 + 2 \times 10 + 2 \times 10 + 0 = 80 \) (fixed-point)

\(^1\) Altmeyer et al., RTNS 2015
The Global Picture

- High-level Program
- Code Generation
- Timing models (static analysis)
- Local WCRT Analysis
- Probabilistic Models
- Tasks + Dependencies
- Static Mapping/Scheduling
- Tasks WCRT + WC Access
- WCRT with Interferences
- Mapping
- Execution Order
- Release Dates
- Binary Generation
- Executable Binary