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

\[
\begin{align*}
\tau_1 & \to \text{NA} \to i_1 \\
\tau_2 & \to \text{NB} \to \tau_3 \\
\tau_4 & \to \text{ND} \\
\tau_5 & \to \text{NE} \\
\tau_6 & \to \text{NF} \to i_2
\end{align*}
\]

High level representation

Single-core code generation

static non-preemptive scheduling

\begin{verbatim}
int main_app(i_1, i_2)
{
    na = NA(i_1);
    ne = NE(i_2);
    nb = NB(na);
    nd = ND(na);
    nf = NF(ne);
    o = NC(nb, nd, nf);
    return o;
}
\end{verbatim}
Execution of Synchronous Data Flow Programs

High level representation

Multi/Many-core code generation

static non-preemptive scheduling
Execution of Synchronous Data Flow Programs

High level representation

Multi/Many-core code generation

static non-preemptive scheduling

✓ Respect the dependency constraints
Execution of Synchronous Data Flow Programs

Respect the dependency constraints

Set the release dates to get precise upper bounds on the interference

Multi/Many-core code generation

static non-preemptive scheduling

PE2

PE1

PE0

1

2

3

4

5

6

NA

NB

NC

ND

NE

NF

i1

τ1

τ2

τ3

τ4

i2

τ5

τ6

High level representation

int NF (...) {
    // task τ4
    return (...) ;
}

int NE (...) {
    // task τ5
    return (...) ;
}

int ND (...) {
    // task τ6
    return (...) ;
}

int NC (...) {
    // task τ3
    return (...) ;
}

int NB (...) {
    // task τ2
    return (...) ;
}

int NA(...) {
    // task τ1
    return (...) ;
}
Contributions

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

![Diagram showing task of interest, P_0, and y with time on the x-axis.]
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
Architecture Model

- 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)
Architecture Model

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
Execution Model

- 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.
Execution Model

- 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
Application Model

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

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

Each task $\tau_i$:
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 ($R_i$)

### Isolation

![Diagram of Application Model](image)
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

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

- Response Time
Response Time Analysis

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

- Response Time
- Processor Demand
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
      (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)
Response Time Analysis

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

- **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.
- Multiple shared resources (memory banks)

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

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

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

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

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

\text{Requires a model of the bus arbiter}
Model of the MPPA Bus

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

\[ i_{BUS} : \text{delay from all accesses + concurrent ones} \]

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

\[ S_{i}^{b} = \text{Memory Demand to bank } b \]
Model of the MPPA Bus

\[ L_{v1} = S_i^b \]

\[ L_{v2} = L_{v1} + \sum_{y=1}^{15} \min(A_i^{y,b}, L_{v1}) \]

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

\[ S_i^b : \text{number of accesses of task } \tau_i \text{ to bank } b \]

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

\[ A_i^{y,b} : \text{number of concurrent accesses from core } y \text{ to bank } b \]
Model of the MPPA Bus

\[ L_{v1} = S_i^b \]
\[ L_{v2} = L_{v1} + \sum_{y=1}^{15} \min(A_i^{y,b}, L_{v1}) \]
\[ L_{v3} = L_{v2} + \min(A_i^{G2,b}, L_{v2}) \]

\( j_b^{\text{BUS}} \): 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 \( b \)
Model of the MPPA Bus

\[ L_v^1 = S_i^b \]
\[ L_v^2 = L_v^1 + \sum_{y=1}^{15} \min(A_{i}^{y,b}, L_v^1) \]
\[ L_v^3 = L_v^2 + \min(A_{i}^{G2,b}, L_v^2) \]
\[ L_v^4 = L_v^3 + A_{i}^{G3,b} \]

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

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

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

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

\[ L_{v1} = S_{i}^{b} \]
\[ L_{v2} = L_{v1} + \sum_{y=1}^{15} \min(A_{i}^{y,b}, L_{v1}) \]
\[ L_{v3} = L_{v2} + \min(A_{i}^{G2,b}, L_{v2}) \]
\[ L_{v4} = L_{v4} + A_{i}^{G3,b} \]

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

\[ I_b^{BUS} = Lv_4 \times \text{Bus Delay} \]

\[ Lv_1 = S_i^b \]
\[ Lv_2 = Lv_1 + \sum_{y=1}^{15} \min(A_i^{y,b}, Lv_1) \]
\[ Lv_3 = Lv_2 + \min(A_i^{G2,b}, Lv_2) \]
\[ Lv_4 = Lv_4 + A_i^{G3,b} \]

- \( I_b^{BUS} \): 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 \( b \)
  \[ A_i^{y,b} = \sum \text{overlapping concurrent accesses} \]

\( \tau_i \): task of interest
Model of the MPPA Bus

\[ \text{Bus Delay} = L_{V4} \times \text{Bus Delay} \]

\[ L_{V1} = S^b_i \]

\[ L_{V2} = L_{V1} + \sum_{y=1}^{15} \min\left( A^y,b_i, L_{V1} \right) \]

\[ L_{V3} = L_{V2} + \min\left( A^{G2,b}_i, L_{V2} \right) \]

\[ L_{V4} = L_{V3} + A^{G3,b}_i \]

\[ I^\text{BUS}_b = \text{delay from all accesses + concurrent ones} \]

\[ S^b_i = \text{number of accesses of task } \tau_i \text{ to bank } b \]

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

\[ A^y,b_i = \text{number of concurrent accesses from core } y \text{ to bank } b \]

\[ A^y,b_i = \sum \text{overlapping concurrent accesses} \]

\[ A^y,b_i \text{ depends on } rel_i \text{ and } R_i \]
Response Time Analysis with Dependencies

Start with initial release dates.

WCRT analysis

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

1. Start with initial release dates.
2. Compute response times

\[ \text{WCRT analysis} \]

\[
\text{for all } i \text{ do}
\]

\[
R_{i+1} = PD_i + I_{BUS}(R_i, rel_i)
\]

end for
1. Start with initial release dates.
2. Compute response times

\[ R_{l+1}^{i} = PD_{l} + I_{BUS}(R_{l}^{i}, rel_{l}) \]

WCRT analysis

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

Response Time Analysis with Dependencies

PE2

PE1

PE0

WCRT analysis

for all $i$

\[
R_{i+1} = PD_i + I_{BUS}(R_i, rel_i)
\]

end for

initial $rel_i^0$

$R_{i+1} \neq R_i$

$R_i$
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
$R_{i+1} = PD_i + I^{BUS}(R_i, rel_i)$
end for

**Update release dates**
for all $i$ do
$rel_i \leftarrow$ latest finish time of all the dependencies
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

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

Update release dates

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

initial \( rel_{i}^{0} \)

new \( rel_{i} \) repeat

\( R_{i}^{l+1} \neq R_{i}^{l} \)
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

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

Update release dates

for all $i$ do
    $rel_i \leftarrow$ 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:

WCRT analysis

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

Update release dates

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

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

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

WCRT analysis

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

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:
  - Monotonic and bounded
- Convergence of the 2nd fixed-point iteration:

WCRT analysis

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

Update release dates

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

\[
R_i^{l+1} \neq R_i^l
\]

Return: (rel, Ri)
Convergence Toward a Fixed-point

- Convergence of the 1<sup>st</sup> fixed-point iteration:
  - Monotonic and bounded ✓
- Convergence of the 2<sup>nd</sup> fixed-point iteration:
  - No monotonicity: \( R_i \) and \( rel_i \) may grow or shrink at each iteration.

WCRT analysis

\[
\begin{align*}
\text{for all } i & \text{ do} \\
R_i^{i+1} & \leftarrow \text{PD}_i + t_{\text{BUS}}(R_i^i, rel_i) \\
\text{end for}
\end{align*}
\]

Update release dates

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

Return: \((rel_i, R_i)\)
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}

*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\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

\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 ✓

- 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\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\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

\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
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
Evaluation: ROSACE Case Study

- Flight management system controller

---

1 Pagetti et al., RTAS 2014
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
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>
<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
**Task** | **Processor Demand (cycles)** | **Memory Demand (accesses)**
--- | --- | ---
altitude | 275 | 22
az_filter | 274 | 22
h_filter | 326 | 24
va_control | 303 | 24
va_filter | 301 | 23
vz_control | 320 | 25
vz_filter | 334 | 25

Table: Task profiles of the FMS controller

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

### 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>

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>
<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

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

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

- Smallest schedulable hyper-period

1 bank vs 5 banks

- MPPA
- RR

Bus Policy
Evaluation: Experiments

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

**Smallest schedulable hyper-period**

- **E5: All accesses interfere**

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

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

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

**Smallest schedulable hyper-period**

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

**MPPA**

- E5: All accesses interfere

**RR**

- E4, E3: We don’t use the release dates
Evaluation: Experiments

- **1 bank**
  0
  4000
  8000
  12000
  16000

- **5 banks**

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

**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
Evaluation: Experiments

- **1-Phase model**
- **2-Phase model**

**memory access pattern**

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

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

**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
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>1 bank</th>
<th>5 banks</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPPA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>E5/E1</td>
<td>4.15</td>
<td>4.12</td>
<td>1.68</td>
<td>1.29</td>
<td>∼1.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E5/E2</td>
<td>4.12</td>
<td>4.12</td>
<td>1.68</td>
<td>1.29</td>
<td>∼1.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E3/E1</td>
<td>1.68</td>
<td>1.68</td>
<td>1.24</td>
<td>1.13</td>
<td>∼1.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E4/E2</td>
<td>1.29</td>
<td>1.29</td>
<td>1.13</td>
<td>1.13</td>
<td>∼1.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E2/E1</td>
<td>∼1.01</td>
<td>∼1.01</td>
<td></td>
<td>∼1.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E4/E3</td>
<td>0.77</td>
<td>0.77</td>
<td></td>
<td>0.91</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Smallest schedulable hyper-period

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>1 bank</th>
<th>5 banks</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPPA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RR</td>
<td></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>

Speedup factors

<table>
<thead>
<tr>
<th></th>
<th>18/21</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPPA</td>
<td></td>
</tr>
<tr>
<td>RR</td>
<td></td>
</tr>
</tbody>
</table>
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>1 bank</th>
<th>5 banks</th>
</tr>
</thead>
<tbody>
<tr>
<td>E5/E1</td>
<td>4.15</td>
<td>4.12</td>
</tr>
<tr>
<td>E5/E2</td>
<td>4.12</td>
<td>1.68</td>
</tr>
<tr>
<td>E3/E1</td>
<td>4.12</td>
<td>1.68</td>
</tr>
<tr>
<td>E4/E2</td>
<td>4.12</td>
<td>1.29</td>
</tr>
<tr>
<td>E2/E1</td>
<td>~1.01</td>
<td>~1.01</td>
</tr>
<tr>
<td>E4/E3</td>
<td>0.77</td>
<td>0.91</td>
</tr>
</tbody>
</table>

**MPPA**

<table>
<thead>
<tr>
<th>Bus Policy</th>
<th>1 bank</th>
<th>5 banks</th>
</tr>
</thead>
<tbody>
<tr>
<td>E5/E1</td>
<td>3.3</td>
<td>3.29</td>
</tr>
<tr>
<td>E5/E2</td>
<td>3.29</td>
<td>1.24</td>
</tr>
<tr>
<td>E3/E1</td>
<td>3.3</td>
<td>1.24</td>
</tr>
<tr>
<td>E4/E2</td>
<td>3.3</td>
<td>1.13</td>
</tr>
<tr>
<td>E2/E1</td>
<td>~1.01</td>
<td>~1.01</td>
</tr>
<tr>
<td>E4/E3</td>
<td>0.91</td>
<td>0.91</td>
</tr>
</tbody>
</table>

**RR**

*Easy to use*
Evaluation: Experiments

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

<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>

Smallest schedulable hyper-period

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

<table>
<thead>
<tr>
<th>Processor cycles</th>
<th>MPPA</th>
<th>RR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 bank</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E5/E1</td>
<td>4.15</td>
<td></td>
</tr>
<tr>
<td>E5/E2</td>
<td>4.12</td>
<td></td>
</tr>
<tr>
<td>E3/E1</td>
<td>1.68</td>
<td></td>
</tr>
<tr>
<td>E4/E2</td>
<td>1.29</td>
<td></td>
</tr>
<tr>
<td>E2/E1</td>
<td>~1.01</td>
<td></td>
</tr>
<tr>
<td>E4/E3</td>
<td>0.77</td>
<td></td>
</tr>
<tr>
<td>5 banks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E5: Pessimistic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E4: 1-Phase (w/o release)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E3: 2-Phase (w/o release)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E2: 1-Phase</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E1: 2-Phase</td>
<td></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>
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.
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

double fixed-point algorithm
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.
Future Work

- Model of the Resource Manager.

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
Future Work

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

Tightest 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.

![Diagram showing resource management and NoC access models](image)

`tighter estimation of context switches and other interrupts`
Future Work

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

Tight estimation of context switches and other interrupts

Use the output of any NoC analysis

Current assumption: bus delay is 10 cycles
Future Work

- Model of the Resource Manager.
- Tighter estimation of context switches and other interrupts
- Model of the NoC accesses.
- Use the output of any NoC analysis
- Memory access pipelining.
- Current assumption: bus delay is 10 cycles
- Model Blocking and non-blocking accesses.
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

- 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.

Questions?
BACKUP
Multicore Response Time Analysis

Example: Fixed Priority bus arbiter, PE1 > PE0

Bus access delay = 10

---

1 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)

---

1 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:
  \[ R_0 = 10 + 3 \times 10 \text{ (response time in isolation)} \]
  \[ R_1 = 10 + 3 \times 10 + 2 \times 10 = 60 \]

---

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)

1Altmeyer et al., RTNS 2015