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

Hamza Rihani, Matthieu Moy, Claire Maiza, Robert I. Davis, Sebastian Altmeyer
Execution of Synchronous Data Flow Programs

High level representation

Single-core code generation

static non-preemptive scheduling

```c
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;
}
```
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

High level representation

Multi/Many-core code generation

static non-preemptive scheduling

Respect the dependency constraints

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

1. Precise accounting for interference on shared resources in a many-core processor
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 (total size: 2 MB)
Per cluster:
- 16 cores + 1 Resource Manager
- NoC Tx, NoC Rx, Debug Unit
- 16 shared memory banks (total size: 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 (total size: 2 MB)
- Multi-level bus arbiter per memory bank
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
- Tasks mapping on cores
- Static non-preemptive scheduling
Execution Model

- Tasks mapping on cores
- Static non-preemptive scheduling
- Spatial Isolation
  - different tasks go to different memory banks

8 shared memory banks

- 8 shared memory banks
- NoC Rx
- NoC Tx
- RM
- DSU

- Single phase: execute and write data.
- Two phases: execute then write data.
Execution Model

- Tasks mapping on cores
- Static non-preemptive scheduling
- Spatial Isolation
  - different tasks go to different memory banks
- Interference from communications

8 shared memory banks

- NoC Rx
- NoC Tx
- DSU

Single phase: execute and write data.

Two phases: execute then write data.

Memory access pattern

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
 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.
Application Model

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

![Diagram of Application Model]

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

---

/ \[\text{Find } R_i \text{ (including the interference)}\]

/ \[\text{Find } rel_i \text{ respecting precedence constraints}\]
Application Model

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

Each task $\tau_i$:

Diagram:

- Network of tasks $\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6$.
- Interference $E_0$.
- Processor Demand and 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

![Diagram of a Direct Acyclic Task Graph with tasks $\tau_1$, $\tau_2$, $\tau_3$, $\tau_4$, $\tau_5$, $\tau_6$ and their dependencies.

- Processor Demand
- Memory access time

Timeline from 00 to 160
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 Diagram]
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 ($R_i$)

```
00 40 80 120 160
```

```
rel_i
```

```
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 \( \text{rel}_i \), response time \( R_i \)

Find \( R_i \) (including the interference)
Find \( \text{rel}_i \) 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

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

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

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

Requires a model of the bus arbiter.
Response Time Analysis

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

- Response Time
  - Processor Demand

- Requires a model of the bus arbiter

- 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)

- 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
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 preemption tasks**
      - *(no preemption: \( I^{PROC} = 0 \))
    - **Interference from DRAM refreshes**
      - *(out of scope. \( I^{DRAM} = 0 \))

\( \sum_{b \in B} I^{BUS}_b(R) \) where \( B \): a set of memory banks

\[ R \] \Rightarrow \text{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.
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

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

- **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)

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

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 \((given \ a \ model \ of \ the \ bus \ arbiter)\)
  - Interference from preemted 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

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

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

\[ j_{b} = S_{b,i} + \sum_{y=1}^{\min(A_{y,b}, b_i)} \]

\[ L_{v1} = S_{b,i} + \sum_{y=1}^{\min(A_{y,b}, b_i)} \]

\[ L_{v2} = L_{v1} + \]

\[ L_{v3} = L_{v2} + \]

\[ L_{v4} = L_{v3} + \]

\[ I_{BUS} = L_{v4} \times \text{Bus Delay} \]

\[ \text{task of interest} \]
Model of the MPPA Bus

- $i_{BUS}^b$: delay from all accesses + concurrent ones
- $S_i^b$: number of accesses of task $\tau_i$ to bank $b$
  
  $$S_i^b = \text{Memory Demand to bank } b$$
Model of the MPPA Bus

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

$Lv_1 = S_i^b$

$Lv_2 = Lv_1 + \sum_{y=1}^{15} \min(A_{y,b}^i, Lv_1)$
Model of the MPPA Bus

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

- \( j_{BUS} \): 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 \( b \)

\[ P_{0} \]

\[ y \]

\[ t \]
Model of the MPPA Bus

\[ P_0 \]

\[ \text{BUS}_{i} \]: 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 \( b \)

\[ L v_{1} = S_{i}^{b} \]

\[ L v_{2} = L v_{1} + \sum_{y=1}^{15} \min( A_{i}^{y,b}, L v_{1} ) \]

\[ L v_{3} = L v_{2} + \min( A_{i}^{G2,b}, L v_{2} ) \]

\[ L v_{4} = L v_{4} + A_{i}^{G3,b} \]
Model of the MPPA Bus

$lv_1 = s_i^b$

$lv_2 = lv_1 + \sum_{y=1}^{15} \min(A_{i}^{y,b}, lv_1)$

$lv_3 = lv_2 + \min(A_{i}^{G2,b}, lv_2)$

$lv_4 = lv_4 + A_{i}^{G3,b}$

$I_{b}^{BUS} = lv_4 \times \text{Bus Delay}$

$I_{b}^{BUS}$: delay from all accesses + concurrent ones

$s_i^b$: number of accesses of task $\tau_i$ to bank $b$

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

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

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

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

\[ I_{BUS}^{P0} = L_{bus} \times \text{Bus Delay} \]

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

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

Task of interest: \( P_0 \)

overlap: \( A_{y,b}^{i} \) depends on \( rel_i \) and \( R_i \)
1 Start with initial release dates.

WCRT analysis

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

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

...
Response Time Analysis with Dependencies

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

\[ \text{WCRT analysis} \]

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

initial \( rel_i^0 \)

initial \( rel_i^0 \neq R_i^l \)
Response Time Analysis with Dependencies

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

WCRT analysis

for all $i$ do
    $\text{new rel}_i \leftarrow \text{PD}_i + I_{\text{BUS}}(R_i, \text{rel}_i)$
end for

initial $\text{rel}_i^0$
1. Start with initial release dates.
2. Compute response times
   ... ... ... a fixed-point is reached!
3. Update the release dates.

**WCRT analysis**

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

\[
\text{initial } rel_{i}^{0}
\]

\[
R_{i}^{l+1} \neq R_{i}^{l}
\]

\[
\text{for all } i \text{ do}
\]
\[
rel_{i} \leftarrow \text{latest finish time of all the dependencies}
\]
\[
\text{end for}
\]
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
for all i do
\[ R_{i+1}^{l+i} \leftarrow PD_i + BUS(R_i^l, rel_i) \]
end for

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

initial \( rel_i^0 \)
\[ R_i^{l+1} \neq R_i^l \]
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).

initial \( r_i \) is updated

WCRT analysis

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

Update release dates

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

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

- Convergence of the 1\textsuperscript{st} fixed-point iteration:

WCRT analysis

\begin{align*}
\text{for all } i & \text{ do} \\
R^{i+1}_i & \leftarrow P_D_i + \text{BUS}(R_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 1st fixed-point iteration:
  - Monotonic and bounded ✓

**WCRT analysis**

for all \(i\) do

\[ R_{i+1}^l \leftarrow PD_i + \sum_{j} BUS(R_j, rel_j) \]

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

\[
\begin{align*}
\text{PE}_2 & \quad \tau_4 \quad \tau_5 \\
\text{PE}_1 & \quad \tau_3 \\
\text{PE}_0 & \quad \tau_0 \quad \tau_1 \quad \tau_2
\end{align*}
\]

\[
\begin{align*}
\text{WCRT analysis} & \\
\text{for all } i & \text{ do} \\
R_{i+1} & \leftarrow PD_i + BUS(R_i, rel_i) \\
\text{end for}
\end{align*}
\]

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

\[
\begin{align*}
\text{return: } (rel_i, R_i)
\end{align*}
\]

initial $rel_i^0$

$R_{i+1} \neq R_i$

$rel_i$ did not change

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.

WCRT analysis

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

Update release dates

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

\[ R_i \]
\[ \text{new } rel_i \]
\[ \text{repeat} \]
\[ R_i^{l+1} \neq R_i^l \]

Initial \( rel_i^0 \)

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:

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 ✓

- 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:

\url{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.

\[ \text{Theorem} \]
\[ \text{At each iteration, at least one task finds its final release date.} \]

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.

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

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

**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](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
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
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
### 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
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 16/21
### 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

Smallest schedulable hyper-period
Evaluation: Experiments

Smallest schedulable hyper-period

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

<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>

E5: All accesses interfere
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**
  - Pessimistic assumption: High priority tasks are bounded by 1 access per bank
  - E5: All accesses interfere
  - E4, E3: We don’t use the release dates
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
Evaluation: Experiments

Memory access pattern

1 bank
5 banks

Bus Policy

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

Processor cycles

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

Phases are modeled as sub-tasks
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: Pessimistic</td>
<td>MPPA</td>
<td>RR</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>
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>4.15</td>
<td>4.12</td>
</tr>
<tr>
<td>E4: 1-Phase (w/o release)</td>
<td>1.68</td>
<td>1.29</td>
</tr>
<tr>
<td>E3: 2-Phase (w/o release)</td>
<td>∼1.01</td>
<td>0.77</td>
</tr>
<tr>
<td>E2: 1-Phase</td>
<td>3.3</td>
<td>3.29</td>
</tr>
<tr>
<td>E1: 2-Phase</td>
<td>1.24</td>
<td>1.13</td>
</tr>
</tbody>
</table>

### Smallest schedulable hyper-period

<table>
<thead>
<tr>
<th>Artifact</th>
<th>Consistent * Complete* Well documented * Easy to reuse * Evaluated * RTNS *</th>
</tr>
</thead>
<tbody>
<tr>
<td>E5/E1</td>
<td>4.15</td>
</tr>
<tr>
<td>E5/E2</td>
<td>4.12</td>
</tr>
<tr>
<td>E3/E1</td>
<td>1.68</td>
</tr>
<tr>
<td>E4/E2</td>
<td>1.29</td>
</tr>
<tr>
<td>E2/E1</td>
<td>∼1.01</td>
</tr>
<tr>
<td>E4/E3</td>
<td>0.77</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MPPA</th>
<th>RR</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPPA</td>
<td>RR</td>
</tr>
</tbody>
</table>

### Speedup factors

\[
\begin{array}{ccccccc}
\text{Speedup factors} & \text{E5/E1} & \text{E5/E2} & \text{E3/E1} & \text{E4/E2} & \text{E2/E1} & \text{E4/E3} \\
\text{MPPA} & 4.15 & 4.12 & 1.68 & 1.29 & ∼1.01 & 0.77 \\
\text{RR} & 3.3 & 3.29 & 1.24 & 1.13 & ∼1.01 & 0.91 \\
\end{array}
\]
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>
<th>MPPA</th>
<th>RR</th>
</tr>
</thead>
<tbody>
<tr>
<td>E5: Pessimistic</td>
<td>16000</td>
<td>4000</td>
<td>8000</td>
</tr>
<tr>
<td>E4: 1-Phase (w/o release)</td>
<td>8000</td>
<td>4000</td>
<td>8000</td>
</tr>
<tr>
<td>E3: 2-Phase (w/o release)</td>
<td>4000</td>
<td>4000</td>
<td>8000</td>
</tr>
<tr>
<td>E2: 1-Phase</td>
<td>8000</td>
<td>4000</td>
<td>8000</td>
</tr>
<tr>
<td>E1: 2-Phase</td>
<td>4000</td>
<td>4000</td>
<td>8000</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]

<table>
<thead>
<tr>
<th>Bus Policy</th>
<th>Processor cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>E5: Pessimistic</td>
<td>8000</td>
</tr>
<tr>
<td>E4: 1−Phase (w/o release)</td>
<td>12000</td>
</tr>
<tr>
<td>E3: 2−Phase (w/o release)</td>
<td>16000</td>
</tr>
<tr>
<td>E2: 1−Phase</td>
<td>4000</td>
</tr>
<tr>
<td>E1: 2−Phase</td>
<td>8000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bus Policy</th>
<th>Processor cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPPA</td>
<td>4000</td>
</tr>
<tr>
<td>RR</td>
<td>8000</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>
<th>Artifact</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>0.77</td>
<td></td>
</tr>
<tr>
<td>MPPA</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>
<td></td>
</tr>
</tbody>
</table>
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>E1: 2-Phase</td>
<td>~1.01</td>
</tr>
<tr>
<td>E2: 1-Phase</td>
<td>~1.01</td>
</tr>
<tr>
<td>E3: 2-Phase (w/o release)</td>
<td>0.77</td>
</tr>
<tr>
<td>E4: 1-Phase (w/o release)</td>
<td>0.91</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].

![Graph showing processor cycles vs bus policy for MPPA and RR]

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

- **Smallest schedulable hyper-period**

- **Speedup factors**:

<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

- Not restricted to SDF model of multi-level arbiter double fixed-point algorithm
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
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.

More information:
- 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**

Questions?
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

Questions?
Future Work

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

Use the output of any NoC analysis

tighter estimation of context switches and other interrupts
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
Future Work

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

**Diagram:**

- **RM** (Resource Manager)
- **NoC Rx** (NoC Receiver)
- **NoC Tx** (NoC Transmitter)
- **DSU** (Data Switch Unit)
- **8 shared memory banks**

**Legend:**

- $P_0, P_1, \ldots, P_7$
- $P_{10}, P_{11}$

- Current assumption: bus delay is 10 cycles
- 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.
- 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

\[ \text{Bus access delay} = 10 \]

---

\footnote{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
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)
  \[ R_1 = 10 + 3 \times 10 + 2 \times 10 = 60 \]

\[ ^1 \text{Altmeyer et al., RTNS 2015} \]
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)

$R_1 = 10 + 3 \times 10 + 2 \times 10 = 60$

$R_2 = 10 + 3 \times 10 + 2 \times 10 + 2 \times 10 = 80$

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

- Task of interest running on PE0:
  \[ R_0 = 10 + 3 \times 10 \quad (\text{response time in isolation}) \]
  \[ R_1 = 10 + 3 \times 10 + 2 \times 10 = 60 \]
  \[ R_2 = 10 + 3 \times 10 + 2 \times 10 + 2 \times 10 = 80 \]
  \[ R_3 = 10 + 3 \times 10 + 2 \times 10 + 2 \times 10 + 0 = 80 \quad (\text{fixed-point}) \]

---

1 Altmeyer et al., RTNS 2015
The Global Picture

High-level Program → Code Generation

Code Generation → Static Mapping/Scheduling

Static Mapping/Scheduling → Tasks + Dependencies

Tasks + Dependencies → Static Mapping/Scheduling

Static Mapping/Scheduling → Mapping

Mapping → Execution Order

Execution Order → Release Dates

Release Dates → Binary Generation

Binary Generation → Executable Binary

Timing models (static analysis) → Local WCRT Analysis

Local WCRT Analysis → Tasks WCRT + WC Access

Tasks WCRT + WC Access → WCRT with Interferences

WCRT with Interferences → Probabilistic Models