Performance Analysis of Reconfigurations in Adaptive Real-Time Streaming Applications

Jun Zhu, Ingo Sander, and Axel Jantsch
{junz, ingo, axel}@kth.se

Royal Institute of Technology (KTH), Stockholm, Sweden

October 9, 2009

Published in ESTIMedia '08.
Outline

1. Motivation
2. Related Work
3. Constraint based Analysis
4. Experimental Results
5. Conclusion
Synchronous data flow preliminaries

**Figure:** An example streaming application in synchronous data flow (SDF) model.

- Process, channel, FIFO, and the "synchronous" name
- Computation \( T = [t_{C,i}, t_{C,j}, t_{C,k}] \) and storage \( \Gamma = [\gamma_{i,j}, \gamma_{j,k}] \)
Figure: An adaptive application model (multiple configurations for $p_i$).

- The re-configuration time for $p_j$ is $t_{R,j}$ (mode dependent).
- A peak/avg. input $\rho_{in}$ and output sustains an avg. $\rho_{out}$.
- The worst case interval $t_{\text{inter}R,j}$ between two consecutive re-configurations of $p_j$. 

---

Performance Analysis of Reconfigurations in Adaptive Real-Time Streaming Applications
The total design cost (area in LE) of the RTR is:

$$A = A_{CC} + \sum_{i=1}^{i=n} A_{M,i} + k_C \cdot \max(A_{M,1}, \ldots, A_{M,n}) + A_{Buffer}$$  \hspace{1cm} (1)
Reconfiguration Scheduling & Problem Statement

(a) Example application (\( p_j \) has two working modes A and B)

(b) A schedule with reconfigurations

Problem Statement: To dimension the minimal buffer requirement \( A_{Buffer} \) or design cost in configuration analysis.
Related Work

- Buffer dimensioning for cyclo-static data flow models [Wiggers et al. DAC '07; Stuijk et al. Trans. Comput. '08]
- Buffer capacities in data dependent task graph (consumption rate changes, but no reconfigurations) [Wiggers et al. DATE ’08]
- Real-time calculus [Chakraborty et al. DATE ’03] and SymTA/S [Richter et al. DAC ’02]
- Model change protocols for adaptive systems [Real and Crespo, Real Time Syst. ’04; Shin et al. DAC ’00]
Cumulative functions [Cruz1995]:
Arrival function $R_{i,j}(t) = \sum_{0}^{t} s_{1}$
Service function $C_{i,j}(t) = \sum_{0}^{t} s_{2}$
Output function $R'_{i,j}(t) = R_{i,j}(t)$
Demand function

$$D_{i,j}(t) = \begin{cases} \sum_{0}^{t} s_{1} + n_{i,j} \\ \sum_{0}^{t} s_{1} \end{cases}$$

Buffer Properties:
Buffer backlog $B_{i,j}(t)$
Buffer requirement $B'_{i,j}(t)$
Execution Semantics in Scheduling

**Constraint**

*(Token ratios)* For process $p_j$, the $R'_{j,k}(t)$ and $C_{i,j}(t)$ follow the static input/output tokens ratio.

$$R'_{j,k}(t) \cdot m_{i,j} = C_{i,j}(t) \cdot n_{j,k} \quad (2)$$

**Constraint**

*(Asynchronous buffer)* The incoming tokens in buffer FIFO$_{i,j}$ takes at least $t_{C,j}$ slots to be served by process $p_j$.

$$R_{i,j}(t) - C_{i,j}(t + \Delta_t) \geq 0, \quad \forall \Delta_t \in [1, t_{C,j}] \quad (3)$$

More constraints (computation and storage) are in the paper.
Throughput on Demand Constraints

Constraint

(Application output throughput) After some start-up time period $\tau_0$ ($\tau_0 > 0$) with no stable output tokens, a specified output throughput $\rho_{out}$ should be sustained at the application sink process $p_k$.

$$C_k(\tau_0 + c \cdot L_{period}) = \rho_{out} \cdot c \cdot L_{period}, \quad \forall c \in \mathbb{N}_0. \quad (4)$$

Empirically, the length periodic phase is specified as

$L_{period} = q \cdot \left\lceil \frac{r_k}{\rho_{out}} \right\rceil, q \in \mathbb{N}\backslash\{\infty\}$, in which $q$ is incremental and leads to an valid $L_{period}$
Throughput Guarantees

Application throughput is guaranteed by periodic phases.

**Constraint**

*(Periodic phase)* The repeated process and FIFO status at time tag \( t' \) and \( t' + L_{\text{period}} \) determines a periodic phase between them with length \( L_{\text{period}} \).

\[
B'_{i,j}(t') = B'_{i,j}(t' + L_{\text{period}}), \quad \forall \text{FIFO}_{i,j} \tag{5}
\]

\[
W'_i(t') = W'_i(t' + L_{\text{period}}), \quad \forall p_i \in P \tag{6}
\]

where

\[
W'_i(t') = \sum_{k=1}^{t_{C,i}} k \cdot C_i(t' + k)
\]

in which variables \( W'_i(t') \) and \( W'_i(t' + L_{\text{period}}) \) are process status (denoted as numbers for each process in schedules).
Timing Phases of Reconfiguration Analysis

- **Prologue**: is the start-up phase with no throughput guarantees.

- **Periodic phase A(B)**: are phases with guaranteed application throughput (can be distinct for different working modes). The length $L_{period}$ is throughput relevant.

- **Reconfiguration phase**: consists of a period working in mode A, during which the reconfiguration starting time tag $t'$ is explored (optimized) upon the reconfiguration request. The reconfiguration stall takes $t_{R,j}$. The length is (worst case) $L_{period} + t_{R,j}$.

- **Transient phase**: has a length $\tau_1$, in which throughput is met but no periodic properties in scheduling yet.

We adopt the **worst case** sizes for each buffer in (colored) iterative phases.
Experimental Results - Example

Reconfiguration analysis framework implemented on Gecode solver.

The required throughput $\rho_{out}$ is guaranteed in two scenarios:

1. We assume different design options have varying $t_{R,j}$, but fixed $t_{C,j}$.
2. We choose design options conform to $t_{C,j} \propto \lceil \frac{1}{\sqrt{t_{R,j}}} \rceil$, which shows different implementation strategies in the speed and area trade-offs.

<table>
<thead>
<tr>
<th>options</th>
<th>#1</th>
<th>#2</th>
<th>#3</th>
<th>#4</th>
<th>#5</th>
<th>#6</th>
<th>#7</th>
<th>#8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_M$</td>
<td>0.5</td>
<td>0.8</td>
<td>1.0</td>
<td>1.3</td>
<td>2.5</td>
<td>3.8</td>
<td>5.0</td>
<td>10</td>
</tr>
<tr>
<td>$t_{R,i}$</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>40</td>
</tr>
<tr>
<td>$t_{C,i}$</td>
<td>16</td>
<td>13</td>
<td>12</td>
<td>10</td>
<td>8</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

Table: Different design options for the adaptive process $p_j$
Experimental Results - Example (contd.)

**Fig. Varying** $t_{R,j}$ & fixed $t_{C,j}$

**Fig. Varying** $t_{R,j}$ & $t_{C,j}$
Experimental Results - Industrial Application

Figure: Adaptive coding and modulation application synopsis from Thales.

Figure: Abstract model of industrial application (with disjoint logic FIFOs in the dashed box).
Design cost of RTR adaptation on the Cipher.
We present a constraint based performance analysis framework for adaptive real-time streaming applications.

The experimental results show that our framework suits well reconfigurations analysis and design trade-offs analysis.

Especially, the industrial case study illustrates the capability of our methodology to cope with the sequential composition of adaptive systems.
Problem Statement: Scheduling on multi-processors with resource constraints (NP-complete [Garey and Johnson, '79]).

Buffer minimization of real-time streaming applications scheduling on hybrid CPU/FPGA architectures.
DATE '09.
Extensions - Scheduling on NoC based MPSoC

Figure: Three phases to route a packet in inter-tile channel $ch_{i,j}$.

- Contention-free flow control in NoC with optimized (throughput and buffer cost) computation and communication scheduling.
- To avoid the scheduling overhead in TDMA-like heuristic schemes.


Constrained global scheduling of streaming applications on MPSoCs.

ASP-DAC ’10.
Future Work

To prototype streaming applications on MPSoCs using FPGAs:

- To verify our scheduling policies (compared with TDMA schemes).
- To consider different design options in implementation (buffer packing, etc).
Thanks for your attention!

Questions?