Generalized Extraction of Real-Time Parameters for Homogeneous Synchronous Dataflow Graphs

Hazem Ismail Ali¹  Benny Akesson²  Luís Miguel Pinho¹

¹CISTER Research Centre/INESC-TEC, Polytechnic Institute of Porto, Portugal
²Czech Technical University in Prague, The Czech Republic

{haali, lmp}@isep.ipp.pt ¹, kessoben@fel.cvut.cz ²

PDP 2015
Turku, Finland
March 4-6, 2015

March 6, 2015
Many multi-core systems have both streaming applications and traditional real-time applications.

The dataflow computational model is suitable for streaming applications because:

1. it enables the use of multi-core systems (parallelization model).
2. it is a natural paradigm for representing them.

A dataflow model is specified by a directed graph, where the nodes are considered as actors and the edges between the nodes as channels of data.
Homogeneous Synchronous Dataflow (HSDF)

- Is a special case of dataflow graphs in which all rates associated with actor ports are equal to 1.
- When each actor is fired once, the distribution of tokens on all channels return to their initial state (graph iteration).
- Other models can be converted to an equivalent HSDF using a conversion algorithm.

Figure: An example HSDF graph.
To enable real-time scheduling techniques on such mixed systems, a *unified model* is required to represent both types of applications running on the system.

**Problem to be addressed:**

*How to Extract Timing Parameters* of real-time streaming applications modeled as HSDF Directed Cyclic Graphs (DCG)?

Existing work is restricted to dataflow applications represented as acyclic applications.
We propose an algorithm for extracting timing parameters \((s_i, C_i, T_i, D_i)\) of HSDF actors, where:

- **\(s_i\)** offset (starting time).
- **\(C_i\)** Worst Case Execution Time (WCET) (Given by the application).
- **\(T_i\)** period.
- **\(D_i\)** relative deadline.

Enables applying traditional real-time schedulers and analysis techniques on HSDF.
System Model

Applications

Dataflow (DF)
(WCET, P/C rates, \( \zeta \))
represented in SDF, CSDF, ... etc.

Non-Dataflow (NDF)
\((s_i, C_i, T_i, D_i)\)

Transformation to HSDF

\[ C_i = \text{WCET}, \ T_i = \frac{1}{\zeta} \]
\[ D_i \text{ and } s_i \text{ calculated by the proposed algorithm} \]

Nodes/Actors

\((s_i, C_i, T_i, D_i)\)

Extract Real-Time Parameters

Arbitrary Deadline Tasks \((s_i, C_i, T_i, D_i)\)

Enable Mapping and Scheduling on Multi-Many-Core Platform
The proposed algorithm extracts the timing parameters \((s_i, C_i, T_i, D_i)\) of dataflow applications with timing constraints at design time.

**Algorithm**

It consists of two main phases:

1. Finding all the possible paths in the applications graph.
2. Extracting the timing parameters of individual actors in the graph using the output information of the previous phase.
Algorithm
Definitions

Path:
A route between two actors $v_x$ and $v_y$ with a latency constraint $D_{xy}$.

Path Sensitivity $\gamma$:
Criticality of a path with respect to path density. The path density is the tightness of the latency constraint $D_{xy}$ for a path $P$ compared to its execution time.

$$\gamma = \sum_{\forall v_j \in P} \frac{C_j}{D_{xy}}$$
We consider two well-known methods for pipelines (Paths):

1) The NORM method

divide the end-to-end deadline $D_{xy}$ of a pipeline proportionally to the computation time of its tasks:

$$D_i = \frac{C_i}{\sum_{\forall j \in P} C_j} \cdot D_{xy}$$

2) The PURE method

distribution of the laxity $\varepsilon = D_{xy} - \sum_{\forall j \in P} C_j$, equally among all tasks of the pipeline, such that each task have equal slack $\delta = \frac{\varepsilon}{|V_p|}$:

$$D_i = C_i + \delta$$
Deriving cycle latency constraints:

HSDF applications can have several cycles. Each cycle requires a latency constraint $D_{xy}^{cycle}$ that satisfies the throughput requirement $\zeta_i$ of the application:

- A quick choice for $D_{xy}^{cycle} = T_i = \frac{1}{\zeta_i}$.

- A better choice of $D_{xy}^{cycle}$ considers the number of tokens involved in this cycle $d_{cycle}$, to relax $D_{xy}^{cycle}$ and enable capturing overlapping iterations.

$$D_{xy}^{cycle} = \frac{d_{cycle}}{\zeta_i}$$

Refer to Section IV.B in our paper for more details.
Deriving end-to-end latency constraint:

In case of an HSDF application without a specified end-to-end latency constraint $D_{xy}$, is defined as:

$$D_{xy} = \max \left\{ T_i, \beta \cdot \sum_{\forall i \in CP} C_i \right\}$$

where $\beta$ is a constant that ranges $[1, \infty)$.

$$\beta = \frac{1}{\max_{\forall \text{cycle} \in G} \{ \gamma_{\text{cycle}} \}}$$

Refer to Section IV.B in our paper for more details.
First phase: Finding all possible paths

1) Creation of source and sink actors:

(a) An example HSDF graph.

(b) Adding source s and sink t.

- Unifies all the paths that traverse the graph from the input to the output of the graph have a uniform form that starts with s and end with t.
- Allows to deal with multiple input/output graphs.
First phase: Finding all possible paths

2) Path enumeration:

Partial Path: \[ P_i = \langle v_x, \ldots, v_j \rangle \]

Extend Partial Path using \( \text{Succ}(v_j) = (v_{j_1}, v_{j_2}, v_{j_3}, \ldots, v_{j_l}) \)

Resulting Paths:
\[ P_{i_1} = \langle v_x, \ldots, v_j, v_{j_1} \rangle \]
\[ P_{i_2} = \langle v_x, \ldots, v_j, v_{j_2} \rangle \]
\[ \vdots \]
\[ P_{i_l} = \langle v_x, \ldots, v_j, v_{j_l} \rangle \]

Finds all timed-constrained paths and orders them (\textit{descendingly}) according to sensitivity \( \gamma \).
Second phase: Extracting timing parameters

Algorithm

The second phase repeats for each application. It do the following:

1. Picks a time-constrained path \( P_i \) in order of sensitivity.
2. Each selected path \( P_i \) is assigned deadlines \( D_j \) and offsets \( s_j \).
Example

**HSDF graph example:**

![HSDF graph](image)

\[ \zeta = 0.5 \]
\[ C_a = C_b = C_c = C_d = C_e = C_f = 1 \]
\[ D_{ed} = 3 \quad D_{ad} = ? \]

**Sol: Algorithm First Phase:**

*We have three paths:*

\[ P_1 = \langle e, f, d \rangle, \quad D^1_{ed} = 3, \quad \gamma_1 = 1 \]
\[ P_2 = \langle b, c \rangle, \quad D^2_{bc} = ? \]
\[ P_3 = \langle a, b, c, d \rangle, \quad D^3_{ad} = ? \]

**Sol: Deriving Latency constraints:**

\[ D^2_{bc} = \frac{d_{cycle}}{\zeta} = \frac{2}{0.5} = 4, \quad \gamma_2 = 0.5 \]
\[ D^3_{ad} = \max \{ T_i, \beta \cdot \sum_{\forall v_i \in P_3} C_i \} = \max \{ 2, \frac{1}{\gamma_2} \cdot 4 \} = 8, \quad \gamma_3 = 0.5 \]

Therefore, \( P = \langle P_1, \gamma_1 \rangle, \langle P_2, \gamma_2 \rangle, \langle P_3, \gamma_3 \rangle \) = \{ \langle (e, f, d), 1 \rangle, \langle (b, c), 0.5 \rangle, \langle (a, b, c, d), 0.5 \rangle \}
Example

**HSDF graph example:**

- $\zeta = 0.5$
- $C_a = C_b = C_c = C_d = C_e = C_f = 1$
- $D_{ed} = 3$, $D_{ad} = 8$

**Sol: Algorithm Second Phase:**

**Individual deadline calculation:**

- $P_1$: $D_e = 1$, $D_f = 1$, $D_d = 1$
- $P_2$: $D_b = 2$, $D_c = 2$
- $P_3$: $D_a = 3$

**Offset assignment:**

$\hat{\mathcal{P}} = \{ \langle P_3, D_{ad}^3 \rangle, \langle P_1, D_{ed}^1 \rangle \}$

- $P_3$: $s_a = 0$, $s_b = 3$, $s_c = 5$, $s_d = 7$
- $P_1$: $s_e = 5$, $s_f = 6$
Therefore:
\[
\{a, b, c, d, e, f\} = \{(0, 1, 2, 3), (3, 1, 2, 2), (5, 1, 2, 2), (7, 1, 2, 1), (5, 1, 2, 1), (6, 1, 2, 1)\}
\]

**HSDF graph example:**

\[
\begin{align*}
\zeta &= 0.5 \\
C_a &= C_b = C_c = C_d = C_e = C_f = 1 \\
D_{ed} &= 3 \\
D_{ad} &= 8
\end{align*}
\]
Through formal proofs (*refer to Section V in the paper*), we assure that the assigned timing parameters by our proposed algorithm guarantees satisfying application timing constraint using any known real-time scheduling algorithm.
The main contribution is that the HSDF graphs can be cyclic or acyclic and the graph actors are modelled as arbitrary-deadline tasks.

We formally proved that the assigned timing parameters satisfies the timing constraints of the application.

It enables applying traditional real-time analysis techniques on dataflow graphs follows from representing as tasks.

A method to assign individual deadlines and offsets for real-time dataflow actors and support for two deadline assignment techniques (NORM/PURE) that are widely used in the literature.
Questions ?