Chapter 7

Hard Real-time Scheduling of Adaptive Streaming Applications

THE initial scheduling framework in the Daedalus^{RT} design flow considers the CSDF MoC as the analysis model and the PPN MoC as the implementation model. For adaptive streaming applications, we have proposed P^{3}N as the implementation model in Chapter 6. However, an analysis MoC for adaptive streaming applications is still missing in Daedalus^{RT}. More importantly, we need proper operational semantics for such a MoC that potentially allows adaptive execution of the MoC and easy HRT analysis. In this chapter, we propose a new analysis MoC in the Daedalus^{RT} design flow that models adaptive streaming applications.

There already exist some adaptive MoCs in literature [89, 114, 129]. Unfortunately, each of them has certain drawback that does not fulfill our needs. For instance, we would like to explicitly have the notion of *mode* in an adaptive MoC. A mode of an adaptive MoC is essentially a static MoC, e.g., the CSDF MoC, when the values of all dynamic parameters are fixed. As a result, the existing HRT analysis developed in Daedalus^{RT} for the CSDF MoC can be reused. For the adaptive MoCs shown in Figure 1.6 on page 10, parameterized CSDF and VPDF MoCs are thus excluded from our consideration because they do not have the notion of mode. At the same time, the expressiveness of MCDF is too restricted.

Furthermore, support for the HRT scheduling and the associated analysis is limited in the existing MoCs, especially during mode transitions. In particular, we wish to have a composable analysis for mode transitions. That is, the analysis of any mode transition is independent from the mode transitions occurred in past. This composable analysis will significantly reduce the complexity of the analysis, as the complexity merely depends on the number of allowed transitions. This is crucial for applications with a large number of modes and possible transitions. As a by-product

Notation	Meaning
c	a computation
Δ	transition delay
L	iteration latency
p	a dynamic parameter
Π	a set of parameter vectors defined in Definition 7.2.1
ψ	parameters used for actors defined in Definitions 7.2.4 and 7.2.5
x	Maximum-Overlap Offset (MOO)

Table 7.1: Additional notations used in Chapter 7 besides the ones introduced in Chapter 2.

of this composable analysis, the implementation efficiency of such a HRT system to support adaptive behavior will be much higher. No complex calculation is needed at run-time, as most of parameters (see Section 7.3) can be computed at compile-time.

Based on the discussion above, we develop a new MoC, Mode-Aware Data Flow (MADF), in this chapter that has the advantages of SADF and VPDF. Inspired by SADF, we characterize the adaptive application behavior with individual modes[1] (see Definition 7.2.7) and transitions (see Definition 7.2.11) between them. Similar to VPDF, the length of production/consumption sequences for an actor varies from one mode to another. The length is only fixed when the mode is known. Based on the clear distinction between modes and transitions, we define operational semantics, in particular a novel transition protocol, to avoid timing interference between modes and transitions. As a result, our HRT analysis is simpler than the state-of-the-art timing analysis [47]. To ease discussion, we use additional notations listed in Table 7.1 besides the ones introduced in Chapter 2.

Scope of Work

We assume that an adaptive streaming application does not have cyclic data dependences. The considered MPSoC platforms in this chapter are homogeneous, i.e., they may contain multiple, but the same type of programmable PEs with distributed memories. Moreover, the platform must be predictable, which means timing guarantees are provided on the response time of hardware components and OS schedulers. The precision-timed (PRET) [79] platform is such an example. On the software side, we assume partitioned scheduling algorithms, i.e, no migration of actors between PEs is allowed. The considered scheduling algorithms on each PE include Fixed-Priority Pre-emptive Scheduling (FPPS) algorithms, such as RM [80], or dynamic

[1]"Scenario" for SADF is equivalent to "mode" in our case.

scheduling algorithms, such as EDF [80].

7.1 Related Work

For FSM-SADF [47], the authors proposed an approach to compute worst-case performance among all mode transitions, assuming the self-timed transition protocol (explained later in Section 7.2.3). Although it is an exact analysis, the approach has inherently exponential time-complexity. Moreover, the approach leads to timing interference between modes upon mode transitions, which makes this approach not applicable for our problem. In contrast, our approach does not introduce interference between modes due to the novel transition protocol proposed in Section 7.2.3. The timing behavior of individual modes and during mode transitions can be analyzed independently. In addition, our approach considers allocation of actors on PEs, which by itself is a harder problem than the one in [47]. In [48], the authors proposed to model scenario transitions in a single FSM. Delays due to scenario reconfiguration are given and explicitly modeled in the FSM. The problem addressed in this chapter is different as we aim at deriving such a delay.

In [45], the author proposes to use a linear model to capture worst-case transition delay and period during scenario transitions of FSM-SADF. Our Maximum-Overlap Offset (MOO, see Section 7.2.3) transition protocol is conceptually very similar to the linear model. However, we obtain the linear model in a different way, specifically simplified for the adopted hard real-time scheduling framework. For instance, finding a reference schedule is not necessary in our case, but being crucial in the tightness of the analysis proposed in [45]. Moreover, our approach solves the problem of changing graph structure during mode transitions, which was not studied in [45].

For VPDF [129], the analysis has been limited to computing buffer sizes under throughput constraints so far. The execution of a VPDF graph on MPSoC platforms under HRT constraints has not been studied. In particular, the allocation of actors and how to switch from one mode to another one are not discussed. Moreover, delay due to mode transitions has not been investigated. Our approach, on the other hand, takes these important factors into account. Therefore, our analysis results are directly reflected in a real implementation.

Mode-controlled data flow (MCDF) [89] is another adaptive MoC whose properties can be partly analyzed at compile-time. The MCDF MoC primarily focuses on SDR applications, where different sub-graphs need to be active in different modes. This is achieved by using *switch* and *select* actors. The author implicitly assumes self-timed scheduling during mode transitions. Based on this assumption, a worst-case timing analysis is developed. Similar to the case of SADF, use of the self-timed scheduling introduces timing interference between modes. As a consequence, the analysis must take into account the sequence of mode transitions of interest. Although the author provides an upper bound of timing behavior for a parameterized sequence of mode transitions, the accuracy is still unknown. In contrast, our approach results in a timing analysis of mode transitions that is independent from already occurred transitions. Moreover, the analysis results are directly reflected in the final implementation. In this sense, our analysis is exact in the timing behavior of mode transitions.

In [93], an analysis is proposed to reason about worst-case response time of a task graph in case of mode change. However, the task graph has very limited expressiveness and is not able to model adaptive application behavior. In this chapter, we define a more expressive MoC that is amenable to adaptive application behavior.

In [104, 108], the authors focus on timing analysis for mode changes of real-time tasks. The starting times of new mode tasks need to be delayed to avoid overloading PEs. The algorithms to compute the starting times were provided. Both works are related to ours because actors allocated on the same PE may also overload the PE after mode transitions. In this case, the starting times of actors in the new mode need to be delayed. In [104, 108], it was assumed that tasks are independent. The proposed algorithms are thus not applicable to the adaptive MoCs, since the starting times of actors in the adaptive MoCs depend on each other due to data dependencies. Moreover, the algorithms in [104, 108] involve high computational complexity because fixed-point equations must be solved at every step in the algorithms.

7.2 Model Definition

7.2.1 Mode-Aware Data Flow (MADF)

Definition 7.2.1 (Mode-Aware Data Flow (MADF)). A Mode-Aware Data Flow (MADF) is a multi-graph defined by a tuple $(\mathcal{A}, A_c, \mathcal{E}, \Pi)$, where

- $\mathcal{A} = \{A_1, \ldots, A_{|\mathcal{A}|}\}$ is a set of dataflow actors;
- A_c is the control actor to determine modes and their transitions;
- $\mathcal{E}$ is the set of edges for data/parameter transfer;
- $\Pi = \{\vec{p}_1, \ldots, \vec{p}_{|\mathcal{A}|}\}$ is the set of parameter vectors, where each $\vec{p}_i \in \Pi$ is associated with a dataflow actor A_i.

Throughout this section, we use graph G_1 shown in Figure 7.1 as the running example to illustrate the definition of MADF and the hard real-time scheduling analysis related to MADF. For G_1, $\mathcal{A} = \{A_1, A_2, A_3, A_4, A_5\}$ is the set of dataflow

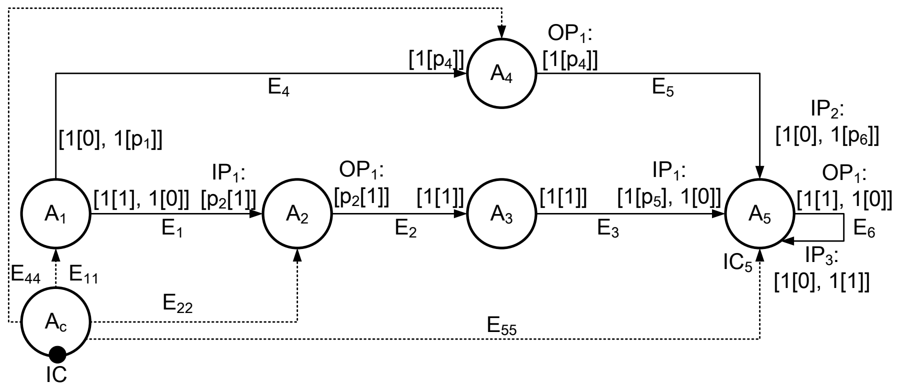


Figure 7.1: An example of MADF graph (G_1).

actors. A_c is the control actor. $\mathcal{E} = \{E_1, E_2, E_3, E_4, E_5, E_6, E_{11}, E_{22}, E_{44}, E_{55}\}$ is the set of edges. For actor A_5, $\vec{p}_5 = [p_5, p_6]$ is the parameter vector. The input port IP_1 of actor A_5 has a consumption sequence $[1[p_5], 1[0]]$, which can be interpreted as $[p_5, 0]$.

Definition 7.2.2 (Dataflow Actor). A dataflow actor A_i is described by a tuple $(\mathcal{I}_i,\ IC_i,\ \mathcal{O}_i, \mathcal{C}_i, M_i)$, where

- $\mathcal{I}_i = \{IP_1, \ldots, IP_{|\mathcal{I}_i|}\}$ is the set of data input ports of actor A_i;
- IC_i is the control input port that reads parameter vector $\vec{p}_i$ for actor A_i;
- $\mathcal{O}_i = \{OP_1, \ldots, OP_{|\mathcal{O}_i|}\}$ is the set of data output ports of actor A_i;
- $\mathcal{C}_i = \{c_1, \ldots, c_{|C|}\}$ is the set of computations. When actor A_i fires, it performs a computation $c_k \in \mathcal{C}_i$;
- $M_i \ :\ \vec{p}_i \rightarrow \{\phi, \bar{C}_i\}$ is a mapping relation, where $\vec{p}_i \in \Pi$, $\phi \in \mathbb{N}^+$, and $\bar{C}_i \subseteq \mathcal{C}_i$ is a sequence of computations $[\bar{C}_i(1), \ldots, \bar{C}_i(k), \ldots, \bar{C}_i(\phi)]$ with $\bar{C}_i(k) \in \mathcal{C}_i, 1 \leq k \leq \phi$.

Actor A_2 in Figure 7.1 has a set of one input port $\mathcal{I}_2 = \{IP_1\}$, a set of one output port $\mathcal{O}_2 = \{OP_1\}$ as well as a control input port IC_2. A set of computations $\mathcal{C}_2 = \{c_1, c_2, c_3\}$ is associated with A_2. The mapping relation M_2 is given in Table 7.2. It can be interpreted as follows: If $p_2 = 2$, actor A_2 repetitively performs computations according to sequence $\bar{C}_2 = [c_1, c_2]$ every time when firing A_2. When $p_2 = 1$, firing A_2 performs computation c_3.

Table 7.2: Mapping relation M_2 for actor A_2 in Figure 7.1.

$\vec{p}_2 = [p_2]$	ϕ	$\bar{C}_2$
2	2	$[c_1, c_2]$
1	1	$[c_3]$

Table 7.3: Function MC_5 defined for actor A_5 in Figure 7.1.

$\mathcal{S}$	$\mathbb{N}^2$
SI^1	$[2,0]$
SI^2	$[1,1]$

Definition 7.2.3 (Control Actor). The control actor A_c is described by a tuple $(IC, \mathcal{O}_c, \mathcal{S}, \mathcal{M}_c)$, where

- $\mathcal{S} = \{SI^1, \ldots, SI^{|\mathcal{S}|}\}$ is a set of mode identifiers, each of which specifies a unique mode;
- IC is the control input port which is connected to the external environment. Mode identifiers are read through the control input port from the environment;
- $\mathcal{O}_c = \{OC_1, \ldots, OC_{|\mathcal{A}|}\}$ is a set of control output ports. Parameter vector $\vec{p}_i$ is sent through $OC_i \in \mathcal{O}_c$ to actor A_i;
- $\mathcal{M}_c = \{MC_1, \ldots, MC_{|\mathcal{A}|}\}$ is a set of functions defined for each actor $A_i \in \mathcal{A}$. For each $MC_i \in \mathcal{M}_c$, $MC_i : \mathcal{S} \to \mathbb{N}^{|\vec{p}_i|}$ is a function that takes a mode identifier and outputs a vector of non-negative integer values.

For G_1 in Figure 7.1, we have two mode identifiers $\mathcal{S} = \{SI^1, SI^2\}$. At run-time, control actor A_c reads these mode identifiers through control port IC (black dot in Figure 7.1). For actor A_5, $MC_5 \in \mathcal{M}_c$ is given in Table 7.3. As explained previously, the parameter vector for actor A_5 is $\vec{p}_5 = [p_5, p_6]$. Therefore, MC_5 takes a mode identifier and outputs a 2-dimensional vector as shown in the second column in Table 7.3. For instance, mode SI^1 results in a non-negative integer vector $[2,0]$.

To further define production/consumption sequences with variable length, we use the notation $n[m]$ for a sequence of n elements with integer value m, i.e.,

$$n[m] = [\overbrace{m, \ldots, m}^{n \text{ times}}]$$

Definition 7.2.4 (Input Port). An input port *IP* of an actor is described by a tuple (CNS, M_{IP}), where

- $CNS = [\phi_1[cns_1], \ldots, \phi_K[cns_K]]$ is the consumption sequence with ϕ phases, where $\phi = \sum_{i=1}^{K} \phi_i$ is determined by the mapping relation M in Definition 7.2.2, and $cns_1, \ldots, cns_K \in \mathbb{N}$;
- $M_{IP} : \vec{p}_i \rightarrow \psi_{IP}$ is a mapping relation, where $\vec{p}_i \in \Pi$ and

$$\psi_{IP} = \{\phi_1, \ldots, \phi_K, cns_1, \ldots, cns_K\}. \tag{7.1}$$

Definition 7.2.5 (Output Port). An output port *OP* of an actor is described by a tuple (PRD, M_{OP}), where

- $PRD = [\phi_1[prd_1], \ldots, \phi_K[prd_K]]$ is the production sequence with ϕ phases, where $\phi = \sum_{i=1}^{K} \phi_i$ is determined by the mapping relation M in Definition 7.2.2, and $prd_1, \ldots, prd_K \in \mathbb{N}$.
- $M_{OP} : \vec{p}_i \rightarrow \psi_{OP}$ is mapping relation, where $\vec{p}_i \in \Pi$ and

$$\psi_{OP} = \{\phi_1, \ldots, \phi_K, prd_1, \ldots, prd_K\}. \tag{7.2}$$

The consumption/production sequence defined here is a generalization of that for the CSDF MoC (see Section 2.2.2 on page 32). We can see that a CSDF actor has a constant ϕ phases in its consumption/production sequences, whereas the length of the phase of an MADF actor is parameterized by $\phi = \sum_{i=1}^{K} \phi_i$. In addition, the mapping relation M_{IP}/M_{OP} must be provided by the application designer. Consider the two input ports IP_1 and IP_2 of actor A_5 in Figure 7.1. The mapping relations M_{IP_1} and M_{IP_2} are represented as follows:

$$M_{IP_1} : \vec{p}_5 = [p_5, p_6] \rightarrow \psi_{IP_1} = \{\phi_1, \phi_2, cns_1, cns_2\} = \{1, 1, p_5, 0\}, \tag{7.3}$$
$$M_{IP_2} : \vec{p}_5 = [p_5, p_6] \rightarrow \psi_{IP_2} = \{\phi_1, \phi_2, cns_1, cns_2\} = \{1, 1, 0, p_6\}. \tag{7.4}$$

It can be seen that parameter p_5 is mapped to cns_1 of IP_1, parameter p_6 is mapped to cns_2 of IP_2, and ϕ_1 and ϕ_2 both are constant equal to 1. Therefore, the consumption sequence of IP_1 is $CNS = [1[p_5], 1[0]]$ and the consumption sequence of IP_2 is $CNS = [1[0], 1[p_6]]$. Similarly considering output port OP_1 of actor A_4, its mapping relation M_{OP_1} is given as:

$$M_{OP_1} : \vec{p}_4 = [p_4] \rightarrow \psi_{OP_1} = \{\phi_1, prd_1\} = \{1, p_4\}. \tag{7.5}$$

In this case, parameter p_4 is mapped to prd_1 and $\phi_1 = 1$. Therefore, production sequence $PRD = [1[p_4]]$ is obtained for OP_1 of A_4.

Definition 7.2.6 (Edge)**.** An edge $E \in \mathcal{E}$ is defined by a tuple

$$\Big((A_i, OP), (A_j, IP)\Big),$$

where

- actor A_i produces a parameterized number of tokens to edge E through output port OP;
- actor A_j consumes a parameterized number of tokens from E through input port IP.

Considering edge E_5 in Figure 7.1, it connects output port OP_1 of actor A_4 to input port IP_2 of actor A_5.

Definition 7.2.7 (Mode of MADF)**.** A mode SI^i of MADF is a live CSDF [30] graph, denoted as G^i, obtained by setting values of Π in Definition 7.2.1 as follows:

$$\forall k \in \Pi \; : \; \vec{p}_k = MC_k(SI^i), \tag{7.6}$$

where function MC_k is given in Definition 7.2.3.

Definition 7.2.8 (Mode of MADF Actor)**.** An actor A_k in mode SI^i, denoted by A_k^i, is a CSDF [30] actor obtained from A_k as follows:

$$\vec{p}_k = MC_k(SI^i). \tag{7.7}$$

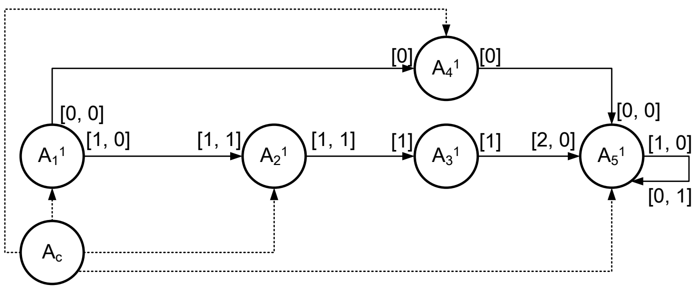
Figure 7.2(a) shows the CSDF graph in mode SI^1 and Figure 7.2(b) shows the CSDF graph in mode SI^2. Consider function MC_5 for actor A_5 in Table 7.3 with parameter vector $\vec{p}_5 = [p_5, p_6]$. For instance, mode SI^1 results in $\vec{p}_5 = [p_5, p_6] = [2, 0]$, where parameter values $p_5 = 2$ and $p_6 = 0$. Consequently, according to mapping relations M_{IP_1} and M_{IP_2} given in Equations 7.3 and 7.4, $cns_1 = p_5 = 2$ can be obtained for input port IP_1 and $cns_2 = p_6 = 0$ for IP_2. This determines actor A_5^1 shown in Figure 7.2(a) for mode SI^1.

Definition 7.2.9 (Inactive Actor)**.** An MADF actor A_i^k is inactive in mode SI^k if the following conditions hold:

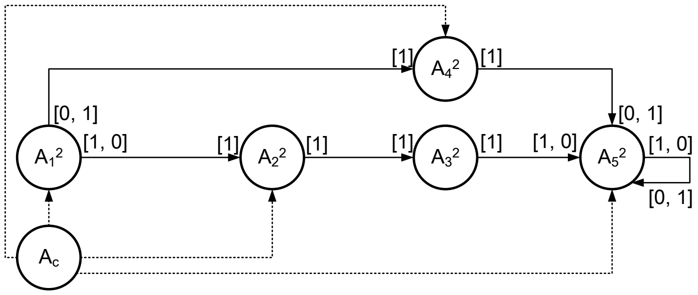
1. $\forall IP \in \mathcal{I}_i \; : \; CNS = [0, \ldots, 0]$;
2. $\forall OP \in \mathcal{O}_i \; : \; PRD = [0, \ldots, 0]$.

Otherwise, A_i^k is called *active* in mode SI^k.

For actor A_4^1 shown in Figure 7.2(a), it has consumption and production sequence $[0]$. Therefore, actor A_4 is said to be inactive in mode SI^1.



(a) CSDF graph G_1^1 of mode SI^1.



(b) CSDF graph G_1^2 of mode SI^2.

Figure 7.2: Two modes of the MADF graph in Figure 7.1.

7.2.2 Operational Semantics

During execution of a MADF graph, it can be either in a steady-state or mode transition.

Definition 7.2.10 (Steady-state)**.** A MADF graph is in a steady-state of a mode SI^i, if it satisfies Equation (7.6) with the same SI^i for all its actors.

Definition 7.2.11 (Mode Transition)**.** A MADF graph is in a mode transition from mode SI^o to SI^l, where $o \neq l$, if some actors have SI^o for Equation (7.7) and the remaining active actors have SI^l for Equation (7.7).

In the steady-state of a MADF graph, all active actors execute in the same mode. As defined previously in Definition 7.2.7 and shown in Figure 7.2(a) and Figure 7.2(b), the steady-state of the MADF graph has the same operational semantics as a CSDF [30] graph. We use $\langle A_i^k, x\rangle$ to denote the xth firing of actor A_i in mode SI^k. At $\langle A_i^k, x\rangle$, it executes computation

$$\bar{C}_i\big((x-1) \mod \phi + 1\big),$$

where $\bar{C}_i$ is given in Definition 7.2.2. The number of tokens consumed and produced are specified according to Definitions 7.2.4 and 7.2.5, respectively. For instance, the xth firing of A_i^k produces the following number of tokens through an output port OP:

$$PRD\big((x-1) \mod \phi + 1\big).$$

In each mode SI^k, the MADF graph is a live CSDF graph and thus has the notion of graph iterations with a non-trivial repetition vector $\vec{q^k} \in \mathbb{N}^{|A|}$ resulting from Equation (2.14) on page 32. Next, we further define mode iterations.

Definition 7.2.12 (Mode Iteration)**.** One iteration It^k of a MADF graph in mode SI^k consists of one firing of control actor A_c and $q_i^k \in \vec{q^k}$ firings of each MADF actor A_i^k.

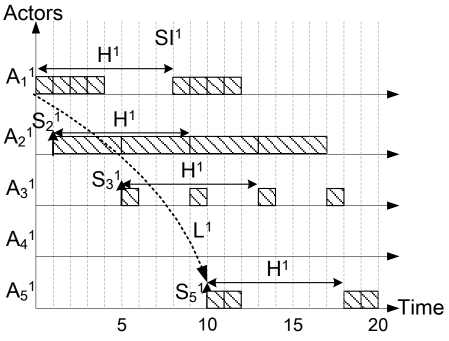
Consider the two modes shown in Figure 7.2(a) and Figure 7.2(b). Repetition vectors $\vec{q^1}$ and $\vec{q^2}$ are:

$$\begin{aligned}\vec{q^1} &= [4,2,2,0,2],\\ \vec{q^2} &= [2,1,1,1,2]. \end{aligned} \tag{7.8}$$

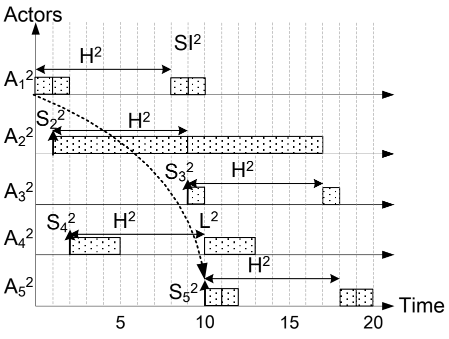
For any mode of a MADF graph, i.e., a live CSDF graph, under *any* valid schedule, it has (eventually) periodic execution in time. This holds for CSDF graphs under self-timed schedule [110], K-periodic schedule [31], and SPS [22]. The length of the periodic execution, called *iteration period*, determines the minimum time interval to complete one graph iteration (*cf.* Definition 7.2.12). The iteration period, denoted by H^k, is equal for any actor in the same mode SI^k. During a periodic execution, the starting time of each actor A_i^k, denoted by S_i^k, indicates the time distance between the start of source actor A_{src}^k and the start of actor A_i^k in the same iteration period. Based on the notion of starting times, we define *iteration latency* L^k of a MADF graph in mode SI^k as follows:

$$L^k = S_{\text{snk}}^k - S_{\text{src}}^k, \tag{7.9}$$

where S_{snk}^k and S_{src}^k are the earliest starting times of the sink and source actors, respectively. Figure 7.3 illustrates the execution of both modes SI^1 and SI^2 given in Figure 7.2 under the self-timed schedule. A rectangle denotes WCET of an actor firing. The WCETs of all actors in both modes are given in the third row of Table 7.4 on page 132. Now, it can be seen in Figure 7.3 that iteration period $H^1 = H^2 = 8$. Based on the starting time of each actor, we obtain iteration latencies $L^1 = S_5^1 - S_1^1 = 10 - 0 = 10$ and $L^2 = S_5^2 - S_1^2 = 10 - 0 = 10$ as shown in Figure 7.3.



(a) Mode SI^1 in Figure 7.2(a).



(b) Mode SI^2 in Figure 7.2(b).

Figure 7.3: Execution of two iterations of both modes SI^1 and SI^2 under self-timed scheduling.

7.2.3 Mode Transition

While the operational semantics of a MADF graph in steady-state are the same as that of a CSDF [30] graph, the transition of MADF graph from one mode to another is the crucial part that makes it fundamentally different from CSDF. The protocol for mode transitions has strong impact on the compile-time analyzability and implementation efficiency. In this section, we propose a novel and efficient protocol of mode transitions for MADF graphs.

During execution of a MADF graph, mode transitions may be triggered at run-time by receiving a Mode Change Request (MCR) from the external environment. We first assume that a MCR can be only accepted in the steady-state of a MADF graph, not in an ongoing mode transition. This means that any MCR occurred during an ongoing mode transition will be ignored. Consider a mode transition from SI^o to SI^l. The transition is accomplished by the control actor reading mode identifier SI^l from its control input port (see the black dot in Figure 7.1) and writing parameter values of $\vec{p}_i$ to the control output port connected to each dataflow actor A_i^l according to function MC_i given in Definition 7.2.3. Then, A_i^l reads new parameter values $\vec{p}_i$ from its control input port and sets the sequence of computations according to mapping relation M_i in Definition 7.2.2. The production and consumption sequences are obtained in accordance with M_{IP} and M_{OP} in Definition 7.2.4 and Definition 7.2.5, respectively. Similar to the P^{3}N MoC, we further define that mode transitions are only allowed at quiescent points [94].

Definition 7.2.13 (Quiescent Point of MADF)**.** For a transition from mode SI^o to SI^l, a quiescent point of a MADF actor A_i is a firing $\langle A_i^l, x\rangle$ in a mode iteration It^l

that satisfies

$$\neg\exists\langle A_i^l, y\rangle \in It^l \; : \; y < x. \tag{7.10}$$

Figure 7.4 shows an execution of G_1 in Figure 7.1 with two mode transitions. For instance, the MCR at time $t_{\mathrm{MCR1}} = 1$ denotes a transition request from mode SI^2 to SI^1. The mode transition of actor A_1 is only allowed at the quiescent point (time 2 in Figure 7.4) right before the first firing in mode iteration It^1 of mode SI^1.

Definition 7.2.13 defines mode transitions of MADF graphs as a partially ordered actor firings. However, it does not specify at which time instance a mode transition actually starts. Therefore, below, we focus on the transition protocol that defines the points in time for occurrences of mode transitions. To quantify the transition protocol, we introduce a metric, called *transition delay*, to measure the responsiveness of a protocol to a MCR.

Definition 7.2.14 (Transition Delay). For a MCR at time t_{MCR} calling for a mode transition from mode SI^o to SI^l, the transition delay $\Delta^{o\to l}$ of a MADF graph is defined as

$$\Delta^{o\to l} = \sigma_{\mathrm{snk}}^{o\to l} - t_{\mathrm{MCR}}, \tag{7.11}$$

where $\sigma_{\mathrm{snk}}^{o\to l}$ is the earliest starting time of the sink actor in the new mode SI^l.

In Figure 7.4, we can compute the transition delay for *MCR1* occurred at time $t_{\mathrm{MCR1}} = 1$ as $\Delta^{2\to 1} = 18 - 1 = 17$.

Self-timed (ST) Transition Protocol

In the existing adaptive MoCs like FSM-SADF [47], a protocol, referred here as *Self-Timed* (ST) transition protocol, is adopted. The ST protocol specifies that actors are scheduled in the self-timed manner not only in the steady-state, but also during a mode transition. For FSM-SADF upon a MCR, a firing of a FSM-SADF actor in the new mode can start immediately after the firing of the actor completes the old mode iteration. The only possible delay is introduced due to availability of input data. One reason behind the ST protocol is that the ST schedule for a (C)SDF graph (steady-state of FSM-SADF[2]) leads to its highest achievable throughput. However, the ST protocol generally introduces interference of one mode execution with another one. The time needed to complete mode transitions also fluctuate as the transition delay of an ongoing transition depends on the transitions occurred in the past. We consider this as an undesired effect because mode transitions using the ST protocol become potentially slow and unpredictable. Another consequence of the

[2]The steady-state of SADF is defined similarly to that of MADF. The only difference is that a scenario of FSM-SADF is a SDF graph, whereas a mode of MADF is a CSDF graph.

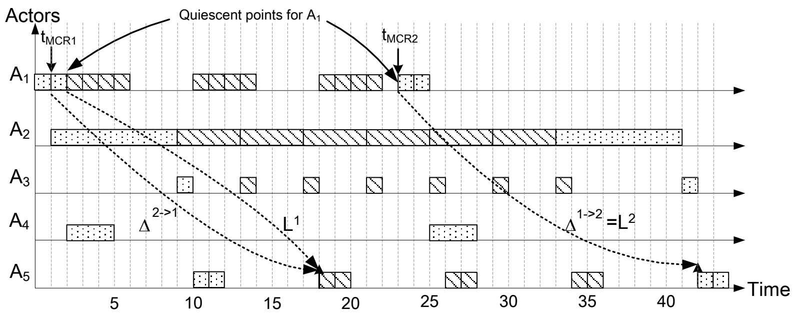


Figure 7.4: An execution of G_1 in Figure 7.1 with two mode transitions under the ST transition protocol. *MCR1* at time t_{MCR1} denotes a transition request from mode SI^2 to SI^1, and *MCR2* at time t_{MCR2} denotes a transition request from mode SI^1 to SI^2.

incurred interference between modes using the ST transition protocol is the high time complexity of analyzing transition delays, because transition delays cannot be analyzed independently for each mode transition. The analysis proposed in [47] uses an approach based on state-space exploration, which has the exponential time complexity.

Consider G_1 in Figure 7.1 and an execution of G_1 with the two mode transitions illustrated in Figure 7.4. The execution is assumed under the ST schedule for both steady-state and mode transitions of G_1. After *MCR1* at time t_{MCR1}, the transition from mode SI^2 to SI^1 introduces interference to execution of the new mode SI^1 from execution of the old mode SI^2. The interference increases the iteration latency of the new mode SI^1 to $L^1 = S_5^1 - S_1^1 = 18 - 2 = 16$ from initially 10 as shown in Figure 7.3(a) when G_1 is only executed in the steady-state of mode SI^1. Even worse, the interference is further propagated to the second mode transition after *MCR2* at time t_{MCR2}. In this case, the iteration latency $L^2 = S_5^2 - S_1^2 = 42 - 23 = 19$ is increased from initially 10 as shown in Figure 7.3(b) when G_1 is only executed in the steady-state of mode SI^2. This example thus clearly shows the problem of the ST protocol. That is, it introduces interference between the old and new modes due to mode transitions, thereby increasing the iteration latency of the new mode in the steady-state after the transition. Furthermore, the increase of iteration latency also potentially increases transition delays as it will be shown in the next section.

Maximum-Overlap Offset (MOO) Transition Protocol

To address the problem of the ST transition protocol explained above, we introduce in this chapter our new transition protocol, called *Maximum-Overlap Offset* (MOO).

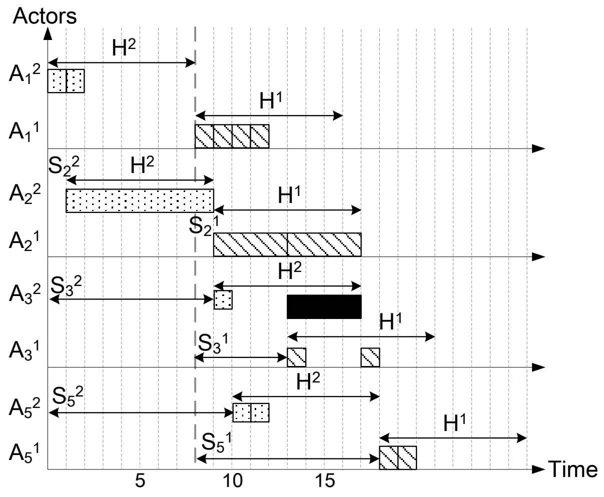


Figure 7.5: An illustration of the Maximum-Overlap Offset (MOO) calculation.

Definition 7.2.15 (Maximum-Overlap Offset (MOO)). For a MADF graph and a transition from mode SI^o to SI^l, Maximum-Overlap Offset (MOO), denoted by x, is defined as

$$x = \begin{cases} \max_{A_i \in \mathcal{A}^o \cap \mathcal{A}^l} (S_i^o - S_i^l) & \text{if } \max_{A_i \in \mathcal{A}^o \cap \mathcal{A}^l} (S_i^o - S_i^l) > 0 \\ 0 & \text{otherwise,} \end{cases} \tag{7.12}$$

where $\mathcal{A}^o \cap \mathcal{A}^l$ is set of actors active in both modes SI^o and SI^l.

Basically, we first assume that the new mode SI^l starts immediately after the source actor A^o_{src} of the old mode SI^o completes its last iteration It^o. All actors A_i^l of the new mode execute according to the earliest starting times S_i^l and iteration period H^l in the steady-state. Under this assumption, if the execution of the new mode overlaps with the execution of the old mode in terms of iteration periods H^o and H^l, we then need to offset the starting time of the new mode by the maximum overlap among all actors. In this way, the execution of the new mode will have the same iteration latency as that of the new mode in the steady-state, i.e., no interference between the execution of both old and new modes.

Consider *MCR1* at time t_{MCR1} shown in Figure 7.4. Obtaining MOO x is illustrated in Figure 7.5. We first assume that the new mode SI^1 starts at the time when the source actor A_1^2 completes the last iteration at time 8 (see bold, dashed line in Figure 7.5). Actors A_i^1 in the new mode start as if they executed in the steady-state of mode SI^1. Then, we can see that, for actor A_3, the execution of A_3^1 in

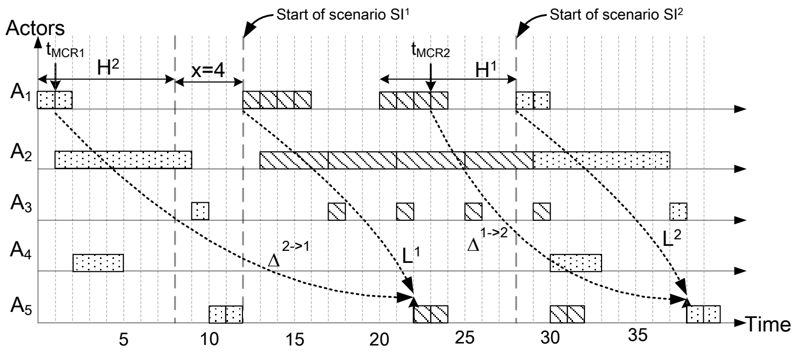


Figure 7.6: The execution of G_1 with two mode transitions under Maximum-Overlap Offset (MOO) protocol.

the new mode SI^1 according to S_3^1 in Figure 7.3(a) overlaps 4 time units (solid bar in Figure 7.5) with the execution of A_3^2 in the old mode SI^2 in terms of iteration periods H^2 and H^1. This is also the maximum overlap between the execution of actors in modes SI^2 and SI^1. According to Definition 7.2.15, x can be obtained through the following equations:

$$
\begin{aligned}
S_1^2 - S_1^1 &= 0 - 0,\\
S_2^2 - S_2^1 &= 1 - 1 = 0,\\
S_3^2 - S_3^1 &= 9 - 5 = 4,\\
S_5^2 - S_5^1 &= 10 - 10 = 0.
\end{aligned}
$$

Therefore, it results in an offset $x = \max(0, 0, 4, 0) = 4$ to the start of mode SI^1 and is shown in Figure 7.6. The starting time of the new mode SI^1, namely the source actor A_1^1, must be first delayed to the time when A_2^1 completes the iteration period H^2 in the last iteration, namely time 8 as shown as the first bold line in Figure 7.6. In addition, the MOO $x = 4$ must be further added to the starting time of A_1^1 (the second bold line in Figure 7.6). Figure 7.6 also shows another transition from mode SI^1 to SI^2 with a MCR occurred at time $t_{\mathrm{MCR2}} = 23$. The starting time of the source actor A_1^2 in the new mode SI^2 must be first delayed to the time 28 (the thrid bold line in Figure 7.6), namely the time when A_1^1 completes the last iteration in the old mode

SI^1. To calculate the MOO x for this transition, the following equations hold:

$$\begin{aligned}
S_1^1 - S_1^2 &= 0 - 0,\\
S_2^1 - S_2^2 &= 1 - 1 = 0,\\
S_3^1 - S_3^2 &= 5 - 9 = -4,\\
S_5^1 - S_5^2 &= 10 - 10 = 0.
\end{aligned}$$

Thus, the equations above result in $x = \max(0, 0, -4, 0) = 0$. For this transition, the new mode SI^2 starts at time 28 as shown in Figure 7.6.

The MOO protocol offers several advantages over the ST protocol. Essentially, the MOO protocol retains the iteration latency of the MADF graph in the new mode the same as the initial value, thereby avoiding the interference between the old and new modes. For instance, after *MCR1* and *MCR2* in Figure 7.6, mode SI^1 and SI^2 still have the initial iteration latency $L^1 = 10$ and $L^2 = 10$ as shown in Figure 7.3. Therefore, efficiently computing the starting time of MADF actors in the new mode becomes feasible and it plays an important role in deriving a hard-real time schedule for the MADF actors. As a result, analysis of the worst-case transition delay is much simpler (see Theorem 7.3.1) than that of the ST protocol, because the transition delay does not depend on the order of the transitions occurred previously.

Concerning the transition delay, it may be the case that the MOO protocol results in initially longer transition delay than the ST protocol does due to the offset given in Definition 7.2.15. For *MCR1* occurred at time t_{MCR1}, the transition delay of the MOO protocol is $\Delta^{2\to1} = 22 - 1 = 21$ as shown in Figure 7.6, whereas the transition delay of the ST protocol is equal to $\Delta^{2\to1} = 18 - 1 = 17$ as shown in Figure 7.4. On the other hand, let us consider the same transition request *MCR2* occurred at time $t_{\mathrm{MCR2}} = 23$ shown in Figures 7.4 and Figure 7.6. For *MCR2*, the ST protocol results in transition delay $\Delta^{1\to2} = 42 - 23 = 19$ as shown in Figure 7.4. In contrast, the transition delay for the MOO protocol is $\Delta^{1\to2} = 38 - 23 = 16$ as shown in Figure 7.6. The MOO protocol could provide shorter transition delay than the ST protocol, thereby faster responsiveness to a mode transition.

7.3 Hard real-time Scheduling of MADF

Based on the proposed MOO protocol for mode transitions, in this section, we propose a HRT scheduling framework for MADF. We further show an analysis technique for mode transitions in MADF to reason about transition delays, such that timing constraints can be guaranteed. The HRT scheduling framework for acyclic MADF graphs is an extension of the SPS [22] framework initially developed for acyclic CSDF graphs.

As explained in Section 2.3, the key concept of the SPS framework is to derive a periodic taskset representation for a CSDF graph. Since the steady-state of a mode can be considered as a CSDF graph according to Definitions 7.2.7 and 7.2.10, it is thus straightforward to represent the steady-state of a MADF graph as a periodic taskset and schedule the resulting taskset using any well-known HRT scheduling algorithm. Using the SPS framework, we can derive the two main parameters for each MADF actor in mode SI^k, namely the period (T_i^k in Equation (2.16) on page 34) and the earliest starting time (S_i^k in Equation (2.17) on page 35). Under SPS, the iteration period in mode SI^k is obtained as $H^k = q_i^k T_i^k$, $\exists A_i^k \in \mathcal{A}$. Below, we focus on determining the earliest starting time of each actor in the new mode upon a transition. From the earliest starting time, we can reason about the transition delay to quantify the responsiveness of a transition.

Upon a MCR, a MADF graph can safely switch to the new mode if all of its actors have completed their last iteration in the old mode. In this case, the firings of MADF actors in the new mode do not overlap with the firings of actors in the old mode. This is called synchronous protocol [104] in real-time systems with mode change. One of its advantages is the simplicity, i.e., the synchronous protocol does not require any schedulability test at both compile-time and run-time. However, other protocols lead to earlier starting times than the synchronous protocol. Therefore, the synchronous protocol sets an upper bound on the earliest starting time for each MADF actor in the new mode.

Lemma 7.3.1. *For a MADF graph G under SPS and a MCR from mode SI^o to SI^l at time t_{MCR}, the earliest starting time of actor A_i^l, $\hat{\sigma}_i^{o\to l}$, is upper bounded by*

$$\hat{\sigma}_i^{o\to l} = F_{src}^o + S_{snk}^o + S_i^l, \tag{7.13}$$

where F_{src}^o indicates the time when the source actor A_{src}^o completes its last iteration It^o of the old mode SI^o and is given by

$$F_{src}^o = t_S^o + \left\lceil \frac{t_{MCR} - t_S^o}{H^o} \right\rceil H^o. \tag{7.14}$$

t_S^o is the starting time of mode SI^o and H^o is the iteration period of mode SI^o.

Proof. As explained previously for a transition from mode SI^o to SI^l, the upper bound of the earliest starting time for each actor A_i^l is computed in such a way that no firings of actors A_i^o and A_i^l occur simultaneously. This means, the start of an actor A_i^l must be later than all actors A_i^o have completed the last iteration It^o of the old

Table 7.4: Actor parameter for G_1 in Figure 7.1.

Mode	SI^1				SI^2				
Actor	A_1^1	A_2^1	A_3^1	A_5^1	A_1^2	A_2^2	A_3^2	A_4^2	A_5^2
WCET	1	4	1	1	1	8	1	3	1
period (T_i)	2	4	4	4	4	8	8	8	4
starting time (S_i)	0	2	6	14	0	4	12	8	20
utilization (u_i)	$\frac{1}{2}$	1	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	1	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{1}{4}$

mode SI^o. Given that mode SI^o starts at time t_S^o, the completion time of all actors A_i^o in the last iteration It^o can be thus computed as

$$F_{\text{snk}}^o = t_S^o + \left\lfloor \frac{t_{MCR} - t_S^o}{H^o} \right\rfloor H^o + S_{\text{snk}}^o + H^o. \tag{7.15}$$

where F_{snk}^o is the time when the old mode SI^o completes the last iteration It^o. It is assumed that the sink actor A_{snk}^o is the last actor to complete the iteration, i.e., $\forall A_i^o \in \mathcal{A}, S_i^o \leq S_{\text{snk}}^o$. Given Equation (7.14), Equation (7.15) can be rewritten as

$$F_{\text{snk}}^o = t_S^o + \left\lceil \frac{t_{MCR} - t_S^o}{H^o} \right\rceil H^o + S_{\text{snk}}^o = F_{\text{src}}^o + S_{\text{snk}}^o.$$

Now, starting the source actor A_{src}^l at any time later than F_{snk}^o is valid without introducing simultaneous execution of actors A_i^o and A_i^l. Therefore, the earliest starting time of source actor A_{src}^l is $\hat{\sigma}_{\text{src}}^{o\to l} = F_{\text{snk}}^o$. For any actor $A_i^l \in \mathcal{A} \setminus A_{\text{src}}^l$, its earliest starting times must satisfy Equation (2.17) on page 35 imposed by the SPS framework. That is, the earliest starting starting time $\hat{\sigma}_i^{o\to l}$ of actor A_i^l can be obtained by adding S_i^l to $\hat{\sigma}_{\text{src}}^{o\to l}$. ■

Let us consider the actor parameters given in Table 7.4 for G_1 in Figure 7.1. The third row shows the WCET for each actor in modes SI^1 and SI^2. Based on WCETs, the period (fourth row in Table 7.4) and the earliest starting time (fifth row in Table 7.4) for each actor in the steady-state of both modes are obtained according to Equations 2.16 and 2.17, respectively. Given $\vec{q}^2$ in Equation (7.8), we can also compute iteration period $H^2 = q_1^2 T_1^2 = 2 \times 4 = 8$. Now consider the mode transition from mode SI^2 to SI^1 shown in Figure 7.7. Assume that the MCR occurs at time $t_{\text{MCR}} = 13$ and mode SI^2 starts at time $t_S^2 = 8$. The completion time of the last iteration It^2 is equal to the completion time of the sink actor A_5^2 computed as

$$F_{\text{snk}}^2 = t_S^2 + \left\lceil \frac{t_{\text{MCR}} - t_S^2}{H^2} \right\rceil H^2 + S_5^2 = 8 + \left\lceil \frac{13-8}{8} \right\rceil 8 + 20 = 36.$$

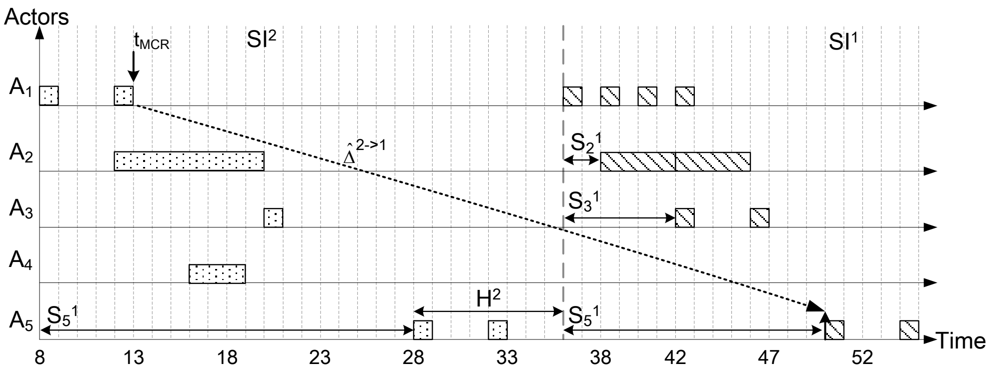


Figure 7.7: Upper bounds of earliest starting times for transition from mode SI^2 to SI^1.

In Figure 7.7, F^2_{snk} corresponds to the earliest starting time of the source actor A_1^1 (bold dashed line). Finally, we can compute the earliest starting time for each actor in the new mode SI^1 by adding S_i^1. Considering for instance the sink actor A_5^1 in the new mode with $S_5^1 = 14$, the upper bound of its earliest starting time can be obtained as

$$\hat{\sigma}_5^{2\to1} = F^2_{\text{src}} + S_5^2 + S_5^1 = F^2_{\text{snk}} + S_5^1 = 36 + 14 = 50.$$

We can thus compute the transition delay (*cf.* Definition 7.2.14) as

$$\hat{\Delta}^{2\to1} = \hat{\sigma}_5^{2\to1} - t_{\text{MCR}} = 50 - 13 = 37.$$

Although the upper bound of the earliest starting times is easy to obtain for MADF actors in the new mode, it does not provide a responsive mode transition. Therefore, here we aim at deriving a lower bound of the earliest starting times with the proposed MOO protocol.

Lemma 7.3.2. *For a MADF graph under SPS and a MCR from mode SI^o to SI^l at time t_{MCR}, the earliest starting time of actor A_i^l using the MOO protocol is lower bounded by $\check{\sigma}_i^{o\to l}$ given as*

$$\check{\sigma}_i^{o\to l} = F^o_{src} + x + S_i^l, \tag{7.16}$$

where F^o_{src} is given in Equation (7.14) *and x is given in Equation* (7.12).

Proof. Under the MOO protocol, the start of actor A_i^l must be later than the time when A_i^o, if any, completes its last iteration in the old mode SI^o. We assume that the source actor A^l_{src} is the first actor to start in the new mode SI^l, i.e., $\forall A_i^l \in \mathcal{A}, S_i^l \geq S^l_{\text{src}}$. Thus, the starting time of the source actor A^l_{src} is at least equal to the completion

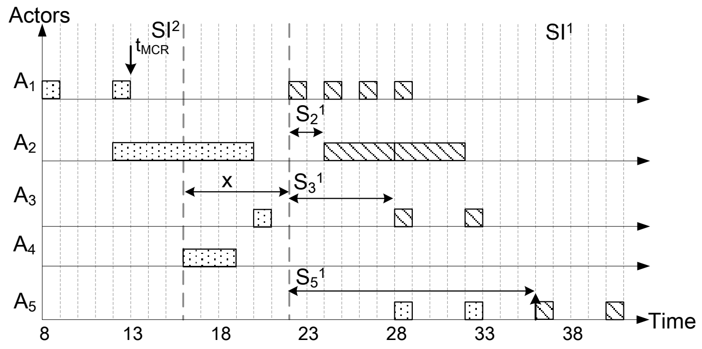


Figure 7.8: Earliest starting times for transition from mode SI^2 to SI^1 with the MOO protocol.

time of the last iteration of A^o_{src}, denoted by F^o_{src}. Given F^o_{src} in Equation (7.14), it thus holds $\check{\sigma}^{o\to l}_{\text{src}} \geq F^o_{\text{src}}$ Then, the offset x because of the MOO protocol given in Equation (7.12) must be taken into account. Consequently, the earliest starting time of A^l_{src} is lower bounded by $\check{\sigma}^{o\to l}_{\text{src}} = F^o_{\text{src}} + x$. For any actor $A^l_i \in \mathcal{A} \setminus A^l_{\text{src}}$, its earliest starting times must satisfy Equation (2.17) on page 35 imposed by the SPS framework. Hence, the earliest starting time $\check{\sigma}^{o\to l}_i$ of actor A^l_i can be obtained by adding S^l_i to $\check{\sigma}^{o\to l}_{\text{src}}$. ■

Let us consider again the transition from mode SI^2 to SI^1. With the MOO protocol, the mode transition is illustrated in Figure 7.8. Upon the MCR at time $t_{\text{MCR}} = 13$ and $t^2_S = 8$, source actor A^2_1 completes its last iteration It^2 in the old mode SI^2 at the time (*cf.* Equation (7.14)) given as

$$F^2_{\text{src}} = F^2_1 = t^2_S + \left\lceil \frac{t_{\text{MCR}} - t^2_S}{H^2} \right\rceil H^2 = 8 + \left\lceil \frac{13-8}{8} \right\rceil 8 = 16$$

This is the earliest possible time at which mode transition is allowed. For MOO, x can be computed according to Equation (7.12). Therefore, the following equations hold:

$$\begin{aligned}
S^2_1 - S^1_1 &= 0 - 0, \\
S^2_2 - S^1_2 &= 4 - 2 = 2, \\
S^2_3 - S^1_3 &= 12 - 6 = 6, \\
S^2_5 - S^1_5 &= 20 - 14 = 6.
\end{aligned}$$

It thus yields $x = \max(0, 2, 6, 6) = 6$, i.e., an offset $x = 6$ is added to F^2_{src}. It can be seen in Figure 7.8 that the source actor A^1_1 starts at time $F^2_{\text{src}} + x = 16 + 6 = 22$. Finally, the earliest starting times of actors in mode SI^1 can be determined by adding S^1_i. Considering for instance A^1_5 in the new mode, the lower bound of its earliest starting time can be obtained as:

$$\check{\sigma}^{2\to1}_5 = F^2_{\text{src}} + x + S^1_5 = 16 + 6 + 14 = 36.$$

Now, the transition delay (*cf.* Definition 7.2.14) can be obtained as

$$\check{\Delta}^{2\to1} = \check{\sigma}^{2\to1}_5 - t_{\text{MCR}} = 36 - 13 = 23.$$

Scheduling Analysis under a Fixed Allocation of Actors

During a mode transition of a MADF graph according to the MOO protocol, actors execute simultaneously in the old and new modes. The derived starting time in Lemma 7.3.2 for each actor is only the lower bound because the allocation of actors on PEs is not taken into account yet. That means, the derived starting times according to Lemma 7.3.2 can be only achieved during mode transitions when each actor is allocated to a separate PE. In a practical system where multiple actors are allocated to the same PE, the PE may be potentially overloaded during mode transitions. To avoid overloading PEs, the earliest starting times of actors may be further delayed.

Lemma 7.3.3. *For a MADF graph under SPS, a MCR from mode SI^o to SI^l, and an m-partition of all actors $\Psi = \{\Psi_1, \ldots, \Psi_m\}$, where m is the number of PEs, the earliest starting time of an actor A^l_i without overloading the underlying PE is given by*

$$\sigma^{o\to l}_i = F^o_{src} + \delta^{o\to l} + S^l_i, \tag{7.17}$$

where F^o_{src} is computed by Equation (7.14) *and $\delta^{o\to l}$ is obtained as*

$$\delta^{o\to l} = \min_{t\in[x, S^o_{snk}]} \{t : U_j(k) \le UB,\ \forall k \in [t, S^o_{snk}] \wedge \forall \Psi_j \in \Psi\}. \tag{7.18}$$

UB denotes the utilization bound of the scheduling algorithm used to schedule actors on each PE. Ψ_j contains the set of actors allocated to PE$_j$. $U_j(k)$ is the total utilization of PE$_j$ at time k demanded by both mode SI^o and SI^l actors, and is given by

$$U_j(k) = \underbrace{\sum_{A^o_d\in\Psi_j} \left(u^o_d - h(k - S^o_d)\cdot u^o_d\right)}_{U^o_j(k)} + \underbrace{\sum_{A^l_d\in\Psi_j} \left(h(k - S^l_d - t)\cdot u^l_d\right)}_{U^l_j(k)}, \tag{7.19}$$

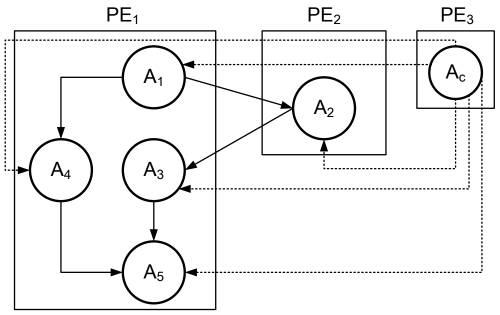


Figure 7.9: Allocation of all MADF actors in Figure 7.1 to 3 PEs.

$A_d^o \in \Psi_j$ is an actor active in the old mode SI^o and allocated to PE_j. $A_d^l \in \Psi_j$ is an actor active in the new mode SI^l and allocated to PE_j. $h(t)$ is the Heaviside step function.

Proof. Lemma 7.3.2 shows the lower bound of the earliest starting time for actor A_i^l in the new mode SI^l. However, starting A_i^l at time $\check{\sigma}_i^{o\to l}$ may overload PE j, i.e., the resulting total utilization of PE j, denoted by $U_j(\check{\sigma}_i^{o\to l})$, exceeds UB. Therefore, in this case, the earliest starting time $\sigma_i^{o\to l}$ must be delayed by $\delta^{o\to l}$ such that $U_j(\sigma_i^{o\to l}) \leq UB$ holds. From Equation (7.17) and Equation (7.16), we can see that $\delta^{o\to l}$ is lower bounded by x which corresponds to the MOO protocol. In addition, $\delta^{o\to l}$ is upper bounded by S_{snk}^o if we consider Equation (7.17) and Equation (7.13) on page 131.

$\delta^{o\to l}$ of interest is the minimum time t in the bounded interval $[x, S_{\text{snk}}^o]$ that satisfies two conditions.

Condition 1: for each PE j, the total utilization cannot exceed UB at time t, i.e., $U_j(t) \leq UB$. The total utilization $U_j(t)$ in Equation (7.19) consists of two parts, namely $U_j^o(t)$ and $U_j^l(t)$. $U_j^o(t)$ denotes the PE capacity occupied by the actors in mode SI^o that are not completed yet. Additional PE capacity $U_j^l(t)$ is demanded by the already released actors in the new mode SI^l.

Condition 2: We need to check all time instants $k > t$ in the interval $[t, S_{\text{snk}}^o]$, such that $U_j(k) \leq UB$, to guarantee that each PE$_j$ is not overloaded during the mode transition. ■

Figure 7.9 shows all actors of G_1 in Figure 7.1 allocated to 3 PEs and let us assume that the actors allocated to each PE are scheduled using the EDF algorithm. The utilization bound of EDF is given as $UB = 1$ [80]. Given this allocation and the transition from mode SI^2 to SI^1 shown in Figure 7.8, the lower bound of the earliest

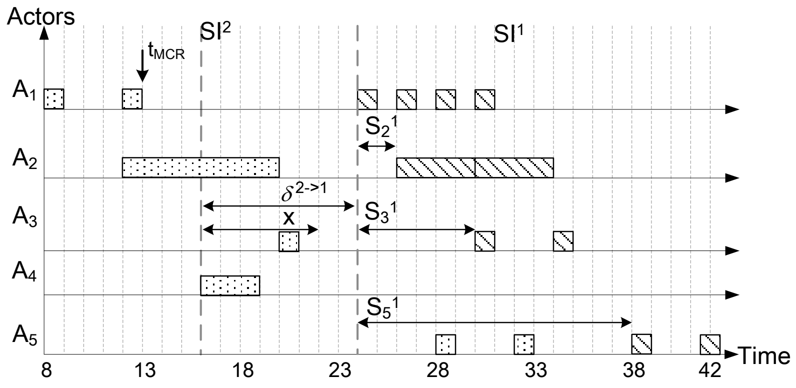


Figure 7.10: Earliest starting times for transition SI^2 to SI^1 on 2 PEs shown in Figure 7.9.

starting time $\check{\sigma}_1^{2\to1} = 22$ for actor A_1^1 cannot be achieved. At time 22, only actor A_1^2 has completed the last iteration It^2 on PE_1. Starting the new mode SI^1 at time 22 corresponds to $\delta^{2\to1} = x = 6$. The total utilization of PE_1 demanded by the actors in the old mode SI^2 at time 22, i.e., $U_1^2(6)$, can be computed as follows:

$$\begin{aligned} U_1^2(6) &= \sum_{A_d^2 \in \Psi_1} u_d^2 - h(6 - S_d^2) \cdot u_d^2,\ d \in \{1,3,4,5\} \\ &= u_1^2 - h(6)\cdot u_1^2 + u_3^2 - h(-6)\cdot u_3^2 + u_4^2 - h(-2)\cdot u_4^2 + u_5^2 - h(-14)\cdot u_5^2 \\ &= 0 + u_3^2 + u_4^2 + u_5^2 = \frac{1}{8} + \frac{3}{8} + \frac{1}{4} = \frac{3}{4} \end{aligned}$$

Enabling A_1^1 in the new mode SI^1 at time 22 would yield

$$U_1(6) = U_1^2(6) + u_1^1 = \frac{3}{4} + \frac{1}{2} > UB = 1,$$

thereby leading to being unschedulable on PE_1. In this case, the earliest times of all actors in mode SI^1 must be delayed by $\delta^{2\to1} = 8$ to time 24 as shown in Figure 7.10. At time 24, the total utilization demanded by mode SI^2 actors is

$$\begin{aligned} U_1^2(8) &= \sum_{A_d^2 \in \Psi_1} u_d^2 - h(8 - S_d^2) \cdot u_d^2,\ d \in \{1,3,4,5\} \\ &= u_1^2 - h(8)\cdot u_1^2 + u_3^2 - h(-4)\cdot u_3^2 + u_4^2 - h(0)\cdot u_4^2 + u_5^2 - h(-12)\cdot u_5^2 \\ &= 0 + u_3^2 + 0 + u_5^2 = \frac{1}{8} + \frac{1}{4} = \frac{3}{8}. \end{aligned}$$

Now, enabling A_1^1 in the new mode at time 24 results in the total utilization of PE_1 as

$$U_1(8) = U_1^2(8) + u_1^1 = \frac{3}{8} + \frac{1}{2} < 1.$$

Next, assuming that the new mode SI^1 starts at time 24, we need to check that the remaining actors in the new mode SI^1, namely A_3^1 and A_5^1, can start with S_3^1 and S_5^1 respectively without overloading PE_1. For instance, enabling A_3^1 at time 24 results in starting time $\sigma_3^{2\to1} = 24 + S_3^1 = 24 + 6 = 30$. At time 30, the total utilization of PE_1 can be obtained according to Equation (7.19) as follows:

$$\begin{aligned}
U_1^2(8+6) &= \sum_{A_d^2 \in \Psi_1} u_d^2 - h(14 - S_d^2) \cdot u_d^2,\ d \in \{1,3,4,5\} \\
&= u_1^2 - h(14) \cdot u_1^2 + u_3^2 - h(2) \cdot u_3^2 + u_4^2 - h(6) \cdot u_4^2 + u_5^2 - h(-6) \cdot u_5^2 \\
&= 0 + 0 + 0 + u_5^2 = \frac{1}{4}, \\
U_1^1(8+6) &= \sum_{A_d^1 \in \Psi_1} \left(h(14 - S_d^1 - 8) \cdot u_d^1 \right),\ d \in \{1,3,5\} \\
&= h(6)u_1^1 + h(0)u_3^1 + h(-8)u_5^1 \\
&= \frac{1}{2} + \frac{1}{4} = \frac{3}{4}, \\
U_1(8+6) &= U_1^2(8+6) + U_1^1(8+6) = 1 = UB.
\end{aligned}$$

Hence, actors A_5^2, A_1^1, and A_3^1 are schedulable on PE_1 using EDF. Similarly, starting A_5^1 at time $\sigma_5^{2\to1} = 24 + S_5^1 = 38$ still keeps the resulting set of actors schedulable on PE_1.

Using Lemma 7.3.3, we can quantify the maximum and minimum transition delays for any transition from mode SI^o to SI^l.

Theorem 7.3.1. *For a MADF graph under SPS, a fixed allocation of all MADF actors $\Psi = \{\Psi_1, \ldots, \Psi_m\}$ to m PEs, and a MCR from mode SI^o to SI^l, the minimum transition delay is given by*

$$\Delta_{min}^{o\to l} = \delta^{o\to l} + S_{snk}^l \tag{7.20}$$

and the maximum transition delay is given by

$$\Delta_{max}^{o\to l} = \delta^{o\to l} + S_{snk}^l + H^o, \tag{7.21}$$

where $\delta^{o\to l}$ is computed by Lemma 7.3.3, S_{snk}^l is the starting time of the sink actor in the new mode SI^l, and H^o is the iteration period of the old mode SI^o.

Proof. For a MCR from mode SI^o to SI^l, the transition delay $\Delta^{o \to l}$ of a MADF graph is given in Definition 7.2.14 as $\Delta^{o \to l} = \sigma_{\text{snk}}^{o \to l} - t_{\text{MCR}}$, where the earliest starting time of the sink actor is calculated as $\sigma_{\text{snk}}^{o \to l} = F_{\text{src}}^o + \delta^{o \to l} + S_{\text{snk}}^l$ according to Lemma 7.3.3. Therefore, $\Delta^{o \to l}$ can be rewritten as $\Delta^{o \to l} = F_{\text{src}}^o + \delta^{o \to l} + S_{\text{snk}}^l - t_{\text{MCR}}$. Essentially, $\Delta^{o \to l}$ is composed of three parts. In the first part, the MOO transition protocol together with a fixed allocation of the MADF actors determine $\delta^{o \to l}$. The second part S_{snk}^l results from the SPS framework. These two parts thus can be determined at compile-time. The third part $F_{\text{src}}^o - t_{\text{MCR}}$ depends on when the MCR occurs, namely at t_{MCR}, which can only be determined at run-time. In the following, we distinguish two cases for t_{MCR}:

Case 1: Assume that the MCR occurs at the end of an iteration of the source actor in the old mode SI^o, i.e., $t_{\text{MCR}} = F_{\text{src}}^o$. Then, the source actor shall be only delayed by $\delta^{o \to l}$ to start in the new mode SI^l according to Lemma 7.3.3, thereby guaranteeing the fastest possible start of the new mode SI^l. As a consequence, it results in the minimum possible transition delay. Therefore, substituting $t_{\text{MCR}} = F_{\text{src}}^o$, we obtain

$$\Delta_{\min}^{o \to l} = F_{\text{src}}^o + \delta^{o \to l} + S_{\text{snk}}^l - F_{\text{src}}^o = \delta^{o \to l} + S_{\text{snk}}^l.$$

Case 2: Assume that the MCR occurs at the beginning of an iteration of the source actor in the old mode SI^o, i.e., $t_{\text{MCR}} = F_{\text{src}}^o - H^o$. Then, the source actor cannot start in the new mode before it completes the whole iteration in the old mode SI^o followed by the delay $\delta^{o \to l}$ according to Lemma 7.3.3. Therefore, the maximum transition delay is computed as follows:

$$\Delta_{\max}^{o \to l} = F_{\text{src}}^o + \delta^{o \to l} + S_{\text{snk}}^l - (F_{\text{src}}^o - H^o) = \delta^{o \to l} + S_{\text{snk}}^l + H^o.$$

■

It can be seen from Theorem 7.3.1 that the maximum and minimum transition delays solely depend on the allocation of MADF actors and the old and new modes in question, irrespective of the previously occurred transitions. The old and new modes determine H^o and S_{snk}^l, respectively, while the allocation of MADF actors determines the value of $\delta^{o \to l}$. Here, the offset x due to our MOO protocol is captured in $\delta^{o \to l}$ and can be considered as performance overhead if $x \neq 0$. The other parts, namely H^o and S_{snk}^l, in the maximum and minimum transition delays cannot be avoided as they will be present in any transition protocol.

7.4 Case Study

In this section, we present a case study of using the proposed MADF MoC and the developed HRT scheduling explained in Section 7.3. With the case study, we show that the MADF MoC is able to capture different application modes and the transitions between them. Then, the main focus of the case study is to analyze the transition delays and to demonstrate the effectiveness of the proposed MOO transition protocol.

We consider a real-life adaptive application from the StreamIT benchmark suit [54], called Vocoder, which implements a phase voice encoder and performs pitch transposition of recorded sounds from male to female. We modeled Vocoder with a MADF graph with 4 modes, which capture different workloads. The MADF graph of Vocoder is shown in Figure 7.11. Depending on the desired quality of audio encoding and various performance requirements, users may switch between four different modes of Vocoder at run-time. The four modes $\mathcal{S} = \{SI^8, SI^{16}, SI^{32}, SI^{64}\}$ specify different lengths of the Discrete Fourier Transform (DFT), denoted by $\mathsf{dl} \in \{8, 16, 32, 64\}$. Mode SI^8 ($\mathsf{dl} = 8$) requires the least amount of computation at the cost of the worst voice encoding quality among all DFT lengths. Mode SI^{64} ($\mathsf{dl} = 64$) produces the best quality of voice encoding among all modes, but is computationally intensive. The other two modes SI^{16} and SI^{32} explore the trade-off between the quality of the encoding and computational workload. A transition from one mode to any other one is possible, thereby resulting in totally 12 possible transitions. At run-time, reconfiguration of the parameter dl is triggered by the environment, e.g., the user in this case. Subsequently, control actor A_c propagates dl to the dataflow actors shown in Figure 7.11 through the dashed-lined edges.

We measured the WCETs of all dataflow actors in Figure 7.11 in the four modes on an ARM Cortex-A9 [1] PE. All dataflow actors were compiled using the compiler `arm-xilinx-eabi-gcc 4.7.2` with the vectorization option. The WCETs of all actors in all four modes are given in Table 7.5. It is worth to note that in mode SI^8, actors Spec2Env and male2female exhibit exceptionally high WCETs. It is because parameter dl represents the size of the inner-most loop in the computation of actors Spec2Env and male2female. Small dl (in this case $\mathsf{dl} = 8$) leads to the fact that the inner-most loop cannot be vectorized by the compiler. In the other modes from SI^{16} to SI^{64}, larger sizes of the inner-most loop (dl equal to 16, 32, and 64, respectively) lead to full vectorization of the computation of actors Spec2Env and male2female. Therefore, in these three modes, the WCETs of actors Spec2Env and male2female are even smaller than the ones in mode SI^8. The dataflow actors of Vocoder are allocated to 4 PEs as shown in Figure 7.12. This allocation guarantees that the shortest periods (maximum throughput) in the steady-states of all modes can be achieved.

Table 7.6 shows the performance results for the four modes in their steady-state under SPS. For instance, the second column at the first row in Table 7.6 indicates that it is guaranteed for sink actor WriteWave to produce 256 samples per 917,451

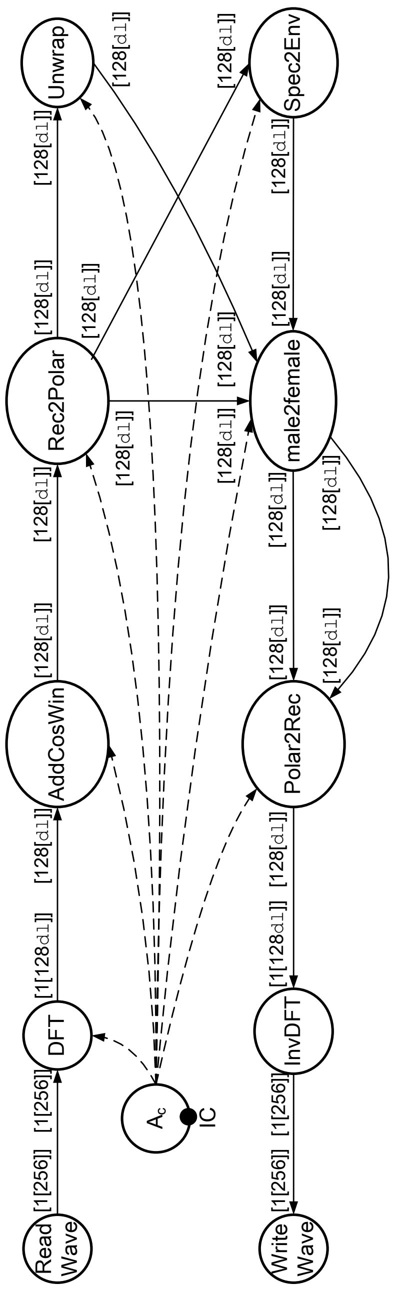


Figure 7.11: MADF graph of Vocoder.

Table 7.5: WCETs of all actors in Vocoder (in clk.).

Mode	ReadWave	DFT	AddCosWin	Rec2Polar	Unwrap	Spec2Env	male2female	Polar2Rec	InvDFT	WriteWave
SI^8	3704	16,775	16	90	359	7,168	1,093	3	236	3660
SI^{16}	3704	35,121	35	183	691	1,163	138	260	644	3660
SI^{32}	3704	71,337	75	366	1,393	1,392	210	507	988	3660
SI^{64}	3704	144,531	150	1,156	2,346	1,696	426	1,056	3,630	3660

clock cycles in mode SI^8. This is the "worst-case" performance among all four modes because the Spec2Env actor exhibits exceptionally high workload (*cf.* WCETs in Table 7.5 and Definition 2.3.2 on page 34) in mode SI^8. Consequently, actor Spec2Env becomes the "bottleneck" actor, so that mode SI^8 cannot be scheduled with

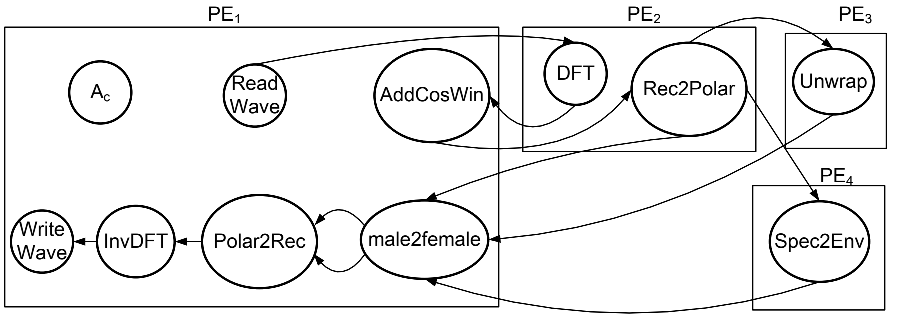


Figure 7.12: Allocation of dataflow actors of Vocoder to 4 PEs. The control edges are omitted to avoid cluttering.

Table 7.6: Performance results of four modes of Vocoder in the steady-state.

Mode	Period (T in clk.)	Total utilization (U)	Iteration latency (L)
SI^8	917,504	1.24	7,339,608
SI^{16}	148,864	2.36	1,191,436
SI^{32}	178,176	3.19	1,425,448
SI^{64}	300,288	3.4	2,402,550

higher throughput (shorter period). Nevertheless, all mode SI^8 actors as a whole require a total utilization of only 1.24 (see the third column in Table 7.6) which is the least among all modes. From Table 7.6, we can see that MADF together with the SPS framework brings another advantage of efficiently utilizing PE resources. For example, in case that Vocoder is switched to a mode with lower utilization, idle capacity of PEs can be efficiently utilized by admitting other applications at run-time without introducing interference to the currently running Vocoder.

Now, we focus on the performance results of the MOO protocol, namely transition delays, for all possible transitions between the four modes of Vocoder. Table 7.7 shows both the minimum and maximum transition delays in accordance with Theorem 7.3.1 for all transitions. We can see in the second column of Table 7.7 that, in the best case, the transition delays for 6 out of 12 transitions remain the same as the iteration latencies of the new modes. This can be seen as $x = 0$ shown in the fourth column. In these 6 transitions, the proposed MOO protocol does not introduce any extra delay. In the 6 remaining transitions, as expected, the MOO protocol introduces offset $x > 0$ to the transitions from an old mode with a longer iteration latency to a new mode with a shorter iteration latency. For instance, the largest x (in bold shown in Table 7.7) happens in case of a transition from mode SI^8 with the longest iteration latency (see the fourth column in Table 7.6) to mode SI^{16}

Table 7.7: Performance results for all mode transitions of Vocoder.

Transition (SI^o to SI^l)	$\Delta_{\min}^{o\to l}$ (in clk.)	$\Delta_{\max}^{o\to l}$ (in clk.)	x (in clk.)	$\delta^{o\to l}$ (in clk.)
$SI^8 \to SI^{64}$	3,636,815	4,554,266	1,234,264	1,234,264
$SI^8 \to SI^{32}$	2,903,988	3,821,439	1,478,540	1,478,540
$SI^8 \to SI^{16}$	2,728,479	3,645,930	**1,537,043**	1,537,043
$SI^{16} \to SI^{64}$	2,402,550	2,551,480	0	0
$SI^{16} \to SI^{32}$	1,425,448	1,574,378	0	0
$SI^{16} \to SI^8$	7,339,608	7,488,538	0	0
$SI^{32} \to SI^{64}$	2,402,550	2,580,731	0	0
$SI^{32} \to SI^{16}$	1,425,448	1,603,629	234,012	234,012
$SI^{32} \to SI^8$	7,339,608	7,517,789	0	0
$SI^{64} \to SI^{32}$	2,402,550	2,702,869	977,102	977,102
$SI^{64} \to SI^{16}$	2,402,550	2,702,869	1,211,114	1,211,114
$SI^{64} \to SI^8$	7,339,608	7,639,927	0	0

with the shortest iteration latency. To quantify x, we compute the percentage of x compared to both minimum and maximum transition delays as

$$\Omega_{\min} = \frac{x}{\Delta_{\min}^{o\to l}} \times 100\%, \quad \Omega_{\max} = \frac{x}{\Delta_{\max}^{o\to l}} \times 100\%.$$

$\Omega_{\min}$ varies from the worst-case 56% to the best case 16% with an average of 41%, whereas $\Omega_{\max}$ varies from the worst-case 44% to the best case 14% with an average of 33%. Therefore, the increase of the transition delays due to the MOO protocol is reasonable for this real-life application.

Next, we consider the effect of the actor allocation shown in Figure 7.12 on the earliest starting times of actors in the new mode upon a transition (*cf.* Lemma 7.3.3). In this particular example, we find out that no extra delay is incurred to any actor in all transitions due to the fixed actor allocation. This can be seen from the fourth and fifth columns in Table 7.7, where $\delta^{o\to l} = x$.