FSM-based SADF*

S. Stuijk, A.H. Ghamarian, B.D. Theelen, M.C.W. Geilen and T. Basten
Eindhoven University of Technology, Department of Electrical Engineering
s.stuijk@tue.nl

1 Introduction

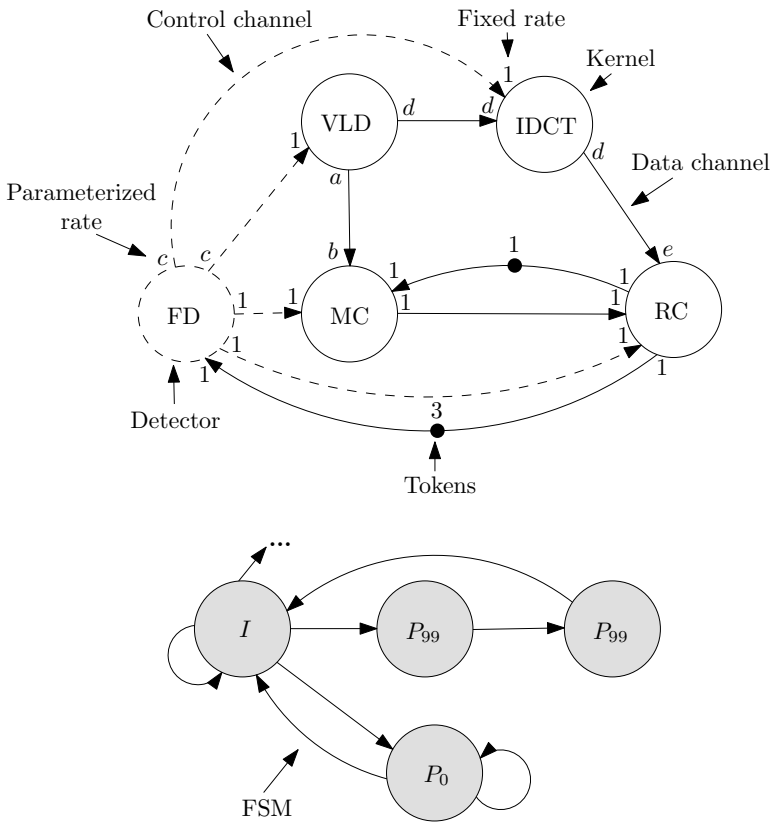
The Synchronous Dataflow (SDF) Model-of-Computation (MoC) [3] has become a popular model for modelling streaming applications. The SDF MoC fits well with the streaming and pipelining characteristics of these applications, it can capture many mapping decisions and resource constraints, and it allows design-time analysis of timing (throughput, latency) and resource usage. When designing a system with a predictable timing behaviour, the SDF model of an application assumes worst-case rates for the edges and worst-case execution times for the actors. In the situation that an application contains a lot of dynamism, the use of worst-case values could lead to over-dimensioning of the system for many run-time situations. Therefore, the concept of system scenarios has been introduced in [1, 2]. However, these scenarios cannot be modelled into an SDF graph. This report introduces an extension to the SDF MoC that can be used to model an application together with its scenarios. This new MoC is called the Finite State Machine-based Scenario-Aware Data-Flow (FSM-based SADF) MoC. The FSM-based SADF MoC is a restricted form of the general SADF MoC that has been introduced in [4, 5]. These restrictions make analysis of the timing behaviour of a model specified in the FSM-based SADF MoC faster than a model specified in the general SADF MoC. However, analysis result may be less tight due to the abstractions made in the FSM-based SADF MoC. The next section informally introduces the FSM-based SADF MoC by explaining how an H.263 decoder can be modelled. Sec. 3 discusses the differences between the FSM-based SADF MoC and the general SADF MoC. Sec. 4 gives some basic definitions which are used in Sec. 5 tp define the FSM-based SADF MoC. Finally, Sec. 6 gives the operational semantics of this MoC.

2 Example FSM-based SADF graph

The example of an H.263 decoder that is modelled with an FSM-based SADF graph is shown in Fig. 1. The nodes represent the processes of an application. Two types of processes are distinguished. Kernels model the data processing part of a streaming application, whereas the detector represents the part that dynamically determines scenarios and controls the behaviour of processes. The Frame Detector (FD) represents for example that segment of the actual Variable Length Decoder (VLD) code that is responsible for determining the frame type and the number of macro blocks to decode. All other processes are kernels. In the graph shown in Fig. 1, it is assumed that two different types of frame exists (I or P type). When a frame of type I is found, a total of 99 macro blocks must always be processed. A frame of type P may require the processing of (up to) 0, 30, 40, 50, 60, 70, 80, or 99 macro blocks. So, in total 9 different scenarios (combinations of frame type and number of macro blocks to decode) exist and must be modelled in the graph. These 9 scenarios capture the varying resource requirements of the application.

The edges in an FSM-based SADF graph, called channels, denote possible dependencies between processes. Two types of channels are distinguished. Control channels (dashed edges) originate from the detector and go to a kernel. Data channels (solid edges) originate from a kernel and go to another kernel or to the detector. Ordered streams of data items (called tokens) are sent over these channels. Similar to the SDF MoC, the FSM-based SADF MoC abstracts from the value of tokens that are communicated via a data channel. However, control channels communicate tokens that are valued. The value of a token on a control channel indicates the scenario in which the receiving kernel will operate. This token value is determined externally by the detector. In the example application, the scenario is determined by the frame type and the number of macro blocks. In reality, this scenario is determined based on the content of the video stream. However, the FSM-based SADF model abstract from such content. It uses a (non-deterministic) FSM which determines the possible scenario occurrences. The states in the FSM

*This document is a MNEMEE internal report. The work presented herein was supported in part by the EC through FP7 IST project 216224, MNEMEE.



Execution time		
VLD	P_0	0
	others	40
IDCT	P_0	0
	others	17
MC	I, P_0	0
	P_{30}	90
	P_{40}	145
	P_{50}	190
	P_{60}	235
	P_{70}	265
	P_{80}	310
	P_{99}	390
RC	I	350
	P_0	0
	P_{30}, P_{40}, P_{50}	250
	P_{60}	300
	P_{70}, P_{80}, P_{99}	320
FD	all	0

Rate	I	P_x	P_0
a	0	1	0
b	0	x	0
c	99	x	1
d	1	1	0
e	99	x	0

$x = \{30, 40, 50, 60, 70, 80, 99\}$

Figure 1: FSM-based SADF graph of an H.263 decoder.

determine the scenario in which the kernels will operate. The transitions reflect possible next scenarios given the current scenario that the detector is in. Fig. 1 shows part of the FSM for the H.263 decoder. The first scenario that is executed is the scenario in which a frame of type I is decoded. This frame (scenario) may be followed by another I frame, or a scenario in which one or more P frames with zero macro blocks are executed followed by an I frame, or a scenario in which two P frames with 99 macro blocks each are executed followed by an I frame.

3 Comparison to general SADF model

There are a number of differences between the FSM-based SADF model that is defined in this document and the general SADF model [4, 5]. In contrast to the general SADF model, the FSM-based SADF model a graph to contain exactly one detector. This detector is not allowed to have control ports. This implies that hierarchical control is not supported and only a set of scenarios can be used (i.e. sub-scenarios are not supported). The FSM-based SADF model also requires each kernel to have exactly one control input. The general SADF model allows kernels to have an arbitrary number of control ports (including zero). Furthermore, the FSM-based SADF model uses a fixed execution per scenario for the processes, whereas the general SADF model allows an execution time distribution per scenario. The FSM-based SADF model restricts the automaton associated with the detector to a (non-deterministic) FSM. The aforementioned differences between the FSM-based SADF model are all additional constraints that are imposed on the general SADF model. The FSM-based SADF model also removes a constraint from the general SADF model by supporting auto-concurrency for kernels.

4 Preliminary Definitions

Let $\mathbb{N}$ denote the positive natural numbers, $\mathbb{N}_0$ the natural numbers including 0, and $\mathbb{N}_0^\infty$ the natural numbers including 0 and infinity (∞).

To simplify the mathematical notation, we consider processes (kernels and detectors) to be connected

through *ports*. Three types of ports are distinghuised: *input ports*, *output ports* and *control ports*. The finite sets of input, output and control ports of a process p are denoted by $\mathcal{I}_p$, $\mathcal{O}_p$ and $\mathcal{C}_p$ respectively. All port sets are pairwise disjoint and we define $\mathcal{I}$, $\mathcal{O}$ and $\mathcal{C}$ to be the union of all input, output and control port sets respectively.

A channel reflects an unbounded FIFO buffer that uniquely connects an output port to either an input port (*data channel*) or a control port (*control channel*). A channel that connects ports of the same process p is also called a *self-loop* channel of p. For any channel b, $\sum_b$ indicates the finite set of all possible values of the tokens that b can transfer. $\sum_b^*$ denotes the set of all finite sequences of tokens in $\sum_b$. For a sequence $\sigma = \sigma_1 \dots \sigma_n$ in $\sum_b^*$, the i^{th} token in σ is indicated with σ_i, whereas $|\sigma|$ is the number n of tokens in σ. For sequences $\sigma, \tau, v \in \sum_b^*$, we use $\sigma + \tau$ to represent the concatentation of σ and τ. In case $v = \sigma + \tau$, we also write $v - \sigma$ to denote τ.

The non-empty finite set of scenarios in which a process p can operate is denoted by $\mathcal{S}_p$. The function $E_p : \mathcal{S}_p \to \mathbb{N}_0$ specifies the execution time of a process p in each of the scenarios. The function $R_k : (\mathcal{I}_k \cup \mathcal{O}_k \cup \mathcal{C}_k) \times \mathcal{S}_k \to \mathbb{N}_0$ specifies, per scenario, the rates of a kernel k. The function $R_d : (\mathcal{I}_d \cup \mathcal{O}_d) \times \mathcal{S}_d \to \mathbb{N}_0$ gives the rates of a detector d. The function R_d does not depend on a set of control ports $\mathcal{C}_d$, since detectors have no control ports (i.e. $\mathcal{C}_d = \emptyset$). The sequences of tokens that detector d produces when operating in scenario s onto an output port $o \in \mathcal{O}_d$ connected to a control channel b is denoted by $t_d(o, s)$ and is in $\sum_b^*$.

5 Formal Definition of the FSM-based SADF MoC

To define the FSM-based SADF MoC, we first introduce the functions ϕ and ψ that capture the status of data and control channels.

Definition 1 (Channel Status) *Let $\mathcal{B}$ denote the set of all channels of an FSM-based SADF graph, where $\mathcal{B}_c \subseteq \mathcal{B}$ is the set of control channels. A channel status is a function $\phi : \mathcal{B} \backslash \mathcal{B}_c \to \mathbb{N}_0$ that returns the number of tokens stored in the buffer of each channel.*

Notice that a channel status abstracts from the values of tokens.

Definition 2 (Control Status) *Let $\mathcal{B}_c$ denote the set of all control channels of an FSM-based SADF graph. A control status is a function $\psi : \mathcal{B}_c \to \bigcup_{c \in \mathcal{B}_c} \sum_c^*$ that returns the sequence of tokens stored in each control channel.*

Using the channel and control status definitions, an FSM-based SADF graph is defined as follows.

Definition 3 (FSM-based SADF graph) *An FSM-based SADF graph is described by a tuple $G = (\mathcal{K}, \mathcal{D}, \mathcal{B}, \phi^*, \psi^*)$, where*

1. *$\mathcal{K}$ and $\mathcal{D}$ are pairwise disjoint finite sets of kernels and detectors respectively such that $\mathcal{K} \cup \mathcal{D} = \emptyset$;*
2. *there is one detector, such that $|\mathcal{D}| = 1$;*
3. *each kernel $k \in \mathcal{K}$ denotes a tuple $(\mathcal{I}_k, \mathcal{O}_k, \mathcal{C}_k, \mathcal{S}_k, R_k, E_k)$ with*
   - *one control port in the set $\mathcal{C}_k$, such that $|\mathcal{C}_k| = 1$,*
   - *$\mathcal{S}_k = \prod_{c \in \mathcal{C}_k} \sum_b$ with b the control channel connected to c,*
   - *for each scenario $s \in \mathcal{S}_k$, $R_k(c, s) = 1$ for the control port $c \in \mathcal{C}_k$;*
4. *a detector $d \in D$ is a tuple $(\mathcal{I}_d, \mathcal{O}_d, R_d, E_d, t_d, \mathcal{S}_d, \mathbb{S}_d, \mathbb{T}_d, \iota_d, \Phi_d)$ with*
   - *all output ports $o \in \mathcal{O}_d$ connected to a control channel,*
   - *for each scenario $s \in \mathcal{S}_d$, $R_d(o, s) > 0$ and $t_d(o, s) \in \{v \in \sum_b^* \mid |v| = R_d(o, s)\}$ for all output ports $o \in \mathcal{O}_d$ that connect d to a control channel b,*
   - *the tuple $(\mathbb{S}_d, \mathbb{T}_d, \iota_d, \Phi_d)$ denotes a FSM, where $\mathbb{S}_d$ is a non-empty set of states, the function $\mathbb{T}_d : \mathbb{S}_d \to \mathbb{S}_d$ defines the state transitions, $\iota_d \in \mathbb{S}$ denotes the initial state and the function $\Phi_d : \mathbb{S}_d \to \mathcal{S}_d$ associates a scenario with every state in the FSM;*
5. *the set of input (output) ports $\mathcal{I}_G$ ($\mathcal{O}_G$) of G, is defined as the union of all input (output) ports of the kernels $k \in \mathcal{K}$ and the detector $d \in \mathcal{D}$ in G, and the set of control ports $\mathcal{C}_G$ of G, is defined as the union of all control ports of the kernels $k \in \mathcal{K}$ in G;*

6. $\mathcal{B} \subseteq \mathcal{O}_G \times (\mathcal{I}_G \cup \mathcal{C}_G)$ *is the finite set of channels with* $\mathcal{B}_c = \mathcal{B} \backslash (\mathcal{O}_G \times \mathcal{I}_G)$ *the set of control channels. A channel includes an unbounded FIFO buffer and uniquely connects an output port to either an input port or a control port. Furthermore, each port in* $\mathcal{I}_G \cup \mathcal{O}_G \cup \mathcal{C}_G$ *is uniquely connected to one channel;*

7. $\phi^* : \mathcal{B} \backslash \mathcal{B}_c \to \mathbb{N}_0$ *is the initial channel status;*

8. $\psi^* : \mathcal{B}_c \to \bigcup_{c \in \mathcal{B}_c} \sum_c^*$ *indicates the initial control status.*

6 Operational Semantics

The operational semantics of an FSM-based SADF graph is defined as a labelled transition system. A labelled transition system describes behaviour in terms of *configurations* (states) and *transitions*, where timeless actions are explicitly distinghuised from advancing time. We start with defining a configuration in our labelled transition system. For this purpose, we introduce the kernel and dectector status.

Definition 4 (Kernel Status) *A kernel status for an FSM-based SADF graph* $(\mathcal{K}, \mathcal{D}, \mathcal{B}, \phi^*, \psi^*)$ *is a function* κ *that assigns to each kernel* k *a multiset with a pair* $(\mathcal{S}_k \times \mathbb{N}_0)$ *for each ongoing firing of* k*. The first item in the pair denotes the scenario in which this firing of* k *operates. The second item denotes the remaining execution time of this firing of* k*.*

Definition 5 (Detector Status) *A detector status for and FSM-based SADF graph* $(\mathcal{K}, \mathcal{D}, \mathcal{B}, \phi^*, \psi^*)$ *is a function* $\delta : \mathcal{D} \to \mathbb{S} \times (\mathbb{N}_0 \cup \{-\})$ *that associates with detector* $d \in \mathcal{D}$ *the state of the FSM, and the remaining execution time of the ongoing firing of* d *or* $-$ *in case there is no ongoing firing of* d*.*

Definition 6 (Configuration) *A configuration of an FSM-based SADF graph is a tuple* $(\phi, \psi, \kappa, \delta)$ *denoting a channel status, control status, kernel status and detector status respecively. An FSM-based SADF graph* $(\mathcal{K}, \mathcal{D}, \mathcal{B}, \phi^*, \psi^*)$ *has initial configuration* $C^* = (\phi^*, \psi^*, \kappa^*, \delta^*)$*, where the initial kernel status* κ^* *is defined by* $\kappa^*(k) = \{\}$ *for all* $k \in \mathcal{K}$ *(with* $\{\}$ *denoting the empty multiset) and the initial detector status* δ^* *is defined by* $\delta^*(d) = (\iota_d, -)$ *for all* $d \in \mathcal{D}$*.*

The dynamic behavior of an FSM-based SADF graph is described by transitions. Five different types are distinguished: kernel start action, kernel end action, detector start action, detector end action, and time progress.

Definition 7 (Kernel Start Action Transition) *A kernel start action transition* $(\phi, \psi, \kappa, \delta) \overset{start(k)}{\to} (\phi', \psi', \kappa', \delta')$ *denotes the start of a firing of kernel* k*. It is enabled when* $|\psi(b_c)| \geq 1$ *for the channel* b_c *connected to the control port* $c \in \mathcal{C}_k$ *and* $\phi(b_i) \geq R_d(\psi(b_c)_1, i)$ *for each channel* b_i *connected to an input port* $i \in \mathcal{I}_k$*. the scenario* s *in which this firing of* k *operates is* $\psi(b_c)_1$*, with* b_c *the channel connected to the control port* $c \in \mathcal{C}_k$*. The resulting configuration* $(\phi', \psi', \kappa', \delta')$ *is given by:* $\phi' = \phi[b_i \to \phi(b_i) - R_k(s, i)]$ *for each channel* b_i *connected to an input port* $i \in \mathcal{I}_k$*,* $\psi' = \psi[b_c \to \psi(b_c) - \psi(b_c)_1]$ *for channel* b_c *connected to the control port* $c \in \mathcal{C}_k$*,* $\kappa' = \kappa[k \to \kappa(k) \uplus \{(s, E_k(s))\}]$*, and* $\delta' = \delta$ *(where* $\uplus$ *denotes multiset union).*

Note that token consumptions in the FSM-based SADF model occur at the start action transition. The operational semantics of the general SADF model consumes these tokens during the end action transition.

Definition 8 (Kernel End Action Transition) *A kernel end action transition* $(\phi, \psi, \kappa, \delta) \overset{end(k)}{\to} (\phi', \psi', \kappa', \delta')$ *denotes the end of a firing of kernel* k*. It is enabled when there exists a* $(s, 0) \in \kappa(k)$ *for some scenario* $s \in \mathcal{S}_k$*. The resulting configuration* $(\phi', \psi', \kappa', \delta')$ *is given by:* $\phi' = \phi[b_o \to \phi(b_o) + R_k(s, o)]$ *for each channel* b_o *connected to an output port* $o \in \mathcal{O}_k$*,* $\psi' = \psi$*,* $\kappa' = \kappa[k \to \kappa(k) \backslash \{(s, 0)\}]$*, and* $\delta' = \delta$*.*

Definition 9 (Detector Start Action Transition) *A detector start action transition* $(\phi, \psi, \kappa, \delta) \overset{start(d)}{\to} (\phi', \psi', \kappa', \delta')$ *denotes the start of a firing of detector* d*. It is enabled when it holds that* $\delta(d) = (s, -)$ *for some state* $s \in \mathbb{S}_d$ *and* $\phi(b_i) \geq R_d(\Phi_d(s), i)$ *for each channel* b_i *connected to an input port* $i \in \mathcal{I}_d$*. The resulting configuration* $(\phi', \psi', \kappa', \delta')$ *is given by:* $\phi' = \phi[b_i \to \phi(b_i) - R_d(\Phi_d(s), i)]$ *for each channel* b_i *connected to an input port* $i \in \mathcal{I}_d$*,* $\psi' = \psi$*,* $\kappa' = \kappa$*, and* $\delta' = \delta[d \to (s, E_d(\Phi_d(s)))]$*.*

Note that in contrast to the general SADF model, the number of tokens consumed by a detector start action transition in the FSM-based SADF model depends on the current state of the FSM and not on the next state.

Definition 10 (Detector End Action Transition) *A detector end action transition* $(\phi, \psi, \kappa, \delta) \overset{end(d)}{\rightarrow} (\phi', \psi', \kappa', \delta')$ *denotes the end of a firing of detector* d*. It is enabled when it holds that* $\delta(d) = (s, 0)$ *for some state* $s \in \mathbb{S}_d$*. The resulting configurations* $(\phi', \psi', \kappa', \delta')$ *are given by:* $\phi' = \phi$*,* $\psi' = \psi[b_c \rightarrow \psi(b_c) + t_d(\Phi_d(s))]$ *for each channel* b_o *connected to an output port* $o \in \mathcal{O}_d$*,* $\kappa' = \kappa$*, and* $\delta' = \delta[d \rightarrow (\mathbb{T}_d(s), -)]$*.*

Definition 11 (Time Transition) *A time transition* $(\phi, \psi, \kappa, \delta) \overset{time(t)}{\rightarrow} (\phi', \psi', \kappa', \delta')$ *denotes progress in time with* t *units. It is enabled if no kernel end transition and no detector end transition are enabled. Time transitions can occur for any submultiset of executing processes and nondeterministically results in* $\phi' = \phi$*,* $\psi' = \psi$*,* $\kappa' = \{(p, M) \mid p \in \mathcal{K}\}$ *with* M *equal to* $\kappa(k)$ *with some arbitrary submultiset of its members decremented by* t*, and* $\delta' = \{(d, M) \mid d \in \mathcal{D}\}$ *with* M *equal to* $\delta(d)$ *with some arbitrary subset of its members decremented by* t*.*

A time transition decreases the remaining execution time of an arbitrary submultiset of ongoing kernel and detector firings with t time-unit. The ability to select different submultisets in different time transitions makes it possible to take resource constraints (e.g. static-order schedules or TDMA schedules) into account.

References

[1] S.V. Gheorghita. *Dealing with dynamism in embedded system design: application scenarios.* PhD thesis, TU Eindhoven, December 2007.

[2] S.V. Gheorghita, M. Palkovic, J. Hamers, A. Vandecappelle, S. Mamagkakis, T. Basten, L. Eeckhout, H. Corporaal, F. Catthoor, F. Vandeputte, and K. De Bosschere. Methods for evaluating and covering the design space during early design developmenta system scenario based approach to dynamic embedded systems. *ACM Transactions on Design Automation of Electronic Systems*, 2009. (To be published in 2009).

[3] E.A. Lee and D.G. Messerschmitt. Static scheduling of synchronous data flow programs for digital signal processing. *IEEE Transactions on Computers*, 36(1):24–35, January 1987.

[4] B.D. Theelen, M.C.W. Geilen, T. Basten, J.P.M. Voeten, S.V. Gheorghita, and S. Stuijk. A scenario-aware data flow model for combined long-run average and worst-case performance analysis. In *4th International Conference on Formal Methods and Models for Co-Design, MEMOCODE 06, Proceedings.*, pages 185–194. IEEE, July 2006.

[5] B.D. Theelen, M.C.W. Geilen, S. Stuijk, S.V. Gheorghita, T. Basten, J.P.M. Voeten, and A.H. Ghamarian. Scenario-aware dataflow. Technical Report ESR-2008-08, TU Eindhoven, 2008.