Timing Analysis Driven Design-Space Exploration of Cause-Effect Chains in Automotive Systems

Matthias Becker
KTH Royal Institute of Technology, Sweden
mabecker@kth.se

Saad Mubeen
Mälardalen University, Sweden
saad.mubeen@mdh.se

***Abstract*—Model-based development and component-based software engineering have emerged as a promising approach to deal with enormous software complexity in automotive systems. This approach supports the development of software architectures by interconnecting (and reusing) software components (SWCs) at various abstraction levels. Automotive software architectures are often modeled with chains of SWCs, also called cause-effect chains that are constrained by timing requirements. Based on the variations in activation patterns of SWCs, a single model of a cause-effect chain at a higher abstraction level can conform to several valid refined models of the chain at a lower abstraction level, which is closer to the system implementation. As a consequence, the total number of valid implementation-level models generated by the existing techniques increases exponentially, thereby significantly increasing the runtime of the timing analysis engines and liming the scalability of the existing techniques. This paper computes an upper bound on the activation pattern combinations that may result from a system of cause-effect chains in a given high-level model of the software architecture. An efficient algorithm is presented that traverses only a reduced number of possible combinations of the cause-effect chains, resulting in the timing analysis of a significantly lower number of implementation-level models of the software architecture. A proof of concept is provided by conducting a case study that shows significant reduction in the runtime of timing analysis engines, i.e., the timing behavior of the considered system is verified by performing the timing analysis of only 27% of all possible combinations of the cause-effect chains.**

I. Introduction

Embedded software development in modern automotive systems is very challenging due to continuous increase in software size and complexity. Already today, the size of the software is in the range of 100 million lines of code, which is distributed over 50 to 120 on-board computers, called Electronic Control Units (ECUs) [1], [2]. In addition to the software complexity and complex coordination among the ECUs, many automotive systems are subject to stringent timing requirements that must be verified during the development and before running the system. Model-Based Development (MBD) [3] and Component-Based Software Engineering (CBSE) [4] are proving effective in dealing with the software complexity, e.g., AUTOSAR [5] and RCM [6] are two examples of the component models that employ the principles of MBD and CBSE. In addition, the multi-abstraction development methodologies such as EAST-ADL [7] are able to lower the complexity by allowing the software development at various abstraction levels (corresponding to development phases), e.g., *design* and *implementation* levels. The design level represents an earlier phase during the development, where a high-level software architecture of the system is modeled. Whereas, the implementation level represents a later phase during the development containing concrete implementation of the software architecture, which is closer to the system implementation. The challenge of pre-runtime verification of timing requirements at various abstraction levels can be met by integrating schedulability analysis techniques [8], [9] with the component models for the automotive software development.

A. Problem Statement and Related Work

Software architectures in automotive embedded systems are often modeled with Software Component (SWC) chains that originate by reading sensor signals and terminate by producing actuator signals, e.g., five different SWC chains are shown in Fig. 1. These chains are also referred to as cause-effect chains. Within a cause-effect chain, the data is always shared between any two neighboring SWCs as it can be seen in all the chains in Fig. 1. However, a SWC may receive an activation trigger either from an independent trigger source (e.g., SWC_2 in Fig. 1(b) and Fig. 1(d)) or from its predecessor SWC (e.g., SWC_2 in Fig. 1(c) and Fig. 1(e)). In the top-down development process for automotive embedded systems, such as the one described by EAST-ADL [7], there is no clear separation between the trigger and data flows along the chain at the earlier development phases. For example, Fig. 1(a) shows a chain at the design level (earlier development phase), where the data flows from the sensor to the actuator via SWC_1 and SWC_2 by means of flow ports (data ports). At the later development phases, such as the implementation level, the data flow is separated from the trigger flow in the chains. For example, trigger information about each SWC in the four cause-effect chains shown in Fig. 1(b)-(e) are explicitly modeled.

At the design level, the system designer has to make an important decision about modeling the trigger flows along the chain, i.e, whether to activate each SWC by an independent trigger source (periodic clock) or by the predecessor SWC. This decision has a significant impact on the timing behavior of the chain because the data-propagation delays from the input to the output of the chain is affected by the trigger pattern along the chain. This can be demonstrated by applying any existing data-propagation delay analysis such as [10], [11], [12], [13]. The effect of periods on data-propagation delays is investigated in [14], [15] by leveraging the analysis in the above-mentioned works. Another set of previous works [11], [12] focus on generating job-level dependencies to meet the data-propagation delay constraints. In [16], a set of priorities, offsets and task-processor mapping is selected with the goal of meeting data-propagation delay constraints.

So far, only few techniques exist that focus on the trigger selection along the cause-effect chains. The works in [17], [18] automatically help the designer in selecting the most suitable trigger patterns in the chain with respect to the data-propagation delay constraints. The existing techniques, in the first step, generate all valid implementation-level models of the chain, e.g., four implementation-level models shown in Fig. 1(b)-(e) are generated from the design-level model of the chain in Fig. 1(a) based on various trigger patterns in the chain. In the second step, all implementation-level models are timing analyzed, the analysis results are verified against the

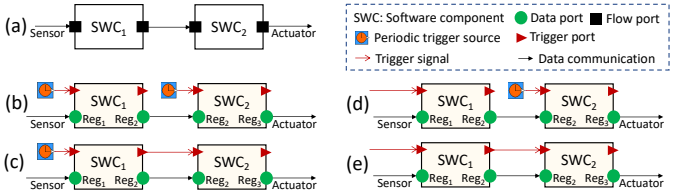

Fig. 1: (a) Example of a two-component chain at the design level, (b) Four equivalent implementation-level models based on various trigger patterns in the chain in (a).

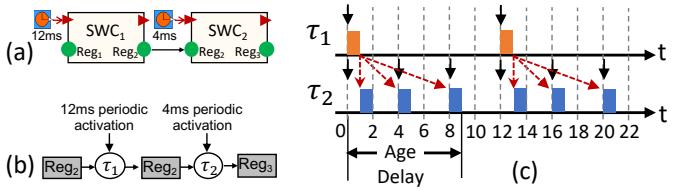

Fig. 2: (a) Two-SWC chain at the design level, (b) runtime model of the chain, (c) data-propagation delay in the chain.

timing requirements, and the most suitable implementation-level model is presented to the designer.

There are several limitations in the existing techniques. First, the computational complexity is exponential because, in general, 2^N implementation-level models need to be generated and timing analyzed for every cause-effect chain, where N represents the number of SWCs along the chain. This hampers with the applicability of the existing techniques to large and complex systems, thereby limiting the scalability. Second, the existing techniques have been applied only to the cause-effect chains, where SWCs in the chains are triggered by the trigger sources with same periods. There is a lack of guidelines for generating all valid implementation-level models in the case of multi-rate cause-effect chains, where SWCs in the chains are triggered by trigger sources with different periods. Without such a support, non-determinism can be introduced when generating the implementation-level models from a single design-level model of the chain. For example, assume that SWC_1 and SWC_2 in the implementation-level model shown in Fig. 1(b) are triggered by 8 ms and 4 ms periodic clocks respectively. When the implementation-level model in Fig. 1(c) is generated, the existing techniques do not provide any guidance for selecting the trigger period of SWC_2, which is triggered by its predecessor SWC_1. Should SWC_2 receive a periodic trigger with 4 ms period or a sporadic trigger with 8 ms (inherited from its predecessor)? This ambiguity introduces non-determinism in the implementation-level models of multi-rate chains. Third, the existing techniques focus on a single cause-effect chain with no guidelines for generating all possible models in the system of cause-effect chains.

B. Paper Contribution

The main contributions of this paper are as follows.

- The possible trigger sources for SWCs (design-time entities) or tasks[1] (runtime entities) in the cause-effect chains are identified in a way that ambiguity in the described model is avoided. This is done by the concept of chain segments.
- Computation of an upper bound on the activation patterns for a system of cause-effect chains that need to be analyzed.
- The main contribution of this paper is an efficient algorithm that utilizes the idea of activation pattern subsets, i.e., sets of activation patterns that have the same data-propagation delay for a certain cause-effect chain. This allows to reduce the number of activation patterns that need to be timing analyzed to verify the timing behavior of the system, thereby reducing runtime of the timing analysis engines. The generated set of activation patterns correspond to a small subset of all valid implementation-level models of a single design-level model of the software architecture.
- A case-study demonstrates the proof of concept for the proposed algorithm, where the algorithm runtime is reduced to 27% of the worst-case runtime by the existing techniques.

[1]A one-to-one mapping is considered between a task and SWC.

II. System Model

This section describes the system model. The application is run on a single-core single-processor platform under Fixed Priority Scheduling [19] and Rate Monotonic priority assignment. Tasks that have the same assigned period have the same priority and are scheduled in Round Robin manner.

A. Task Model

We consider an application that consist of a set of SWCs. Each SWC is directly mapped to a task τ. Γ describes the complete task-set, where each task $\tau_i \in \Gamma$ can be described by the tuple $\{C_i, T_i, D_i\}$. C_i describes the task's Worst-Case Execution Time (WCET), T_i describes its minimum inter-arrival time, and D_i describes the relative deadline of the task. In this work, we assume that tasks are subject to implicit deadlines, i.e. the deadline D_i of a task is equal to its period T_i. Tasks themselves exchange data using *register communication*. This means, a sending task writes to a shared register (i.e. a global variable) and a receiving task reads from the shared register. For example, two tasks τ_1 and τ_2 share register Reg_2 as shown in Fig. 2(b). This way of communication leaves both tasks independent of each other's execution, but also leads to over- and under-sampling situations. Oversampling occurs when the receiving task is activated with a smaller period than the sending task. The receiving task reads the shared value more often than the sending task updates it, which leads to multiple reads of the same data as shown in Fig. 2(c). Similarly, under-sampling occurs when the period of the sending task is smaller than the receiving task's period. In this case, some values that are written to the shared register are never read by the receiving task.

B. Cause-Effect Chain

Cause-Effect Chains are specified on semantically related tasks and describe the way data propagates through the system. The end-to-end propagation delay of a cause-effect chain can be subject to timing constraints, where different delay semantics are possible [12]. In general, a cause-effect chain ζ is a Directed Acyclic Graph (DAG), where the vertices representing tasks $\in \Gamma$, and the edges of the DAG represent the data propagation through the cause-effect chain. All cause-effect chains that are specified on the application are in the set $\mathcal{Z}$. Note that a cause-effect chain can have forks and joins, but only one start and end vertex can exist. This means, a cause-effect chain with multiple paths can be seen as a number of single-path cause-effect chains that share the same timing constraints. Thus, without loss of generality, in the following all cause-effect chains are sequential.

A sporadic trigger can be defined for the cause-effect chain. This trigger can then be used to activate the initial task in the cause-effect chain. Note that this trigger source represents a trigger option for the system but only at a later point the exact trigger source is selected. A cause-effect chain that is initiated by arriving network messages can for example be triggered by

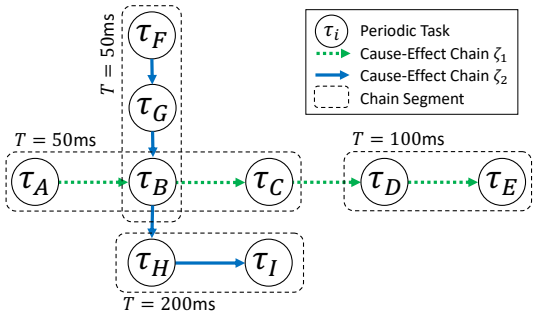

Fig. 3: Running example: two connected cause-effect chains and the different chain segments.

a sporadic event if the message arrival initiates the first task in the cause-effect chain, or by a periodic clock if the messages are received based on polling.

Cause-effect chains in industrial applications include tasks of different activation periods, however, as stated in [20], cyclic dependencies between periods do not exist. I.e. tasks of same period appear consecutively in a cause-effect chain. We call such a sequence of same period tasks a *chain segment* Ψ. A chain segment Ψ hence is represented as a list of tasks $\{\tau_1, \tau_2, \ldots, \tau_k\}$.

The chain itself can then be represented as a list of n chain segments $\zeta = \{\Psi_1, \Psi_2, \ldots, \Psi_n\}$.

Fig. 3 introduces a running example, consisting of two cause-effect chains that in total contain nine tasks (shown by dotted green arrows and full blue arrows respectively). The chain segments are represented by dashed boxes, where each task within a box has the same prescribed period. Note that both cause-effect chains include the task τ_B. In this example, there is no sporadic trigger specified for the cause-effect chains, and both cause-effect chains have a specified maximum data age constraint of $100\,\mathrm{ms}$ (not shown in the figure).

C. Task Activation

While each task has an assigned period, the activation trigger is not yet determined. In general, a task can either be triggered by a periodic clock (where the period is equal to the tasks initial minimum inter-arrival time), or by the termination of another task. In general, for a cause-effect chain, each task can be triggered either by the periodic clock or by it's predecessor task. However, to avoid inconsistency (as discussed in Sec. I-A) the first task in a chain segment can not be triggered by a predecessor task in the same chain.

In a cause-effect chain, the activation of a task by its predecessor is beneficial as it introduces an execution ordering between successive tasks' instances in the cause-effect chain which reduces the experienced worst-case data-propagation delays. On the other hand, it restricts the scheduling decisions and may render task sets unschedulable. For the running example, this means that τ_A, τ_F, τ_D, and τ_H need to be triggered by a periodic clock, whereas all other tasks may additionally be triggered by their predecessors.

D. Timing Analysis

The timing analysis considered in this paper corresponds to the state-of-the-art data-propagation delay analysis [10], which has been implemented in several industrial tools such as SymTA/S [21] and Rubus-ICE [9]. The analysis can calculate various types of data-propagation delays in the cause-effect chains, e.g., Fig. 2 (c) highlights one such delay, called the age delay, in a cause-effect chain that consists of two SWCs shown in Fig. 2 (a). This means, the two-SWC chain in Fig. 2 (a) corresponds to the two-task chain at run-time in Fig. 2 (b). The tasks τ_1 and τ_2 are activated independently with periods of 12 ms and 4 ms respectively. Assuming the priority of τ_1 to be higher than τ_2, the data can propagate from the input (Reg_1) to the output of the chain (Reg_3) through three different paths identified by the red dotted arrows between τ_1 and τ_2 in Fig. 2 (c). The data age delay is equal to the delay in the longest path among all these paths from the input to the output of the chain. It is interesting to note that the calculations for the age delay depend only on the activation times of the first and last tasks and response time of the last task in the longest path in the chain [10]. The response times of the tasks are calculated using the state-of-the-art response-time analysis [19], [22], [23].

III. Computation of Possible Activation Patterns

A. Single Cause-effect Chain

The number of tasks in one chain segment Ψ_i is denoted as $|\Psi_i|$. The number of possible activation pattern combinations α_i that need to be checked can be expressed as:

$$\alpha_i = 2^{|\Psi_i - 1|} \tag{1}$$

While the first task in a chain segment is always triggered by an independent periodic clock, the activation of all consecutive tasks in the chain segment may either be triggered by a independent periodic clock or by the predecessor task.

Based on Equation 1, the total number of activation pattern combinations in a cause-effect chain ζ_j can be expressed as:

$$\beta_j = \begin{cases} 2 \cdot \prod_{\forall \Psi_i \in \zeta_j} \alpha_i & \text{if } \zeta_i \text{ has sporadic trigger} \\ \prod_{\forall \Psi_i \in \zeta_j} \alpha_i & \text{otherwise} \end{cases} \tag{2}$$

The initial factor of 2 needs to be added if a sporadic trigger is specified for the cause-effect chain and thus the first task can be activated either by the period clock or the sporadic trigger.

ζ_1 in the running example includes two chain segments, τ_A, τ_B, and τ_C in the first segment, and τ_D and τ_E in the second. Thus, $\alpha_1 = 4$ and $\alpha_2 = 2$. The total number of possible activation patterns β is computed as: $\beta_1 = \alpha_1 \cdot \alpha_2 = 8$. As the second cause-effect chain has the same pattern as the first cause-effect chain, also $\beta_2 = 8$.

In contrast, an approach that does not consider the chain segments would compute $\beta_1 = 16$ and $\beta_2 = 16$ activation pattern combinations that then need to be checked for the respective cause-effect chain.

B. Multiple Cause-effect Chains

If a system contains n cause-effect chains, the same task may be part of multiple such chains. A straight forward way to compute all possible combinations in the system is obtained by multiplying the combinations of each cause-effect chain:

$$\beta_\Gamma = \prod_{\forall \zeta_i \in \mathcal{Z}} \beta_i \tag{3}$$

For the running example this would mean that there are $\beta_\Gamma = \beta_1 \cdot \beta_2 = 8 \cdot 8 = 64$ possible combinations (the simple approach [17] that does not consider the chain segments would even lead to $\beta_\Gamma = 256$ combinations). However, in a system where two cause-effect chains share a common task, this leads to an over-approximation of the number of valid combinations that need to be checked. This is the case because a task which is common in more than one cause-effect chain is considered in every chains' set of possible activation patterns. Hence, Eq. 3 accounts for cases in which a shared task is triggered

by more than one predecessor. In our running example these are all combinations where τ_B is triggered by both, τ_A and τ_F. According to the system model this is not possible.

For each task $\tau_i \in \Gamma$ we define the set Φ_i denoting all sporadic triggers that are options for this task resulting from the system model. A sporadic trigger is added to Φ_i in the following cases:

1) If τ_i is the initial task of a cause-effect chain and there is a sporadic trigger source defined for the cause-effect chain, then this sporadic trigger is added to Φ_i.
2) Each predecessor task of τ_i in a cause-effect chain is part of Φ_i, if the two tasks are in the same activation pattern.

Note that each trigger can be included in Φ_i at most one time.

In order to get the number of all possible valid combinations the product of all tasks' total number of possible activation sources needs to be computed (Φ_i triggers by other task plus one trigger through the periodic clock).

$$\beta_\Gamma = \prod_{\forall \tau_i \in \Gamma} (\Phi_i + 1) \tag{4}$$

Again we look at the running example. Tasks τ_A, τ_D, τ_F, and τ_H are all first tasks in their respective chain segments and thus $\Phi_A = \Phi_D = \Phi_F = \Phi_H = 0$. Tasks τ_C, τ_E, τ_G, and τ_I are part of only one cause-effect chain which leads to $\Phi_C = \Phi_E = \Phi_G = \Phi_I = 1$. Finally, τ_B is part of both cause-effect chains which leads to $\Phi_B = 2$. With these values, β_Γ results in only 48 combinations of activation patterns that need to be verified.

To improve the readability, in the remainder of the paper we refer to β_Γ as the set of all activation patterns[2].

IV. Verifying Data-propagation Delay of the System under all Generated Activation Patterns

In this section we present an algorithm that generates all possible valid combinations of activation patterns such that all timing constraints are met. To overcome the combinatorial complexity of all possible settings that may be valid candidates and thus need to be tested, we identify the property of *activation pattern subset* and show how it can be applied to efficiently generate all schedulable activation pattern combinations without exhaustively verifying all of them.

A. Activation Pattern Subset

In Section III-B we have shown how to generate the set β_Γ of all possible candidate activation patterns for a system of cause-effect chains. To verify all timing constraints, timing analysis of the data-propagation delay needs to be performed for each cause-effect chain with each activation pattern in β_Γ.

However, in systems that contain multiple cause-effect chains, several activation patterns in β_Γ are redundant to check for a cause-effect chain ζ_k, as each activation pattern leads to the same result of the data-propagation delay analysis. We call these subsets $\delta \in \beta_\Gamma$ *activation pattern subset*. For a given cause-effect chain ζ_k, two activation patterns $p \in \beta_\Gamma$ and $p' \in \beta_\Gamma$ are in the same activation pattern subset δ_j^k, if:

1) $\forall \tau_i \in \zeta_k$ the activation in p and p' is the same.
2) $\forall \tau_i \notin \zeta_k$ that is in *trigger relation* to a task $\in \zeta_i$ and has a specified trigger (i.e. not clock driven) the same trigger is specified in p and p'.
3) $\forall \tau_i \notin \zeta_k$ that have no specified trigger in p and a specified trigger in p' have no *trigger relation* to a task of ζ_k.

[2]The concrete computation of all combinations is left out due to space limitations but can be done in a straight-forward way.

In the following we show that the above statement holds.

Lemma 1: When analyzing the maximum data-propagation delay of a cause-effect chain ζ_k, all activation patterns that are part of δ_j^k result in the same maximum data-propagation delay.

Proof 1: For the data-propagation delay of a cause-effect chain, either the start or the release time of the involved tasks has to change [10]. Hence, we have to show that the start and the release time of tasks in a cause-effect chain does not change for different activation pattern that are part of the same activation pattern subset δ_j^k. For two activation patterns p and p' of the same activation pattern subset it holds that if tasks that have different trigger specified in p and p', then these trigger do not directly or indirectly activate a task of the cause-effect chain. I.e. by definition the resulting release time of the tasks in the chain are the same.

The interference experienced by the tasks in the chain also have to be identical in p and p' for them to be in the same activation pattern subset. We know that tasks that can have a trigger relation to any task, they must have the same initial period specified, i.e. they also have the same priority assigned. Thus, as long as there is no direct or indirect trigger path to a task of the chain, the timing analysis has to consider that all tasks arrive just before the task under analysis. Hence, they are considered as interference (regardless of the trigger relations amongst each other). Hence, also the response time of all tasks in a cause-effect chain are the same for p and p'. □

B. Relevant Activation Pattern Combinations

In order to find all possible activation pattern combinations that result in a maximum data age of a cause-effect chain ζ_k which is smaller than the specified timing constraint, the concept of activation pattern subset can be applied.

As each of the activation patterns in one subset results in the same maximum data-propagation delay for ζ_k, only one candidate out of each activation pattern subset of ζ_k needs to be analyzed to verify ζ_k's timing constraint. We define the function $\texttt{getRelevantPatterns}(\zeta_\texttt{k}, \beta_\Gamma)$ to return a set of activation patterns that include one candidate pattern for each activation pattern subset that can be identified for ζ_k.

If a tested activation pattern $p \in \delta_j^k$ results in the verdict `unschedulable` for the cause-effect chain ζ_k, then it can be removed from all possible candidate combinations in β_Γ for this chain. Due to the property of δ, all other activation patterns that are part of δ_j^k can also be removed from β_Γ. We define the function $\texttt{getSubset}(\texttt{p}, \zeta_\texttt{k}, \beta_\Gamma)$ to return the activation pattern subset δ_j^k which has p as member.

C. Efficient Algorithm to Generate Valid Activation Patterns

The properties of the activation pattern subset can be used while traversing the search space of all β_Γ possible activation pattern combinations (see Sec. III-B) for each cause-effect chain $\in \mathcal{Z}$. Algorithm 1 presents the pseudocode of such an algorithm. The algorithm receives the set of all cause-effect chains $\mathcal{Z}$, the set of all tasks Γ, and the set of all possible activation patterns β_Γ. The objective of the algorithm is to return a subset of β_Γ, where each activation pattern p yields a schedulable system (i.e. all specified timing constraints, both on tasks but also on cause-effect chains, are met). This is represented by the set β^{out} which initially holds all activation patterns of β_Γ (line 2).

The algorithm has to verify the timing properties for each cause-effect chain (line 3). For each cause-effect chain, one candidate pattern of all activation pattern subsets for the chain

Algorithm 1: GenerateValidActivationPattern($\mathcal{Z}, \Gamma, \beta_\Gamma$)

```
1  begin
2      β^out = β_Γ;
3      for ∀ζ_i ∈ Z do
4          β_i^out = β_Γ;
5          β_ζi = getRelevantPatterns(ζ_i, β_Γ);
6          for ∀p ∈ β_ζi do
7              if Analysis(ζ_i, p, Γ) ≠ schedulable then
8                  β_i^out = getSubset(p, ζ_i, β_Γ);
9                  β^out = β^out − β_i^out;
10     return β^out
```

needs to be analyzed. In line 5, the set of activation pattern candidates is obtained. Each of the patterns is then analyzed (line 7). If the analysis result indicates that a timing constraint is violated, i.e. the result `unschedulable` is returned, the activation pattern subset to which p belongs is computed and subsequently removed from the output set β^{out}. As this is done for each cause-effect chain, the set β^{out} is gradually reduced by all activation patterns that violate the constraints on the cause-effect chains. Thus, after all cause-effect chains are analyzed, the remaining patterns are the ones that do not violate timing constraint of any cause-effect chain in the system.

V. Case Study

A case-study is used to demonstrate the applicability of the proposed approach. An application with 15 different tasks is considered (τ_A to τ_O). The parameters of each task are shown in Table I, additionally the predecessor and successor tasks of each task are shown. There are two cause-effect chains in the application, denoted by ζ_1 and ζ_2. ζ_1 is specified as: $\tau_A \rightarrow \tau_B \rightarrow \tau_C \rightarrow \tau_D \rightarrow \tau_E$. And ζ_2 is specified as: $\tau_E \rightarrow \tau_F \rightarrow \tau_B \rightarrow \tau_H \rightarrow \tau_I$. Each cause-effect chains has a specified maximum data age constraint of $100\,\mathrm{ms}$. Note that these are the cause effect chains shown in Fig. 3. Thus, the maximum number of trigger patterns that need to be checked for each cause-effect chain in order to provide the system designer with all combinations which result in a schedulable system is $\beta_\Gamma = 48$, according to Sec. III-B.

A. Pattern Generation

First we demonstrate the reduction in search space obtained by the function `getRelevantPattern` that is based on Lemma 1. Fig. 4 shows all trigger pattern $\in \beta_\Gamma$, where tasks that are part of a cause-effect chain are shown (tasks not part of a cause-effect chain are activated by a periodic clock and thus do not introduce additional trigger combinations).

For the cause-effect chain ζ_1, the number of activation pattern subsets is 14, thus only 14 out of the 48 activation patterns from the set β_Γ need to be analyzed for ζ_1. This is shown in Fig. 4 by the dashed boxes labeled δ_1^1 to δ_{14}^1. An activation pattern is part of the visualized subset only if a mark is placed in the dashed box below the pattern (either a check for the result schedulable or a cross for the result unschedulable, when testing the respective cause-effect chain). Similarly, the number of activation pattern subsets for cause-effect chain ζ_2 is 12, thus only 12 out of the 48 activation patterns of β_Γ need to be analyzed for ζ_2. The generated subsets that are provided by `getRelevantPattern` for both chains requires only 26 tests in total, instead of 96 (i.e. when testing all 48 pattern of β_Γ for each cause-effect chain). This means, the amount of tested configurations is reduced to only 27% of the exhaustive test. Always the first activation pattern of each activation pattern subset is selected for timing analysis, and all timing analysis results are obtained using Rubus-ICE [9].

TABLE I: Application properties of the case study.

	Period T	**WCET** C	**Predecessor**	**Successor**
τ_A	50 ms	159 µs	-	-
τ_B	50 ms	109 µs	τ_A, τ_B	τ_C, τ_H
τ_C	50 ms	139 µs	τ_B	τ_D
τ_D	100 ms	111 µs	τ_C	τ_E
τ_E	100 ms	179 µs	τ_D	-
τ_F	50 ms	93 µs	-	τ_G
τ_G	50 ms	198 µs	τ_F	τ_B
τ_H	200 ms	103 µs	τ_B	τ_I
τ_I	200 ms	134 µs	τ_H	-
τ_J	50 ms	124 µs	-	-
τ_K	50 ms	182 µs	-	-
τ_L	100 ms	127 µs	-	-
τ_M	10 ms	155 µs	-	-
τ_N	20 ms	159 µs	-	-
τ_O	20 ms	193 µs	-	-

As a result, 24 valid activation pattern are identified for ζ_1, and 23 for ζ_2, as shown in Fig. 4. The intersection of the two sets reported by the algorithm includes 12 activation patterns that result in a schedulable system (i.e. all tasks' deadlines are met and all timing constraints of the cause-effect chains are met as well), this can also be seen in Fig. 4.

B. Runtime

The execution time of the algorithm is one of the key motivating factors for this work. In our experimental setup, the algorithm has a total execution time of $119.62\,\mathrm{s}$. The total execution time is comprised of the timing analysis of ζ_1 and ζ_2, for each identified activation patterns (24 and 12 respectively), plus the execution time of Algorithm 1. The analysis of the data-propagation delay [10], [9] experiences an average analysis time of $4.64\,\mathrm{s}$, with a standard deviation of $0.19\,\mathrm{s}$ for ζ_1. ζ_2 has an average analysis time of $4.55\,\mathrm{s}$ with a standard deviation of $0.17\,\mathrm{s}$. On the other hand, Algorithm 1 experiences an execution time of $11\,\mathrm{ms}$ (max. observed value out of 1000 experiment runs, excluding the timing analysis of the cause-effect chains). In comparison, only the timing analysis of 48 combinations for each cause-effect chain would take $441\,\mathrm{s}$ (the proposed algorithm requires only 27% of this time). Thus, the time to analyse the cause-effect chains contributes the majority of the total runtime. The reduction in execution time due to the limited subset of activation patterns that need to be tested outweighs the additional complexity that is required to find this subset.

VI. Summary and Conclusion

The possibility to represent one model of the high-level software architecture with several lower-level models (that are closer to the system implementation) during the automotive software development process poses challenges for the system developers. Especially when timing properties of data propagation through cause-effect chains are of importance. Changing activation triggers of software components (design-time entities) or tasks (runtime entities) in the cause-effect chains can have significant impact on the timing behavior of the overall system. The large number of possible activation pattern

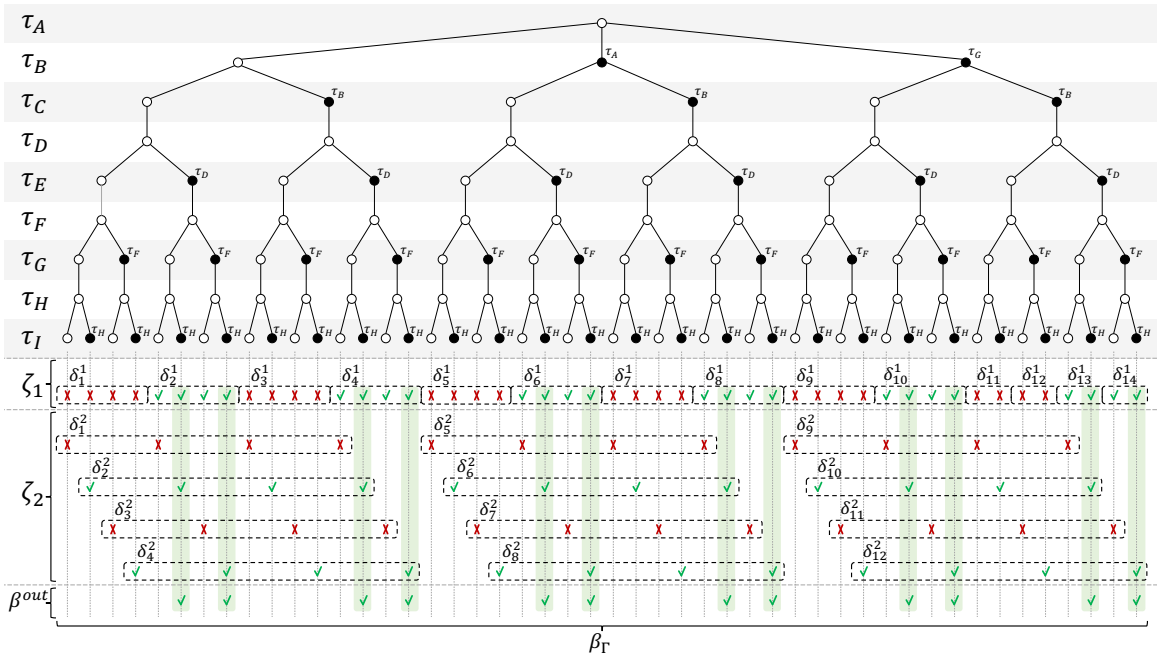

Fig. 4: (Top) Visualization of all activation patterns (tasks not part of the chain are not visualized, but triggered by their periodic clock). (Bottom) the activation pattern subsets for both cause effect chains with the results of the timing analysis.

combinations in a system of cause-effect chains agravates the problem of identifying valid combinations, i.e. a trigger combination that results in a schedulable system. We observe properties of such systems that allow us to test only a limited number of activation patterns, while still covering the whole search space which can reduce the design time significantly by automating such a process. A case study demonstrates this, where all valid activation patterns are computed by the presented algorithm in only 27% of the time compared with the time it takes to test all possible patterns by the existing approaches.

ACKNOWLEDGMENT

This work is supported by the Swedish Knowledge Foundation (KKS) via the projects PreView and DPAC. The work is partially supported by ÅForsk Foundation. We thank all our industrial partners, especially Arcticus Systems, Volvo CE and BAE Systems.

REFERENCES

[1] C. Ebert and J. Favaro, "Automotive software," *IEEE Software*, vol. 34, no. 3, pp. 33–39, May-Jun. 2017.

[2] I. Baas, A glimpse into the future of travel and its impact on marketing, 2016, http://www.thedrum.com/opinion/2016/01/11/glimpse-future-travel-and-its-impact-marketing, accessed July 25, 2018.

[3] T. A. Henzinger and J. Sifakis, "The Embedded Systems Design Challenge," in *14th International Symposium on Formal Methods (FM)*. Springer, 2006, pp. 1–15.

[4] I. Crnkovic and M. Larsson, *Building Reliable Component-Based Software Systems*. Norwood, MA, USA: Artech House, Inc., 2002.

[5] "AUTOSAR Technical Overview, Release 4.1, Rev. 2, Ver. 1.1.0., The AUTOSAR Consortium, Oct., 2013," http://autosar.org.

[6] K. Hänninen et al., "The Rubus Component Model for Resource Constrained Real-Time Systems," in *IEEE Symposium on Industrial Embedded Systems*, 2008.

[7] "EAST-ADL Domain Model Specification, V2.1.12,," http://www.east-adl.info/Specification/V2.1.12/EAST-ADL-Specification_V2.1.12.pdf.

[8] N. Audsley, A. Burns, R. Davis, K. Tindell, and A. Wellings, "Fixed priority pre-emptive scheduling: an historic perspective," *Real-Time Sys.*, vol. 8, no. 2/3, 1995.

[9] S. Mubeen, J. Mäki-Turja, and M. Sjödin, "Support for end-to-end response-time and delay analysis in the industrial tool suite: Issues, experiences and a case study," *Computer Science and Information Systems*, vol. 10, no. 1, 2013.

[10] N. Feiertag, K. Richter, J. Nordlander, and J. Jonsson, "A Compositional Framework for End-to-End Path Delay Calculation of Automotive Systems under Different Path Semantics," in *International Workshop on Compositional Theory and Technology for Real-Time Embedded Systems (CRTS)*, 2008.

[11] M. Becker, D. Dasari, S. Mubeen, M. Behnam, and T. Nolte, "Synthesizing job-level dependencies for automotive multi-rate effect chains," in *22nd IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2016, pp. 159–169.

[12] M. Becker, S. Mubeen, D. Dasari, M. Behnam, and T. Nolte, "A generic framework facilitating early analysis of data propagation delays in multi-rate systems (invited paper)," in *23th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2017, pp. 1–11.

[13] M. Becker, D. Dasari, S. Mubeen, M. Behnam, and T. Nolte, "End-to-end timing analysis of cause-effect chains in automotive embedded systems," *Journal of Systems Architecture*, vol. 80, pp. 104 – 113, 2017.

[14] M. Becker, S. Mubeen, M. Behnam, and T. Nolte, "Extending automotive legacy systems with existing end-to-end timing constraints," in *Information Technology - New Generations (ITNG)*. Springer International Publishing, 2018, pp. 597–605.

[15] S. Mubeen, M. Sjödin, T. Nolte, J. Lundbäck, M. Gålnander, and K. L. Lundbäck, "End-to-end timing analysis of black-box models in legacy vehicular distributed embedded systems," in *21st IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2015, pp. 149–158.

[16] J. Schlatow, M. Möstl, S. Tabuschat, T. Ishigooka, and R. Ernst, "Data-age analysis and optimisation for cause-effect chains in automotive control systems," in *13th International Symposium on Industrial Embedded Systems (SIES)*, 2018.

[17] A. Bucaioni, A. Cicchetti, F. Ciccozzi, R. Eramo, S. Mubeen, and M. Sjödin, "Anticipating implementation-level timing analysis for driving design-level decisions in east-adl," in *International Workshop on Modelling in Automotive Software Engineering*, 2015.

[18] A. Bucaioni, A. Cicchetti, F. Ciccozzi, S. Mubeen, A. Pierantonio, and M. Sjödin, "Towards design-space exploration of component chains in vehicle software," in *42nd Euromicro Conference series on Software Engineering and Advanced Applications, (WiP)*, Sep. 2016.

[19] L. Sha, T. Abdelzaher, K.-E. Årzén, A. Cervin, T. P. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. P. Lehoczky, and A. K. Mok, "Real Time Scheduling Theory: A Historical Perspective," *Real-Time Systems*, vol. 28, no. 2/3, pp. 101–155, 2004.

[20] S. Kramer, D. Ziegenbein, and A. Hamann, "Real World Automotive Benchmarks for Free," in *6th Int. Workshop on Analysis Tools and Methodologies for Embedded and Real-Time Systems (WATERS)*, 2015.

[21] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst, "System level performance analysis - the symta/s approach," *Computers and Digital Techniques*, vol. 152, no. 2, pp. 148–166, March 2005.

[22] M. Joseph and P. Pandya, "Finding response times in a real-time system," *Computer Journal*, vol. 29, no. 5, pp. 390–395, 1986.

[23] J. Mäki-Turja and M. Nolin, "Fast and tight response-times for tasks with offsets," in *17th Euromicro Conference on Real-Time Systems (ECRTS)*, 2005, pp. 127–136.