Embedded System Modeling and Verification Based on Deterministic and Stochastic Petri Net *

Gang HOU, Junwang CHANG, Kuanjiu ZHOU*, Mingchu LI

School of Software Technology, Dalian University of Technology, Dalian 116024, China

Abstract

Embedded systems are interrupt-driven systems, which achieve interactions with peripherals and environment through the interrupt mechanism, and handle exceptions. However, due to the randomness of trigger method, response with priority, and preemptive execution, interrupt behaviors are hard to accurately predict and interrupt defects are difficult to track. Once a program error is caused by interrupt, it will lead to crashes of the entire embedded system. In this paper, a modeling method of embedded system is proposed based on deterministic and stochastic Petri net (DSPN), which can simulate interrupt processing through three types transitions of DSPN. The upper limit processing time calculation methods of interrupt service are provided for the determinate timed transition. In addition, a model verification method based on the continuous stochastic logic (CSL) for DSPN model is given to analyze the influences of interrupt nesting for embedded system performance, and functional verification and performance evaluation of interrupt-driven embedded systems in one model are also implemented.

Keywords: Interrupt Nesting; Embedded System; Deterministic and Stochastic Petri Net; Performance Evaluation; Continuous Stochastic Logic

1 Introduction

Interrupt plays an indispensable role in the process of embedded system development, and the use of interrupt makes emergency handling, real-time processing and synchronous processing become more efficient in embedded systems. Since the triggered method of interrupt is random and uncertain, which makes the interrupt errors are difficult to be checked out through dynamic testing or static code analysis, and brings huge difficulty to the interrupt tests.

Petri net is a relatively complete formal modeling method, which has a precise formal definition, strictly regulate derivation method and better tool support, and especially suitable to describe the system’s control flow, concurrency and asynchronous behavior. In recent years, academia proposed many methods to apply Petri nets to embedded system modeling and verification. [1] provided a modeling and analysis integration approach for real-time system based on...

*Project supported by the National Nature Science Foundation of China (No. 61272174).
*Corresponding author.
Email address: zhoukj@dlut.edu.cn (Kuanjiu ZHOU).

This paper presents an embedded system interrupt model verification and analysis method based on DSPN. Firstly, analyze the main problems of interruption behavior modeling. Secondly, give the formal definition of DSPN, fire rules and calculation method of the interrupt processing time constraint; Thirdly, design a DSPN model checking method based on CSL for interrupt system; Finally, analysis the influence of interrupt for embedded system through the experiments, and verifies the practicability of this method.

2 Interrupt DSPN Model

2.1 Interrupt behavior analysis

An interrupt is a mechanism for pausing execution of whatever a processor is currently doing and executing a pre-defined code sequence called an interrupt service routine (ISR) or interrupt handler. A complete interrupt handling process should include: interrupt request, interrupt arbitration, interrupt response, interrupt service and interrupt return. In the process of interrupt modeling, it is necessary to consider the following aspects:

1) **Interrupt request randomness**: Interrupt request is triggered by the peripheral or artificially control, so the interrupt request is random. In the use of Petri nets for modeling, it is necessary to consider the expression of this random behavior.

2) **Respond according to the priority**: Different types of interrupts have different priorities, when multiple interrupt sources issue an interrupt request at the same time, CPU responds preferentially to the high-priority interrupt. This behavior can be resolved by introducing transition priority or inhibitor arcs.

3) **Interrupt nesting (preemptive execution)**: When the low-priority interrupt is executed, if there is a high priority interrupt request occurs, CPU will transfer to respond to high-priority interrupt. After it is processed, CPU will return to process low-priority interrupt, which requires modeling methods to support the preemption behavior expression.

4) **Strict execution time constraints**: Embedded system is real-time system, whose system correctness not only depends on the correctness of the results, but also depends on the time of this outcome. Therefore, the execution time needs to be considered when modeling the embedded systems, especially considering the maximum execution time of an operation. However, due to interrupt nesting behavior, the interrupt execution time has uncertainty factors:

1) The highest priority interrupt execution time is determined, because the highest priority interrupt handling process is not disturbed by other interrupts and the length of interrupt handlers is fixed when software design is completed.

2) The lower limit of interrupt execution time for the remaining priority interrupts is deter-
mined, but the upper limit is uncertain. Since these interrupt processing may be interrupted by a high priority interrupt, the upper limit is related to the high-priority interrupt trigger frequency, execution time and its own lower limit execution time. Therefore it is uncertain.

2.2 Related petri nets concepts

Definition 1 (Generalized stochastic Petri nets) \[6\] A GSPN = (S, T; F, W, M_0) where:

1. (S, T; F) is a net, S represents position, T represents transitions; F represents the flow relation between S and T; Inhibitor arcs are allowed in F, and only exist in the arc from the position to transition. The original enabling condition which is connected by the inhibitor arcs becomes disenable, the original disenable condition which is connected by the inhibitor arcs becomes enabling, and when the connected transition is firing, no mark is removed from the connected position;

2. W : F → \mathbb{N}^+ is the arc weight function;

3. M_0 : S → \mathbb{N} is the initial identification (marking), satisfy: ∀s ∈ S : M_0(s) ≤ K(s).

4. \( \lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_m\} \) is the average transition rate set; \( \lambda_i \) is the average transition firing rate of \( t_i \in T \), representing the average firing times per unit time in the case that it can be enabling.

5. Transition set T is divided into two subsets: \( T = T_t \cup T_i, T_t \cap T_i = \phi \), timed transition set \( T_t = \{t_1, t_2, \ldots, t_k\} \) and immediate transition set \( T_i = \{t_{k+1}, \ldots, t_n\} \), the average transition firing rate set associated with timed transition set is \( \lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_k\} \);

Definition 2 (Deterministic and stochastic Petri net) \[6\] A DSPN = (S, T; F, W, M_0), where:

1. Definition of S, W, M_0, \( \lambda \) is same as GSPN;

2. Transition set adds the transition T_d which has fixed time based on GSPN, T_d can be divided into two types:
   1) Independent transitions: Execution cannot be interrupted by other transitions;
   2) Preemptive transitions: Execution can be interrupted by other transitions.

2.3 Interrupt behavior modeling

When modeling interrupt behavior using DSPN, all aspects of interrupt handling characteristics need to be given full consideration. Interrupt processing is divided into four aspects: interrupt request, interrupt response, interrupt service and interrupt return.

Fig. 1 shows the interrupt behavior model of two interrupt request sources, which Interrupt 2 has a higher priority than Interrupt 1. In Fig. 1, \{T1, T5\} are timed transitions, which mean interrupt request and will fire according to the rate \{\lambda_1, \lambda_5\}; \{T2, T6\} are immediate transitions, which mean interrupt response and will fire immediately when the places before them have tokens; T3 is a determinate timed transition, which means interrupt service and its lower
limit execution time $\alpha_3$ is determinate but the upper limit execution time $\beta_3$ is uncertain, because of the inhibitor arcs from $S_8$ pointing to $T_3$; $T_7$ is also a determinate timed transition, which means interrupt service and its lower limit execution time $\alpha_7$ is equal to upper limit execution time $\beta_7$, because Interrupt 2 has the highest priority and $T_7$ cannot be interrupted; $\{T_4, T_8\}$ are immediate transitions, which mean interrupt return and will fire immediately when the places before them have tokens.

![Interrupt behavior model with two interrupt priority](image)

Because the embedded systems focus more on the worst execution time in real-time inspection, so the determinate timed transition of interrupt DSPN model are all represented by upper limit execution time. The calculation method of interrupt upper limit execution time is given below.

**Theorem 1** Let embedded system has $n$-level interrupt priority, the lowest priority is 1, the highest priority is $n$, interrupt execution time of all levels interrupt source is $[\alpha_i, \beta_i]$, firing frequency is $\lambda_i$, $i \in [1, n]$, then the upper limit interrupt execution time of each level interrupt source is $\beta_i$:

$$
\begin{align*}
\beta_n &= \alpha_n, \\
\beta_{n-1} &= \frac{\alpha_{n-1}}{1 - \lambda_n \alpha_n}, \\
&\quad \cdots, \\
\beta_1 &= \frac{\alpha_1}{1 - \lambda_2 \alpha_2 - \lambda_3 \alpha_3 - \cdots - \lambda_n \alpha_n}.
\end{align*}
$$

**Proof** (1) As the highest priority interrupt cannot be interrupted by other interrupts, so the upper limit execution time of the $n$-level interrupt equals to the lower limit execution time, namely $\beta_n = \alpha_n$;

(2) For the $n$-1 level interrupt:

$$
\begin{align*}
\beta_{n-1} &= \alpha_{n-1} + n_n \alpha_n, \\
n_n &= \lambda_n \beta_{n-1}.
\end{align*}
\Rightarrow \beta_{n-1} = \frac{\alpha_{n-1}}{1 - \lambda_n \alpha_n}.
$$

(3) For the 1 level interrupt:
\[
\begin{aligned}
\beta_1 &= \alpha_1 + n_2\alpha_2 + n_3\alpha_3 + \ldots + n_n\alpha_n, \\
n_2 &= \lambda_2\beta_1, \\
n_3 &= \lambda_3\beta_1, \\
&\quad\ldots, \\
n_n &= \lambda_n\beta_1. \\
\Rightarrow \beta_1 &= \frac{\alpha_1}{1 - \lambda_2\alpha_2 - \lambda_3\alpha_3 - \ldots - \lambda_n\alpha_n}. 
\end{aligned}
\]  

(3)

Where, \(n_i\) represents the execution times of the \(i\) level during \(\beta_i\). The above formula derivation is based on two characteristics of the interruption DSPN model:

1) All interrupt sources making interrupt request can be completed during \(\beta_1\);

2) If multiple interrupt requests occur at same time, interrupts are executed according to the priority, low-level interrupt requests will not be discarded. Therefore, the worst execution time of 1 level interrupt \(\beta_1\) should be \(\alpha_1\) plus execution time of all other level interrupts during \(\beta_1\).

3 DSPN Model Verification Based on CSL

For the DSPN model solution, usually construct an embedded Markov chain (EMC) isomorphic with DSPN reachability graph, calculate transition probability matrix of the EMC model, get steady-state probability of each state in the reachability graph, and thus solve the relevant statistical indicators. But for the verification of interrupt, model function indicators and performance indicators should be taken into account. For example: 1) Function indicators: Whether low-level interrupt can get a response, that is, whether an interrupt request can be smoothly executed until the interrupt return; 2) Performance indicators: Whether the whole interrupt handle process can be completed within a certain time, or what is the probability to complete within a certain time. These properties cannot be solved through traditional DSPN analysis methods, so we use a DSPN model verification method based on CSL. Related properties of DSPN model can be highly expressed by CSL, and complete solution and verification of the DSPN interrupt model.

3.1 DSPN reachability graph and steady-state probability

Calculation methods of DSPN model are as follows:

(1) Construct an embedded Markov chain (EMC) isomorphic with DSPN reachability graph: Assume there is an EMC isomorphic with the DSPN reachability graph, get EMC\(^2\) after removal of disappearing state, and then construct the corresponding Markov chain.

(2) Remove the disappearing state from EMC, calculate the existential state transition probability matrix only in the compressed EMC, and perform the system performance analysis.

The calculating formulas of stability probability of each state are as follows:

\[
\begin{aligned}
\prod Q &= 0, \\
\sum \pi_i &= 1 \quad 1 \leq i \leq n.
\end{aligned}
\]  

(4)

where: \(\pi_i\) is the steady state probability of state \(M_i\), \(\prod\) is steady state probability vector of the system \((\pi_0, \pi_1, \pi_2, \ldots, \pi_n)\); \(Q\) is transition probability matrix of each state, in which the element
$q_{ij}(i \neq j)$ is the probability from the $M_i$ of transfer to $M_j$, and diagonal elements $q_{ii}$ equals to the negative of the sum of rate labeled in each arc outputted from the state $M_i$. If there are no connected arcs, then $q_{ij} = 0$. Steady-state probability and inter-state transition probability matrix of each state in the DSPN reachability graph can be obtained through the above methods.

### 3.2 CSL model checking

There are two types of formulae in CSL: state formulae (which are true or false in a specific state), and path formulae (which are true or false along a specific path). A state formula is given by the following syntax:

1. $a$ for $a \in A$.
2. If $f_1$ and $f_2$ are state formula, then so are $\neg f_1, f_1 \lor f_2$.
3. If $g$ is a path formula, then $Pr_{\geq c}(g)$ is a state formula, where $c$ is rational between 0 and 1 expressed as the ratio of two binary coded integers.

Path formulas are formulas of the form $f_1 \cup [a_1] f_2 \cup [a_2] \ldots \cup f_n$, where $f_1, f_2, \ldots, f_n$ are state formulas, and $a_1, b_1, \ldots, a_{n-1}, b_{n-1}$ are nonnegative rational expressed as the ratio of two binary coded integers. CSL is the set of state formulae that are generated by the above rules.

Let $f$ be a state formula and $g$ be a path formula. We now define the satisfaction relation ($|=M$) using induction on the length of the formula. For a state formula $f$ we use $[[f]]_M$ to denote the set of states satisfying $f$.

1. $f$ is of the form $a : s |=_M f$ iff $\theta(s) = a$.
2. $f$ is of the form $(\neg f_1) : s |=_M f$ iff $s \not|=_M f_1$.
3. $f$ is of the form $(f_1 \lor f_2) : s |=_M f$ iff $s |=_M f_1$ or $s |=_M f_2$.
4. $f$ is of the form $Pr_{\geq c}(g) : s |=_M f$ iff $\mu^s([\pi \in U^s | \pi |=_M g]) > c$.
5. $g$ is a path formula of the form $f_1 \cup [a_1] f_2 \cup [a_2] \ldots \cup f_n : |=_M g$ iff there exist real numbers $t_1, \ldots, t_n$ such that for each integer in $[1, n]$ we have $(a_i \leq t_i \leq b_i) \land (\forall t' \in [t_{i-1}, t_{i}))(\pi(t) \in [[f]]_M)$, where $t_{n-1}$ is defined to be 0 for notational convenience. The CSL model checking problem is as follows: given a continuous time Markov chain $M$, a state $s$ in the chain, and a CSL formula $f$, is it the case that $s |=_M f$?

CSL model verification can be completed through the model verification tools PRISM [7], which is developed by the University of Birmingham, and is the more successful probability model verification tools. Considering different system characteristics, PRISM can support the probability properties of CSL and probability computation tree logic (PCTL). In terms of interrupt DSPN model verification, two main aspects need to be considered:

1. Probability values of established path formula: $\phi_1 = P_t(X_{[t_1,t_2]} \psi)$ or $\phi_2 = P_t(\psi_1 U_{[a_1,b_1]} \psi_2 U_{[a_2,b_2]} \ldots \psi_n)$.
2. Constraints specified by satisfied probability comparison operators of established path formula: $\phi_1 = P_{relop,c}(X_{[t_1,t_2]} \psi), \phi_2 = P_{relop,c}(\psi_1 U_{[a_1,b_1]} \psi_2 U_{[a_2,b_2]} \ldots \psi_n)$. Where $relop: \geq, >, \leq, <; c \in [0, 1]$. 

$q_{ij}(i \neq j)$ is the probability from the $M_i$ of transfer to $M_j$, and diagonal elements $q_{ii}$ equals to the negative of the sum of rate labeled in each arc outputted from the state $M_i$. If there are no connected arcs, then $q_{ij} = 0$. Steady-state probability and inter-state transition probability matrix of each state in the DSPN reachability graph can be obtained through the above methods.
4 Experiment

We take interrupt DSPN model shown in Fig. 1 as an example, analyzing the influences of multiple interrupts for embedded system performance.

(1) Construct DSPN simplified reachability graph

Fig. 2 shows the simplified reachability graph of interrupt DSPN model shown in Fig. 1. $M_0$ is the initial state or the state after the interrupt return; $M_3$ represents that the CPU will provide interrupt service for Interrupt 2, and interrupt 1 has no interrupt request; $M_4$ represents that the CPU will provide interrupt service for Interrupt 1, and interrupt 2 has no interrupt request; $M_8$ indicates that interrupt 1 and interrupt 2 are waiting for the CPU interrupt service at the same time, but interrupt 2 has a higher priority than interrupt 1, so CPU will execute the interrupt 2 service in the next phase.

![Fig. 2: Simplified reachability graph of interrupt DSPN model](image)

Interrupt DSPN model can be solved through TimeNET4.0 to obtain stability probability and state transition matrix. TimeNET4.0 is developed by Professor Zimmermann A.’s research team of the Technical University of Berlin, and it is an effective tool supporting DSPN modeling and analysis [8].

Table 1 shows the stability probability statistics of existential states with different firing rates and interrupt service time. We can see that the firing rate of high-level interrupt and the actual execution time of low-level interrupt will affect the reachability and real-time indicators of low-level interrupt.

![Table 1: State stability probability of DSPN model](image)

(2) Property expression and verification of CSL

Table 2 shows the property expression and verification of CSL for above DSPN model properties, and analyzes four properties of interrupt DSPN.

Above methods can analyze the success probability of interrupt system state transition when satisfying the time constraints and the performance influences of interrupt nesting depth for the interrupt system reachability and time constraints can be further analyzed by this means.
Table 2: Property expression and verification of CSL

<table>
<thead>
<tr>
<th>Property</th>
<th>Probability measure/Logical value</th>
<th>Illustrate</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_r(X_{[0.1,0.3]} \neg M_0 \lor \neg M_3)$</td>
<td>0.1146</td>
<td>the probability of state is neither in $M_0$ nor in $M_3$ at $[0.1,0.3]$</td>
</tr>
<tr>
<td>$P_r(M_8 U_{[0.1,0.2]} M_0)$</td>
<td>0.9284</td>
<td>the probability of state is in $M_8$ until it transfers to $M_0$ at $[0.1,0.2]$</td>
</tr>
<tr>
<td>$P_{&gt;0.1}(X_{[0.1,0.3]} \neg M_0 \lor \neg M_3 \lor M_8)$</td>
<td>F</td>
<td>Whether it true or not that the probability is large than 0.1 which the state is in $M_8$ not in $M_0$ and $M_3$ at $[0.1,0.3]$</td>
</tr>
<tr>
<td>$P_{&gt;0.2}(M_0 U_{[0.1,0.2]} M_3)$</td>
<td>T</td>
<td>Whether it true or not that the probability is large than 0.2 which state is in $M_0$ until it transfers to $M_3$ at $[0.1,0.2]$</td>
</tr>
</tbody>
</table>

5 Conclusions

In this paper, a modeling method of embedded system is proposed based on deterministic and stochastic Petri net (DSPN), which simulates interrupt processing through three type transitions of DSPN. The upper limit processing time calculation methods of interrupt service are provided for the determinate timed transition. In addition, a model verification method based on the continuous stochastic logic (CSL) for DSPN model is given to realize functional verification and performance evaluation of interrupt-driven embedded systems in one model.

References