A Proof for Memory Access Patterns for the Analysis of MPSoCs

Simon Schliecker, Matthias Ivers, Rolf Ernst
Institute for Computer and Communication Network Engineering
Technical University of Braunschweig
Email: [schliecker, ivers, ernst]@ida.ing.tu-bs.de

Abstract

This technical report contains the proof to the algorithm presented in [1].

I. Proof

Theorem 1: Let $\mathbb{I}$ be the set of all i invocations of tasks that may arrive within a time window of size t. Furthermore, let $\delta_\tau(m)$ be the minimum amount of time a task invocation τ needs to execute in order to produce a total of m requests, with $\delta_\tau(2) > 0$. Then the minimum amount of time $\delta_{req}(n)$ to send n requests from the processor can be calculated as follows :

```
| // Consider immediate requests:
| $\forall_{\tau\in\mathbb{I}} : t_\tau = 0, m_\tau = 1$
| $\delta_{req}(1) = .. = \delta_{req}(i) = 0$, $m = i$
| // Stepwise construction of minimum request distances:
| until $\delta_{req}(m) > t$ OR $\forall s \in S : m_s \geq q_s^{max}$
| | // Smallest necessary additional execution
| | $\forall_{\upsilon\in\mathbb{I}} : d_\upsilon = \delta_\upsilon(m_\upsilon + 1) - t_\upsilon$
| | $\tau = \{\tau \in \mathbb{I}\,|\min_{\upsilon\in\mathbb{I}}\{d_\upsilon\} = d_\tau\}$
| | // update requests that are considered
| | $\delta_{req}(m) = \delta_{req}(m-1) + d_\tau$
| | $m_\tau = m_\tau + 1$, $t_\tau = t_\tau + d_\tau$
| | $m = m + 1$
| repeat
| return $\delta_{req}$
```

Proof: (Induction Start.) Initially, during the smallest possible time, it is conservative to assume that no more requests can be sent than the sum over the requests that may be sent instantaneously by any task invocation in $\mathbb{I}$. This is the result of one task sending its requests, and immediately being removed from the running tasks and replaced by another that does the same. This can go on until all tasks have sent their first request. Thus the request times of the first i requests are zero.

(Induction Step.) Let $\delta_{req}(m)$ be the size of the smallest time interval in which m requests can be sent, but $m+1$ can not. Furthermore let t_1 to t_s be the amount of times that the the algorithm above has chosen tasks 1 to $s \in \mathbb{I}$ which produced the minimum distances between the first m requests. The amount of requests produced by each task was m_1, $...m_s$. Then the next request by any invocation τ can only be sent when it executes for at least $\delta_\tau(m_i + 1) - t_i$. The request $m+1$ from the processor can not occur earlier than if the task with the smallest necessary computation time were executing exclusively. If at time point t any invocation υ were chosen instead of τ, the following request could only be sent later, as υ would require more computation time than τ to send the $m_\upsilon + 1$-th request. Also no successive request could occur earlier, as execution of no other task but υ would progress so that their required execution time until the next request would remain the same. Thus the minimum time to produce m request is given by:

$$\delta_{req}(m+1) = \delta_{req}(m) + \min_{\tau=1..s}\{\delta(m_\tau + 1) - t_\tau\}.$$

Additionally, to proceed with the proof of induction, $t + d$ is the size of the smallest time interval in which $m+1$ requests can be sent. ■

References

[1] S. Schliecker, M. Ivers, and R. Ernst, "Memory access patterns for the analysis of mpsocs," in *NewCAS conference*. Gatineau, Canada: IEEE, June 2006.