From Quasi-Synchrony to LTTA

Guillaume Baudart Timothy Bourke Marc Pouzet

21 November 2013

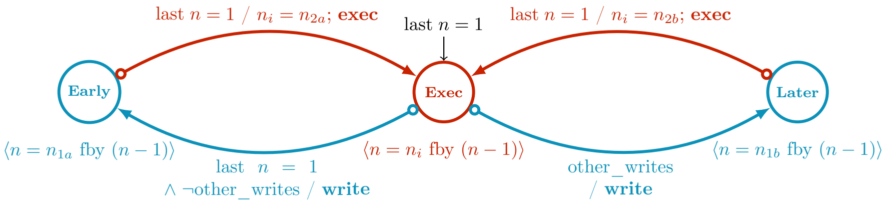
A Quasi-periodic System is one where every process P is periodic with a nominal period T_P^n and a jitter of ϵ. The time between two ticks may thus vary between 'small margins' during an execution:

$$T_P^n - \epsilon \leq \kappa_i - \kappa_{i-1} \leq T_P^n + \epsilon.$$

Signal values are sent across a bus to one-place buffers at a receiver, whence they are sampled periodically.

In his 'cooking book' [2], Paul Caspi showed how to build abstractions for implementing discrete systems on top of this architecture. These discrete abstractions can be expressed in a synchronous language and used to simulate quasi-synchronous systems [4]. In later work, with Albert Benveniste and others [1, 3, 5], he proposed communication protocols for preserving the discrete semantics of signal flows.

We present a brief survey of this work. In particular, we explain the simple relations between the periods and jitters of real-time tasks, and overwriting and oversamplings of values between writers and readers (it's all a matter of fence posts). We generalize (slightly) the idea of quasi-synchronous traces. We also clarify one of the communication protocols by modelling it in the hybrid synchronous language Zélus (see Figure 1 and the corresponding code below).


last $n = 1$ / $n_i = n_{2a}$; **exec**
last $n = 1$
last $n = 1$ / $n_i = n_{2b}$; **exec**
Early
Exec
Later
$\langle n = n_{1a}$ fby $(n-1)\rangle$
$\langle n = n_i$ fby $(n-1)\rangle$
$\langle n = n_{1b}$ fby $(n-1)\rangle$
last $n = 1$
$\wedge$ ¬other_writes / **write**
other_writes
/ **write**

```
let node controller (sow) = (write, exec, n) where
  rec init n = 1
  and automaton
      | Exec(ni) ->
          do n = ni fby (n - 1)
          unless sow then do emit write = () in Later
          else (last n = 1) then do emit write = () in First
      | First ->
          do n = n1a fby (n - 1)
          unless (last n = 1) then do emit exec = () in Exec(n2a)
      | Later ->
          do n = n1b fby (n - 1)
          unless (last n = 1) then do emit exec = () in Exec(n2b)
  init Exec(0)
```

Figure 1: Time-Based LTTA Protocol

References

[1] Benveniste, A., Bouillard, A., and Caspi, P. A unifying view of loosely time-triggered architectures. In *Proceedings of the 10th ACM International Conference on Embedded Software (EMSOFT'10)* (Scottsdale, Arizona USA, Oct. 2010), ACM Press, pp. 189–198.

[2] Caspi, P. The quasi-synchronous approach to distributed control systems. Tech. Rep. CMA/009931, VERIMAG, Crysis Project, May 2000.

[3] Caspi, P., and Benveniste, A. Time-robust discrete control over networked loosely time-triggered architectures. In *Proceedings of the 47th IEEE Conference on Decision and Control* (Cancun, Mexico, Dec. 2008), IEEE, pp. 3595–3600.

[4] Halbwachs, N., and Mandel, L. Simulation and verification of aysnchronous systems by means of a synchronous model. In *Proceedings of the 6th International Conference on Application of Concurrency to System Design (ACSD 2006)* (Turku, Finland, June 2006), IEEE Computer Society, IEEE, pp. 3–14.

[5] Tripakis, S., Pinello, C., Benveniste, A., Sangiovanni-Vincent, A., Caspi, P., and Di Natale, M. Implementing synchronous models on loosely time triggered architectures. *IEEE Transactions on Computers 57*, 10 (2008), 1300–1314.