The Importance of Timing

- In “hard real-time” systems, meeting timing requirements is just as important as getting the logic of the functionality correct
- In “soft real-time” systems, meeting timing requirements is important, but it is usually treated as a probabilistic requirement

- The pacemaker is definitely a hard real-time system!

Functional Timing Reqs - 1

- We need to examine functional timing requirements in more detail.
- As an example, consider a sensor trip that depends on a sensor (signal) being in a trip state for a sustained period of time.

Functional Timing Reqs - 2

$f\_HTLFsentrip_i$, i=1,..,4

{For each i = 1,..,4}

Condition		Result $f\_HTLFsentrip_i$
$t_{now} \geq$ k_HTLFdelay	($f\_HTLFimmsentrip_i$ = e_Trip) Held for (k_HTLFdelay) *{Immediate sensor trip active for at least past k_HTLFdelay time period}*	e_Trip
	NOT [($f\_HTLFimmsentrip_i$ = e_Trip) Held for (k_HTLFdelay)] *{Immediate sensor trip not active for at least one instant in past k_HTLFdelay time period}*	e_NotTrip
$t_{now} <$ k_HTLFdelay	($f\_HTLFimmsentrip_i$ = e_Trip) Held for (t_{now}) *{Immediate sensor trip active since initialization}*	e_Trip
	NOT [($f\_HTLFimmsentrip_i$ = e_Trip) Held for (t_{now})] *{Immediate sensor trip not active for at least one instant since initialization}*	e_NotTrip

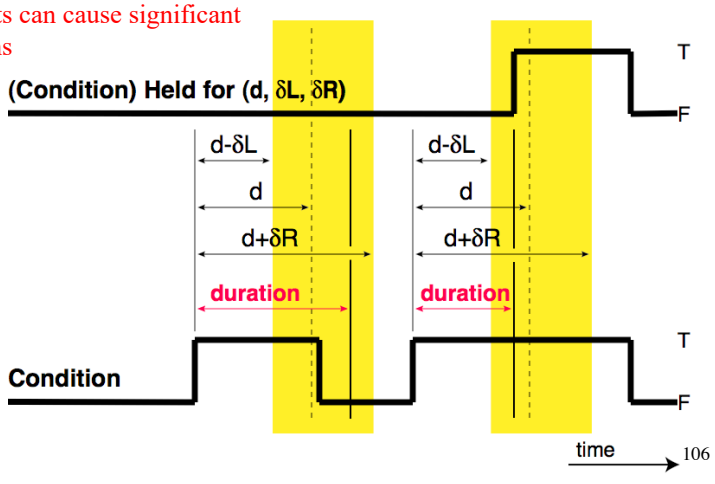
So - what is this "Held for" thing?

105

105

Functional Timing Reqs - 3

Allowing tolerances on timing constants can cause significant problems



106

Definition of Held for

(Condition: bool) Held for (d: $\mathbb{R}^{>0}$, δL, δR: $\mathbb{R}^{\geq 0}$): bool
duration(Condition: bool): [d-δL, d+δR]
Event_start_time(Condition: bool): $\mathbb{R}^{\geq 0}$
Initially:
duration = any value in [d-δL, d+δR]; Event_start_time$_{-1}$ = 0; Condition$_{-1}$ = False

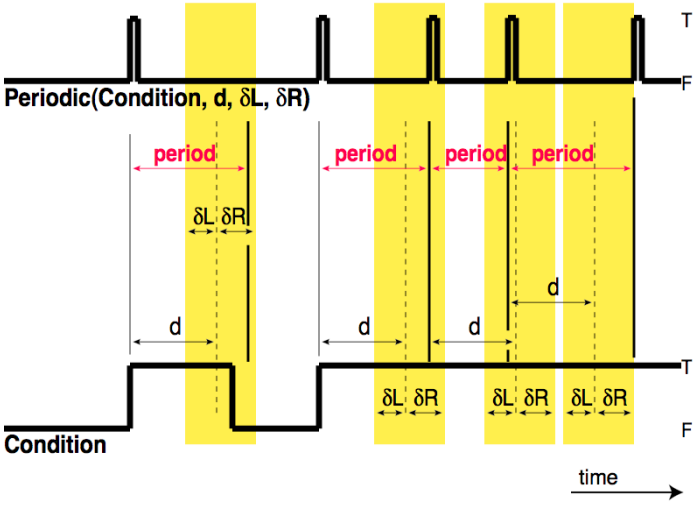
	duration	Event_start_time
(Condition = True) & (Condition$_{-1}$ = False)	Any value in [d-δL, d+δR]	t_{now}
(Condition = False) or (Condition$_{-1}$ = True)	No Change	No Change

		Held for
Condition = True	t_{now} – Event_start_time ≥ duration	True
	t_{now} – Event_start_time < duration	False
Condition = False		False

107

107

Another General FTR

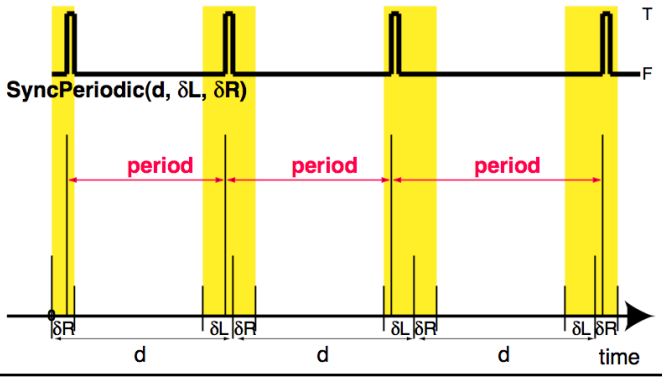


108

108

And Another General FTR

This one is synchronized with an external clock.



109

109

Functional Timing Requirements

- Functional timing requirements are timing requirements that are directly related to the required behaviour of the application
  - So, “Held for”, “Periodic” and “SyncPeriodic” are examples of templates describing functional timing requirements
  - Other common functional timing requirements can be modeled in similar fashion
  - These requirements are interpreted within the constraints of the governing model, in our case the discrete time FSM with arbitrarily small clock-tick. This describes *idealized behaviour* with the capability of including tolerances on all timing durations

110

110

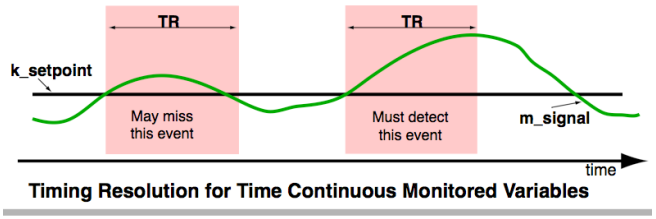
Performance Timing Requirements

- Idealized behaviour totally ignores the fact that an implementation cannot continuously monitor sensor values and requires a finite, non-zero amount of time to process its results
- To complete the description of the required behaviour, a requirements document must also specify the performance tolerances that are allowed in meeting functional timing requirements
- We identify two different performance timing requirements
  - Timing Resolution
  - Response Allowance

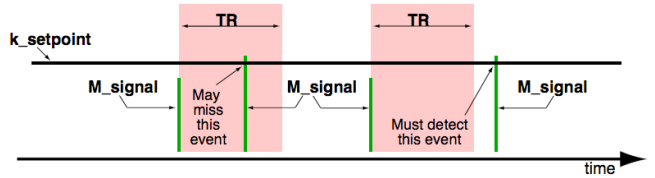
111

111

Timing Resolution



Timing Resolution for Time Continuous Monitored Variables



Timing Resolution for Time Discrete Monitored Variables

112

112

Response Allowance

- Response Allowance
  - The Response Allowance (RA) for a controlled-monitored variable pair specifies an allowable processing delay
  - Each controlled variable must have an RA specified for it. The RA applies to the controlled variable and the particular monitored variable on which the controlled variable's behaviour depends
  - The RA is measured from the time the event *actually occurred in the physical domain*, until the time the value of the controlled variable is generated and *crosses the application boundary into the physical domain*

113