Agenzia nazionale per le nuove tecnologie, l'energia
e lo sviluppo economico sostenibile

Ministero dello Sviluppo Economico

RICERCA DI SISTEMA ELETTRICO

Studio e analisi di sistemi di controllo implementabili con logiche programmabili per il miglioramento delle prestazioni e della sicurezza di impianti nucleari di nuova concezione

S. Di Gennaro, B.Castillo–Toledo, F. Memmi

RdS/2012/160

STUDIO E ANALISI DI SISTEMI DI CONTROLLO IMPLEMENTABILI CON LOGICHE PROGRAMMABILI PER IL MIGLIORAMENTO DELLE PRESTAZIONI E DELLA SICUREZZA DI IMPIANTI NUCLEARI DI NUOVA CONCEZIONE

Stefano Di Gennaro, Bernardino,Castillo–Toledo, Fabrizio Memmi Università dell'Aquila

Settembre 2012

Report Ricerca di Sistema Elettrico

Accordo di Programma Ministero dello Sviluppo Economico - ENEA

Area: Governo, Gestione e Sviluppo**,** del Sistema Elettrico Nazionale

Progetto: Nuovo Nucleare da Fissione: Collaborazioni Internazionali e sviluppo Competenze in Materia Nucleare

Responsabile del Progetto: Massimo Sepielli,ENEA

Center of Excellence DEWS

Department of Electrical and Information Engineering

University of L'Aquila, V. G. Gronchi 18, 67100, L'Aquila, Italy

Deliverable 1

Study and analysis of control systems implementable by programmable logic for performance and safety improvements of novel nuclear plants

---

Studio ed analisi di sistemi di controllo implementabili con logiche programmabili per il miglioramento della prestazioni e della sicurezza di impianti nucleari di nuova concezione

---

Authors:
Stefano Di Gennaro, Bernardino Castillo–Toledo, Fabrizio Memmi

Principal Investigator:
Prof. Stefano Di Gennaro

Project PAR 2011

June 30, 2012

Abstract

In this deliverable, the aspects concerning the digital implementation of a control law on a physical device are analyzed, from the viewpoint of the study, design and realization of supervisory, control and protection systems for improving the performance and safety of novel nuclear plants. In fact, it is well known that the implementation of control laws with zero order holders, commonly used, introduce a delay and hence could bring to unstable behaviors. A technique called self–triggered control allows determining the sampling times necessary to implement the controller, preserving the desired performance. In the digital logic system scenario, there are many architectures suitable for realizing different kinds of control algorithms, considering performances in terms of timing, power consumption and resources availability. Different implementation solutions are hence studied, from microprocessors to custom devices for digital processing, in order to obtain an evaluation about benefits and drawbacks. In particular, FPGA technology will be studied in detail, explaining how designers can exploit their potentialities in applications of parallel computation, control and digital signal processing.

Riassunto

In questo documento vengono analizzati gli aspetti relativi all'implementazione digitale di una legge di controllo su un dispositivo fisico, dal punto di vista dello studio, progetto e realizzazione di sistemi di supervisione, controllo e protezione per aumentare la prestazione e sicurezza di impianti nucleari di nuova concezione. Infatti è ben noto che l'implementazione di leggi di controllo mediante organi di tenuta di ordine zero, comunemente utilizzati, introducono un ritardo e quindi possono portare a comportamenti instabili. Una tecnica chiamata controllo auto–innescante permette di determinare i tempi di campionamento successivi per implementare il controllore, preservando le prestazioni desiderate. Nello scenario dei sistemi a logica digitale, vi sono molte architetture adatte a realizzare diversi tipi di algoritmi di controllo, considerando prestazioni in termini di tempi, consumo di potenza e disponibilità di risorse. Varie soluzioni implementative sono quindi studiate, dai microprocessori a dispositivi personalizzabili, per ottenere una valutazione dei benefici e dei svantaggi. In particolare la tecnologia FPGA sarà studiata in dettaglio, spiegando come i progettisti possono sfruttare i loro potenziali in applicazioni di calcolo parallelo, controllo e processamento del segnale digitale.

1 The Nonlinear Continuous Time Controllers Implemented by Digital Devices

In [1], [2], the supervision, control and protection systems for nuclear reactors of new generation has been analyzed. Using a mathematical model of the primary circuit, simple enough for the control purposes but accurate enough to capture the nonlinear, time–varying, switching nature of the plant, a dynamic level controller is determined for the pressurizer water level. Moreover, two dynamics controllers have been designed for the pressurizer pressure. These controllers may not use measurements of the pressurizer pressure, relying only on the pressurizer wall temperature measurements.

The aforementioned controllers are continuous time, namely the assume that the state variables, necessary to implement the feedback, are continuously measured, while the control is computed at each time instant and applied to the controlled system. In reality, such controllers are implemented by digital devices, much more flexible than analog devices. Other notable characteristics are the multitasking capability of the digital controllers, and the possibility of parallel data acquisition and computation. This last property allows faster computation times, so approaching the performance of the digitally implemented controller to that obtainable with an analogically implemented controller.

1.1 Digital Implementation of Control Strategies: Some Important Issues to Take into Account

A very popular way of determining and implementing a controller on a digital device is to design the controller assuming the continuous time behavior of the system, and then implement the continuous controller by means of zero order holders, commonly used for digital implementations. Such controller are usually named “emulated” controllers. It easy to understand that this design technique could bring to unsatisfactory behaviors of the controlled system if the sampling time is high. In fact, it is well known that the effect of the presence of zero order holders is equivalent to a delay of approximatively the half of the sampling time. As well known also in the case of linear systems, a delay could bring to unstable behaviors.

The effects of the sampling and zero order holders can be seen also in a alternative way. Let us

suppose that the controller is determined considering the continuous time dynamics

$$\dot{x} = f(x, u)$$

where $x \in \mathcal{D}_x$ represent the vector of the state variables, $u \in \mathcal{D}_u$ is the vector of the input variables, $\in \mathcal{D}_x \subset \mathbb{R}^n$, $\in \mathcal{D}_u \subset \mathbb{R}^p$ are the domains where x, u take values, and $f : \mathcal{D}_x \times \mathcal{D}_u \to \mathbb{R}^n$ is the function describing the system dynamics. Usually, this function is requested to fulfill suitable conditions to ensure existence and unicity of the solution for each initial state $x(0) = x_0$, such as the (local) Lipschitz condition, namely

$$\|f(x_1, u) - f(x_2, u)\| \leq L\|x_1 - x_2\|$$

with $x_1, x_2 \in \mathcal{D}_x$ two generic state values, and L a constant (in the local version of this property, L depends on the region Ω considered) called Lipschitz constant. The value of L measures how much f varies when x_2 differs from x_1, and is a measure of the "nonlinearity" of f. Considering that L can be determined considering the maximum value the jacobian norm $\|\partial f/\partial x\|$ takes on the region Ω that is taken into account, it is clear that L measures the "derivative" of f.

Let us assume that a controller $u = \alpha(x)$ has been determined somehow. This is a continuous time control, and should be better denoted as $u_t = \alpha(x_t)$, where t is the continuous time. For the sake of simplicity, the dependence on t is usually dropped. This controller is normally implemented considering the sampled state value x_k at time $t = k\delta$, and the (constant) numerical value $u_k = \alpha(x_k)$ is applied to the system by means of zero holder devices, that hold the value u_k over the time interval $[k\delta, (k+1)\delta)$, where δ is the sampling time and $k \in \mathbb{Z}$ is an integer.

It is therefore clear that eventually the dynamics of the controlled system is given by

$$\dot{x} = f(x, \alpha(x_k))$$

which apparently differ from the ones

$$\dot{x} = f(x, \alpha(x))$$

obtained when the continuous time controller is applied. It is clear that the effect of the sampling is equivalent to a disturbance d acting on the system, since

$$\dot{x} = f(x, \alpha(x_k)) = f(x, \alpha(x)) + d$$

where

$$d = f(x, \alpha(x_k)) - f(x, \alpha(x)).$$

This disturbance is periodically zero at the sampling time $t = k\delta$, while grows when $t > k\delta$. The growth of this disturbance depends on the "nonlinearity" of f and, obviously, on the value of δ. In particular,

when δ is small the disturbance remains small enough to affect greatly the performance of the controller. But in the case in which δ could take (relatively) big values, d can take bigger values. The measure of how big this disturbance value can be is given, again, by the Lipschitz constant L, since

$$\|d\| = \|f(x, \alpha(x_k)) - f(x, \alpha(x))\| \leq L\|x - x_k\| \leq L \max_{x\in\Omega} \|x - x_k\| = d_{\max}.$$

The effect of this disturbance on the closed loop behavior is to take away x from the desired value. Let x us assume[1] that one desires that x goes asymptotically to $x = 0$, by means of the control $u = \alpha(x)$. This control can be designed[2] making use of a (positive definite) Lyapunov function V such that, when $d = 0$

$$\dot{V} = \frac{\partial V}{\partial x} f(x, \alpha(x))$$

is definite negative. If $\alpha_3(\|x\|)$ is a $\mathcal{K}$ function[3] this can be expressed saying that

$$\dot{V} = \frac{\partial V}{\partial x} f(x, \alpha(x)) \leq -\alpha_3(\|x\|).$$

This means that V, which can take the meaning of a (generalized) energy function, is decreasing as time passes. Therefore, asymptotically x tends to the origin. When d is nonzero, namely when the emulated controller is used, what can be ensured[4] is that $\dot{V}$ is negative definite outside a region

$$\dot{V} = \frac{\partial V}{\partial x} f(x, \alpha(x_k)) = \frac{\partial V}{\partial x}\Big[f(x, \alpha(x)) + d\Big] \leq -\alpha_3(\|x\|) + \alpha_4(\|x\|)d_{\max}$$

where

$$\left\|\frac{\partial V}{\partial x}\right\| \leq \alpha_4(\|x\|)$$

and $\alpha_4(\|x\|)$ is an appropriate $\mathcal{K}$ function[5]. To determine such a region, a ball of the origin of radius μ, one has to "spend" a part of the term $-\alpha_3(\|x\|)$ (ensuring the asymptotic convergence of x to the origin) to "counteract" the positive term $\alpha_4(\|x\|)d_{\max}$

$$\dot{V} \leq -(1 - \vartheta)\alpha_3(\|x\|) + \alpha_4(\|x\|)d_{\max} - \vartheta\alpha_3(\|x\|) \leq -(1 - \vartheta)\alpha_3(\|x\|)$$

for

$$\|x\| \geq \mu := \alpha_3^{-1}\Big(\frac{1}{\vartheta} \max_{x\in\Omega} \alpha_4(\|x\|)d_{\max}\Big)$$

where $\vartheta \in (0, 1)$. As a result of the fact that $\dot{V}$ is negative definite outside the ball of radius μ, the trajectories of the controlled system tend asymptotically to the ball of radius μ and remains in there. This

[1] It is possible to show that the origin can be considered as equilibrium point, after an appropriate change of coordinate.
[2] The controllers designed in [1] have been obtained precisely with a Lyapunov–based technique.
[3] A $\mathcal{K}$ function $\alpha_3(r)$ is a continuous function such that $\alpha_3(0) = 0$, and is strictly increasing, see [8].
[4] A "worst–case scenario is here considered.
[5] It is possible to show that such a function $\alpha_4(\|x\|)$ does exists.

property is usually called "practical stability" of the origin, since generalizes the property of asymptotic stability to the origin. It is also called "ultimate boundedness" of the trajectories, since the solution x_t of the differential equation will enter, at a certain time instant, the ball of radius μ and will remain in it. The choice of ϑ determines a trade–off between of the dimension of the region about the origin and the convergence velocity of the state trajectories to this ball.

Other aspects can be also be addressed[6]. Notably, the fact that the sampling of the state variables necessary to the control implies that the control law is applied with a time delay δ, so that the system dynamics are more precisely given by

$$\dot{x} = f(x, u_{k-1}).$$

The resolution of this problem, that can be addressed by means of predictors of the form

$$\dot{\xi} = f(\xi, \alpha(\xi_k))$$

$$u_k = \alpha(\xi_k)$$

goes far beyond the scope of the deliverable. In any case, also this effect can be seen as a disturbance acting on the system, since the system dynamics can be written as

$$\dot{x} = f(x, \alpha(x_{k-1})) = f(x, \alpha(x)) + \bar{d}$$

with

$$\bar{d} = d + f(x, \alpha(x_{k-1})) - f(x, \alpha(x_k))$$

and treated in a similar way.

From the previous discussion, it is clear that the effect due to the sampling can be seen as due, mainly, to a persistent perturbation acting on the system. It is also clear that the effect of this disturbance is smaller if smaller is the amplitude of the disturbance which, in turn, diminishes as the sampling time δ is smaller. The aim of the present deliverable is to study criteria ensuring better implementations of control laws and logics on digital programmable devices, with the goal of improving the performance and safety in nuclear plants of novel conception. In particular, more effective digital platform will be studied, which render the implementation of control law more flexible and effective.

1.2 A Mathematical Model of the Primary Circuit of a PWR

In [1] a mathematical model of the primary circuit of a PWR has been considered. The reader can find in [1] the details of this model, given by[7]

[6]Further effects, such as quantizations effects, are not considered here, since considered negligible.

[7]The subindices "r", "pc", "sg", "pr" refer to the reactor, primary circuit, steam generator, and pressurizer, respectively.

$$
\begin{aligned}
\dot{N} &= -\frac{p_0 + p_1 v + p_2 v^2}{\Lambda} N + S \\
\dot{M}_{pc} &= m_{in} - m_{out} \\
\dot{T}_{pc} &= \frac{1}{c_{p,pc} M_{pc}} \Big[c_{p,pc} m_{in} (T_{pc,i} - T_{pc}) + c_{p,pc} m_{out} \Delta + c_\psi N - n_{sg} k_{t,sg} (T_{pc} - T_{sg}) - W_{loss,pc} \Big] \\
\dot{T}_{sg} &= \frac{1}{c^l_{p,sg} M_{sg}} \Big[c^l_{p,sg} m_{sg} T_{sg,sw} - c^v_{p,sg} m_{sg} T_{sg} - m_{sg} E_{\text{evap},sg} + k_{t,sg} (T_{pc} - T_{sg}) - W_{loss,sg} \Big] \\
\dot{T}_{pr} &= \frac{1}{c_{p,pr} M_{pr}} \Big[- k_{wall} (T_{pr} - T_{pr,wall}) + W_{heat,pr} + \delta_{pr} \Big(c_{p,pc} m_{pr} (T_{pc} + \Delta) - c_{p,pr} m_{pr} T_{pr} \Big) \Big] \\
\dot{T}_{pr,wall} &= \frac{1}{c_{p,wall}} \Big[k_{wall} (T_{pr} - T_{pr,wall}) - W_{loss,pr} \Big]
\end{aligned}
\tag{1}
$$

where the state variables are the neutron flux N (in %), the overall mass in the primary circuit M_{pc} (in kg), the average temperature of the water in the primary circuit T_{pc} (in °C), the average secondary circuit liquid temperature T_{sg} (in °C), the pressurizer water/wall temperature T_{pr}, $T_{pr,wall}$ (in °C), and where

$$
\begin{aligned}
M_{pr} &= M_{pc} - \varphi(T_{pc}) V_{pc,0}, \qquad \varphi(T_{pc}) = c_{\varphi,0} + c_{\varphi,1} T_{pc} - c_{\varphi,2} T_{pc}^2 \\
m_{pr} &= m_{in} - m_{out} - \frac{1}{c_{p,pc} M_{pc}} \frac{\partial \varphi(T_{pc})}{\partial T_{pc}} V_{pc,0} \Big[c_{p,pc} m_{in} (T_{pc,i} - T_{pc}) + c_{p,pc} m_{out} \Delta \\
&\qquad + c_\psi N - n_{sg} k_{t,sg} (T_{pc} - T_{sg}) - W_{loss,pc} \Big].
\end{aligned}
$$

In (1) v, m_{in}, $W_{heat,pr}$ are the input variables, while $T_{pc,i}$, m_{out}, m_{sg}, M_{sg}, $T_{sg,sw}$ can be considered as disturbances. The model (1) is hybrid and nonlinear, since the equation of T_{pr} contains the switching term δ_{pr}, which is 1 if $m_{pr} > 0$ and 0 if $m_{pr} \leq 0$. Moreover, it is simple enough for the control purposes but accurate enough to capture the nonlinear, time–varying, switching nature of the plant.

The (measurable) outputs of the systems are the reactor power $W_r(N)$, the steam generator pressure p_{sg} (in kPa), the pressurizer pressure p_{pr} (in kPa), the pressurizer water level l_{pr} (in m)

$$
\begin{aligned}
W_r(N) &= c_\psi N \\
p_{sg} &= p_{*,T}(T_{sg}) = c_0 - c_1 T_{sg} + c_2 T_{sg}^2 \\
p_{pr} &= p_{*,T}(T_{pr}) = c_0 - c_1 T_{pr} + c_2 T_{pr}^2 \\
l_{pr}(M_{pc}, T_{pc}) &= \frac{1}{A_{pr}} \left(\frac{M_{pc}}{\varphi(T_{pc})} - V_{pc,0} \right).
\end{aligned}
\tag{2}
$$

The model parameters are reported in Table 1.

1.3 The Pressurizer Inventory and Pressure Controllers

The controller for pressurizer water level and pressure determined in [1] are dynamic controller relying only on the pressurizer wall temperature measurements. Considering a reference level $l_{pr,\text{ref}}$, usually

Reactor			
Neutron flux (state variable)	N	99.3	%
Control rod position (input)	v	0	cm
Reactor power (output)	W_r	13.654×10^8	W
Constant in the reactor power equation	c_ψ	13.75×10^6	W/%
Generation time	Λ	10^{-5}	s
Rod reactivity coefficients	p_0	2.85×10^{-4}	m
	p_1	6.08×10^{-5}	m^{-1}
	p_2	1.322×10^{-4}	m^{-2}
Flux of the constant neutron source	S	2830.5	%/s
Total fraction of delayed neutrons	β	0.0064	
Average half–life	λ	0.1	s^{-1}
Primary circuit			
Overall mass in the primary circuit (state)	M_{pc}	2×10^5	kg
Water average temperature (state)	T_{pc}	281.13	°C
Inlet mass flow rate (input)	m_{in}	1.4222	kg/s
Outlet mass flow rate (disturbance)	m_{out}	2.11	kg/s
Hot leg water temperature	$T_{pc,hl}$	296.13	°C
Cold leg water temperature	$T_{pc,cl}$	266.13	°C
Inlet temperature (disturbance)	$T_{pc,i}$	258.85	°C
Specific heat at 282°C	$c_{p,pc}$	5355	J/kg/K
Heat transfer coefficient	$k_{t,sg}$	9.5296×10^6	W/K
Heat loss	$W_{loss,pc}$	2.996×10^7	W
Water nominal volume	$V_{pc,0}$	242	m^3
Water nominal mass	$M_{pc,0}$	2×10^5	kg
Differences $T_{pc,hl}-T_{pc}=T_{pc}-T_{pc,cl}$	Δ	15	°C
Pressurizer			
Water temperature (state)	T_{pr}	326.57	°C
Heating power (input)	$W_{heat,pr}$	168	kW
Water level (output)	l_{pr}	4.8	m
Pressure (output)	p_{pr}	123×10^2	kPa
Water specific heat at 325°	$c_{p,pr}$	6873.1	J/kg/K
Heat capacity of the wall	$c_{p,wall}$	6.4516×10^7	J/°C
Wall heat transfer coefficient	k_{wall}	1.9267×10^8	W/°C
Heat loss	$W_{loss,pr}$	1.6823×10^5	W
Water mass	M_{pr}	19400	kg
Vessel cross section	A_{pr}	4.52	m^2
Vessel volume	$V_{pr,vessel}$	44	m^3
Steam generator			
Average secondary circuit liquid temperature (state)	T_{sg}	257.78	°C
Secondary circ. water specific heat at 260°	$c^l_{p,sg}$	3809.9	J/kg/K
Secondary circ. vapor specific heat at 260°	$c^v_{p,sg}$	3635.6	J/kg/K
Heat loss	$W_{loss,sg}$	1.8932×10^7	W
Evaporation energy at 260°	$E_{\text{evap},sg}$	1.658×10^6	J/kg
Water mass	M_{sg}	34920	kg
Water level	l_{sg}	1.850	m
Steam pressure (output)	p_{sg}	45.3×10^2	kPa
Secondary water mass flow rate (disturbance)	m_{sg}	119.31	kg/s
Secondary circ. steam mass flow rate	$m_{sg,ss}$	119.31	kg/s
Secondary circ. water mass flow rate	$m_{sg,sw}$	119.31	kg/s
Secondary circ. inlet temperature (disturbance)	$T_{sg,sw}$	220.85	°C
Number of steam generators	n_{sg}	6	
Power transferred to the steam generators	$n_{sg}W_{sg}$	13.351×10^8	W
Functions			
Saturated vapor pressure	$p_{*,T}(T)$		kPa
Coefficients for quadratic approximation	c_0	28884.78	kPa
	c_1	258.01	kPa/°C
	c_2	0.63455	$kPa/°C^2$
Water density	φ(T)		kg/m^3
Coefficients for quadratic approximation	$c_{\varphi,0}$	581.2	kg/m^3
	$c_{\varphi,1}$	2.98	$kg/m^3/°C$
	$c_{\varphi,2}$	0.00848	$kg/m^3/°C^2$

Table 1: Model parameters

proportional to a mean value between the cold and the hot leg temperatures, with a drift to give a proper value [16]

$$l_{pr,\mathrm{ref}} = c_{r,1}(T_{pc,cl} + T_{pc,hl}) - c_{r,2} = 2c_{r,1}T_{pc} - c_{r,2}$$

the inventory control for the pressurizer water level l_{pr} is given by

$$\begin{aligned}
\dot{I}_{e_{l_{pr}}} &= l_{pr} - l_{pr,\mathrm{ref}} \\
m_{in} &= \frac{A_{pr}}{\psi(M_{pc}, T_{pc})}\Bigg[-\Big(k_p(l_{pr} - l_{pr,\mathrm{ref}}) + k_i I_{e_{l_{pr}}}\Big)\varphi^2(T_{pc}) + m^{\circ}_{out}\varphi(T_{pc}) \\
&\quad + \frac{1}{c_{p,pc}}\Big(\frac{2c_{r,1}}{M_{pc}}\varphi^2(T_{pc}) + \frac{\partial\varphi(T_{pc})}{\partial T_{pc}}\Big)\Big(c_{p,pc}m^{\circ}_{out}\Delta^{\circ} + c_{\psi}N - n_{sg}k_{t,sg}(T_{pc} - T_{sg}) - W^{\circ}_{loss,pc}\Big)\Bigg]
\end{aligned} \tag{3}$$

with $k_p, k_i > 0$,

$$\begin{aligned}
\psi(M_{pc}, T_{pc}) &= \varphi(T_{pc}) - (T^{\circ}_{pc,i} - T_{pc})\frac{\partial\varphi(T_{pc})}{\partial T_{pc}} - \frac{2c_{r,1}A_{pr}}{M_{pc}}(T^{\circ}_{pc,i} - T_{pc})\varphi^2(T_{pc}) \\
\varphi(T_{pc}) &= c_{\varphi,0} + c_{\varphi,1}T_{pc} - c_{\varphi,2}T^2_{pc} \\
\frac{\partial\varphi(T_{pc})}{\partial T_{pc}} &= c_{\varphi,1} - 2c_{\varphi,2}T_{pc}.
\end{aligned}$$

Due to the the nominal disturbance values Δ°, $T^{\circ}_{pc,i}$, m°_{out}, $W^{\circ}_{loss,pc}$, and to their variations $\delta_{\Delta} = \Delta - \Delta^{\circ}$, $\delta_{T_{pc,i}} = T_{pc,i} - T^{\circ}_{pc,i}$, $\delta_{m_{out}} = m_{out} - m^{\circ}_{out}$, $\delta_{W_{loss,pc}} = W_{loss,pc} - W^{\circ}_{loss,pc}$, this controller ensures practical exponential stability of the error level, i.e. $e_{l_{pr}}$, $\dot{e}_{l_{pr}}$ tend to a neighborhood of the origin of radius

$$\mu = \frac{\kappa}{\vartheta\lambda^{Q}_{\min}}\|P\|\delta_{\max}$$

where $\kappa = \max_t \|\Psi\|$,

$$\Psi = \frac{1}{\varphi^2(T_{pc})}\frac{1}{A_{pr}}\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ -\frac{\partial\varphi(T_{pc})}{\partial T_{pc}}m_{in} & -\varphi(T_{pc}) - \frac{\partial\varphi(T_{pc})}{\partial T_{pc}}\Delta^{\circ} & -\frac{\partial\varphi(T_{pc})}{\partial T_{pc}}m^{\circ}_{out} & \frac{1}{c_{p,pc}}\frac{\partial\varphi(T_{pc})}{\partial T_{pc}} & -\frac{\partial\varphi(T_{pc})}{\partial T_{pc}} \end{pmatrix}.$$

and $\vartheta \in (0,1)$, P is solution of $PA + A^T P = -2Q$ for a fixed $Q = Q^T > 0$, $\lambda^{Q}_{\min}$ is the minimum eigenvalue of Q,

$$A = \begin{pmatrix} 0 & 1 \\ -k_i & -k_p \end{pmatrix}$$

and $\delta_{\max}$ is the maximum variation with respect to the nominal disturbance values.

As far as the pressurizer pressure controller is concerned, a first nonlinear dynamic controller can be designed transforming the pressurizer pressure reference $p_{pr,\mathrm{ref}}$ into a pressurizer water reference temperature $T_{pr,\mathrm{ref}}$, inverting the relation obtained from (2)

$$p_{pr,\mathrm{ref}} = c_0 - c_1 T_{pr,\mathrm{ref}} + c_2 T^2_{pr,\mathrm{ref}}$$

that can be uniquely inverted about the operating point of pressurizer temperature

$$T_{pr,\mathrm{ref}} = \frac{c_1 + \sqrt{c_1^2 - 4c_2(c_0 - p_{pr,\mathrm{ref}})}}{2c_2}$$
$$\dot{T}_{pr,\mathrm{ref}} = \frac{2\dot{p}_{pr,\mathrm{ref}}}{\sqrt{c_1^2 - 4c_2(c_0 - p_{pr,\mathrm{ref}})}}.$$

The dynamic controller, which depends on the (measured) temperature $T_{pr,wall}$, but not on the (unmeasured) temperature T_{pr}

$$\begin{aligned} W_{heat,pr,\mathrm{ref}} = c_{p,pr}M_{pr}\Big[\dot{T}_{pr,\mathrm{ref}} + \frac{k_{wall}}{c_{p,pr}M_{pr}}(T_{pr,\mathrm{ref}} - T_{pr,wall,\mathrm{ref}}) \\ - \delta_{pr}\Big(\frac{c_{p,pc}m^\circ_{pr}}{c_{p,pr}M_{pr}}(T_{pc} + \Delta^\circ) - \frac{m^\circ_{pr}}{M_{pr}}T_{pr,\mathrm{ref}}\Big)\Big] \\ W_{heat,pr} = u_{e,W} + W_{heat,pr,\mathrm{ref}} \end{aligned} \tag{4}$$

$$\begin{aligned} \dot{\hat{T}}_{pr} &= -\left(\frac{m^\circ_{pr}}{M_{pr}} + \frac{k_{wall}}{c_{p,pr}M_{pr}}\right)\hat{T}_{pr} + \frac{k_{wall}}{c_{p,pr}M_{pr}}T_{pr,wall} + \frac{1}{c_{p,pr}M_{pr}}W_{heat,pr} + \frac{c_{p,pc}m^\circ_{pr}}{c_{p,pr}M_{pr}}(T_{pc} + \Delta^\circ) \\ \dot{T}_{pr,wall,\mathrm{ref}} &= \frac{k_{wall}}{c_{p,wall}}T_{pr,\mathrm{ref}} - \frac{k_{wall}}{c_{p,wall}}T_{pr,wall,\mathrm{ref}} + k_i I_{e_{T_{pr,wall}}} - \frac{1}{c_{p,wall}}W^\circ_{loss,pr} \\ \dot{I}_{e_{T_{pr,wall}}} &= T_{pr,wall} - T_{pr,wall,\mathrm{ref}} \\ W_{heat,pr} &= -k_{wall}(T_{pr,wall} - T_{pr,wall,\mathrm{ref}}) - \frac{c_{p,pr}}{c_{p,wall}}k_{wall}M_{pr}B^T P\begin{pmatrix} I_{e_{T_{pr,wall}}} \\ T_{pr,wall} - T_{pr,wall,\mathrm{ref}} \end{pmatrix} \\ & \quad c_{p,pr}M_{pr}\Big[\dot{T}_{pr,\mathrm{ref}} + \frac{k_{wall}}{c_{p,pr}M_{pr}}(T_{pr,\mathrm{ref}} - T_{pr,wall,\mathrm{ref}}) - \delta_{pr}\Big(\frac{c_{p,pc}m^\circ_{pr}}{c_{p,pr}M_{pr}}(T_{pc} + \Delta^\circ) - \frac{m^\circ_{pr}}{M_{pr}}T_{pr,\mathrm{ref}}\Big)\Big] \end{aligned} \tag{5}$$

ensures the practical exponential stability of the error temperatures, with $k_i > 0$, $k_p = \frac{k_{wall}}{c_{p,wall}} > 0$, $T_{pr,wall,\mathrm{ref}}(0) = T_{pr,\mathrm{ref}}(0) - W^\circ_{loss,pr}/k_{wall}$, $M_{pr} = M_{pc} - \varphi(T_{pc})V_{pc,0}$,

$$\begin{aligned} m^\circ_{pr} = m_{in} - m^\circ_{out} - \frac{1}{c_{p,pc}M_{pc}}\frac{\partial\varphi(T_{pc})}{\partial T_{pc}}V_{pc,0}\Big[c_{p,pc}m_{in}(T^\circ_{pc,i} - T_{pc}) + c_{p,pc}m^\circ_{out}\Delta^\circ \\ + c_\psi N - n_{sg}k_{t,sg}(T_{pc} - T_{sg}) - W^\circ_{loss,pc}\Big] \end{aligned}$$

and $P = P^T > 0$ solution of the Lyapunov equation

$$PA + A^T P = -2Q, \qquad A = \begin{pmatrix} 0 & 1 \\ -k_i & -\frac{k_{wall}}{c_{p,wall}} \end{pmatrix}$$

for a fixed $Q = Q^T > 0$, and $B = \begin{pmatrix} 0 & 1 \end{pmatrix}^T$.

An alternative (dynamic switching nonlinear) controller for the pressurizer pressure is given by

$$\begin{aligned}
\dot{I}_{e_{ppr}} &= c_0 - c_1\hat{T}_{pr} + c_2\hat{T}_{pr}^2 - p_{pr,\text{ref}} \\
\dot{\xi} &= \hat{T}_{pr} - T_{pr,wall} - \frac{1}{k_{wall}}W^{\circ}_{loss,pr} - \frac{1}{k}\frac{1}{c_{p,pr}M_{pr}}C_{pr} \\
\hat{T}_{pr} &= \kappa\Big(\frac{c_{p,wall}}{k_{wall}}T_{pr,wall} - \xi\Big) \\
C_{pr} &= \frac{c_{p,pr}M_{pr}}{-c_1 + 2c_2\hat{T}_{pr}}\Big(\dot{p}_{pr,\text{ref}} - K_p\big(c_0 - c_1\hat{T}_{pr} + c_2\hat{T}_{pr}^2 - p_{pr,\text{ref}}\big) - K_i I_{e_{ppr}}\Big) \\
W_{heat,pr} &= k_{wall}(\hat{T}_{pr} - T_{pr,wall}) + C_{pr} + \delta_{pr}\big(c_{p,pr}m^{\circ}_{pr}\hat{T}_{pr} - c_{p,pc}m^{\circ}_{pr}(T_{pc} + \Delta^{\circ})\big)
\end{aligned} \tag{6}$$

where $\kappa, K_p, K_i > 0$, $I_{e_{ppr}}(0) = 0$, $\xi(0) = -\hat{T}_{pr}(0)/k + c_{p,wall}T_{pr,wall}(0)/k_{wall}$ and, as already defined, δ_{pr} is 1 if $m_{pr} > 0$ and 0 otherwise.

1.4 Digital Implementation of the Control Laws

We have already noted that the implementation of control laws by means of digital devices determines a disturbances acting on the feedback system. In this section we consider a further aspect, the digital implementation of derivatives. In fact, in (3), (5), (6) the derivative $\dot{I}_{e_{l_{pr}}}$, $\dot{\hat{T}}_{pr}$, $\dot{T}_{pr,wall,\text{ref}}$, $\dot{I}_{e_{T_{pr,wall}}}$, $\dot{I}_{e_{ppr}}$, $\dot{\xi}$ have to be implemented numerically. Let us consider a feedback system with $u = \alpha(x)$ a certain control law applied to control the system $\dot{x} = \bar{f}(x, u)$. The closed–loop dynamics is hence

$$\dot{x} = \bar{f}(x, \alpha(x)) := f(x)$$

A simple method to approximate by a digital computer the real time solution of this differential equation is to consider the Euler's rule, relying on the definition of derivative

$$\dot{x} = \lim_{\Delta t \to 0} \frac{\Delta x}{\Delta t}.$$

Here Δx is the change in x over the time interval Δt. Even if Δt does not tends to zero, the previous relation is approximately true

$$\dot{x}|_{t_k} \simeq \frac{x_{k+1} - x_k}{\delta}$$

where $\dot{x}|_{t_k}$ is the derivative of $x(t)$ calculated at time $t_k = k\delta$, $\delta = t_{k+1} - t_k$ is the sampling interval, $k \in \mathbb{Z}$, and $x_k = x(k\delta)$, $x_{k+1} = x((k+1)\delta)$ are the value of $x(t)$ at $t_k = k\delta$, $t_{k+1} = (k+1)\delta$. This is the so called forward rectangular rule, and leads to the following expression

$$x_{k+1} = x_k + \delta f(x_k).$$

There exists also a backward rectangular rule

$$\dot{x}(k) \simeq \frac{x_k - x_{k-1}}{\delta}$$

leading to

$$x_{k+1} = x_k + \delta f(x_{k+1}).$$

Another method is the trapezoidal rule, where eventually one gets

$$x_{k+1} = x_k + \frac{\delta}{2}\Big(f(x_k) + f(x_{k+1})\Big).$$

These approximations can be used in place of the derivatives that appear in the controller differential equations, to obtain difference equations repetitively solved with time steps of length δ. In the following we will consider the Euler's rule, for the sake of clarity, but the same arguments can be used with the other integration methods.

When dealing with the classical proportional, integral, and derivative control actions

$$u_p(t) = k_p e(t), \qquad u_i(t) = \frac{k_p}{T_i}\int_0^t e(\tau)\,d\tau, \qquad u_d(t) = k_p T_d \dot{e}(t)$$

with k_p the proportional gain, T_i the integral time, T_d the derivative time, they can be approximated with the following algebraic relations, which can be implementable with digital computers

$$u_{p,k} = k_p e_k, \qquad u_{i,k} = u_{i,k-1} + \frac{k_p}{T_i}\delta e_k, \qquad u_{d,k} = k_p T_d \frac{e_k - e_{k-1}}{\delta}$$

where $u_{i,k}$, $u_{d,k}$ are the results of the forward rule of the Euler approximation. Usually, these control actions are used together and their combination need to be done carefully. Hence, considering the Laplace transform for a classical (linear) PID controller

$$G(s) = \frac{u(s)}{e(s)} = k_p\left(1 + \frac{1}{T_i s} + T_d s\right)e(s)$$

so that

$$su(s) = k_p\left(s + \frac{1}{T_i} + T_d s^2\right)e(s)$$

and

$$\dot{u} = k_p\left(\dot{e} + \frac{1}{T_i}e + T_d\ddot{e}\right)$$

the Euler's method, applied twice for $\ddot{e}$, gives

$$u_k = u_{k-1} + k_p\left[\left(1 + \frac{\delta}{T_i} + \frac{T_d}{\delta}\right)e_k - \left(1 - 2\frac{T_d}{\delta}\right)e_{k-1} + \frac{T_d}{\delta}e_{k-2}\right].$$

For linear systems (and controllers) with bandwidths of a few Hz, sample rates are often on the order of 100 Hz, so that δ is on the order of 10^{-2} s, and errors from the approximation is quite small. An empirical rule is that the sample rate should be faster that 30 times the bandwidth in order to assure that the digital controller can be made to closely match the performance of the continuous controller. Except

Shannon rule, regarding the aliasing problem, there is not a theoretic rule to fix δ in order to ensure this close match.

The absence of a systematic rule to choose the sampling period δ is even more problematic for nonlinear systems. In fact, it is well–known that nonlinear systems can experience a finite escape times, which roughly means that the state can go to infinity in finite time intervals [8]. This is a great difference with linear systems, where the state can go to infinity asymptotically, namely when time goes to infinity. As a result, from a conservative point of view, the controllers need to be implemented with the fastest sampling time.

With this important observation in mind, the controllers (3), (5), (6), containing PI terms, can be implemented as follows.

Digital implementation of the pressurizer water level controller (3)

$$
\begin{aligned}
I_{e_{l_{pr}},k+1} &= I_{e_{l_{pr}},k} + \delta(l_{pr,k} - l_{pr,\mathrm{ref},k}) \\
m_{in,k} &= \frac{A_{pr}}{\psi(M_{pc,k},T_{pc,k})}\Bigg[-\Big(k_p(l_{pr,k} - l_{pr,\mathrm{ref},k}) + k_i I_{e_{l_{pr}},k}\Big)\varphi^2(T_{pc,k}) + m^\circ_{out}\varphi(T_{pc,k}) \\
&\quad + \frac{1}{c_{p,pc}}\Big(\frac{2c_{r,1}}{M_{pc,k}}\varphi^2(T_{pc,k}) + \frac{\partial\varphi(T_{pc})}{\partial T_{pc}}\Big|_k\Big)\Big(c_{p,pc}m^\circ_{out}\Delta^\circ + c_\psi N_k - n_{sg}k_{t,sg}(T_{pc,k} - T_{sg,k}) - W^\circ_{loss,pc}\Big)\Bigg]
\end{aligned}
\tag{7}
$$

with $k_p, k_i > 0$,

$$
\begin{aligned}
\psi(M_{pc,k},T_{pc,k}) &= \varphi(T_{pc,k}) - (T^\circ_{pc,i} - T_{pc,k})\frac{\partial\varphi(T_{pc})}{\partial T_{pc}}\Big|_k - \frac{2c_{r,1}A_{pr}}{M_{pc,k}}(T^\circ_{pc,i} - T_{pc,k})\varphi^2(T_{pc,k}) \\
\varphi(T_{pc,k}) &= c_{\varphi,0} + c_{\varphi,1}T_{pc,k} - c_{\varphi,2}T^2_{pc,k} \\
\frac{\partial\varphi(T_{pc})}{\partial T_{pc}}\Big|_k &= c_{\varphi,1} - 2c_{\varphi,2}T_{pc,k}.
\end{aligned}
$$

Digital implementation of the pressurizer pressure controller (5)

$$\begin{aligned}
\hat{T}_{pr,k+1} &= \hat{T}_{pr,k} + \delta\Bigg[-\left(\frac{m^{\circ}_{pr,k}}{M_{pr,k}} + \frac{k_{wall}}{c_{p,pr}M_{pr,k}}\right)\hat{T}_{pr,k} + \frac{k_{wall}}{c_{p,pr}M_{pr,k}}T_{pr,wall,k} \\
&\qquad + \frac{1}{c_{p,pr}M_{pr,k}}W_{heat,pr,k} + \frac{c_{p,pc}m^{\circ}_{pr,k}}{c_{p,pr}M_{pr,k}}(T_{pc,k} + \Delta^{\circ})\Bigg] \\
T_{pr,wall,\mathrm{ref},k+1} &= T_{pr,wall,\mathrm{ref},k} + \delta\left(\frac{k_{wall}}{c_{p,wall}}T_{pr,\mathrm{ref},k} - \frac{k_{wall}}{c_{p,wall}}T_{pr,wall,\mathrm{ref},k} + k_i I_{e_{T_{pr,wall}},k} - \frac{1}{c_{p,wall}}W^{\circ}_{loss,pr}\right) \\
I_{e_{T_{pr,wall}},k+1} &= I_{e_{T_{pr,wall}},k} + \delta\left(T_{pr,wall,k} - T_{pr,wall,\mathrm{ref},k}\right) \\
W_{heat,pr,k} &= -k_{wall}(T_{pr,wall,k} - T_{pr,wall,\mathrm{ref},k}) - \frac{c_{p,pr}}{c_{p,wall}}k_{wall}M_{pr,k}B^T P\begin{pmatrix} I_{e_{T_{pr,wall}},k} \\ T_{pr,wall,k} - T_{pr,wall,\mathrm{ref},k}\end{pmatrix} \\
&\quad + c_{p,pr}M_{pr,k}\Bigg[\dot{T}_{pr,\mathrm{ref}}\Big|_k + \left(\frac{m^{\circ}_{pr,k}}{M_{pr,k}} + \frac{k_{wall}}{c_{p,pr}M_{pr,k}}\right)T_{pr,\mathrm{ref},k} - \frac{k_{wall}}{c_{p,pr}M_{pr,k}}T_{pr,wall,\mathrm{ref},k} \\
&\qquad - \frac{c_{p,pc}m^{\circ}_{pr,k}}{c_{p,pr}M_{pr,k}}(T_{pc,k} + \Delta^{\circ})\Bigg] \\
T_{pr,\mathrm{ref},k} &= \frac{c_1 + \sqrt{c_1^2 - 4c_2(c_0 - p_{pr,\mathrm{ref},k})}}{2c_2}, \qquad \dot{T}_{pr,\mathrm{ref}}\Big|_k = \frac{2\dot{p}_{pr,\mathrm{ref},k}}{\sqrt{c_1^2 - 4c_2(c_0 - p_{pr,\mathrm{ref},k})}}
\end{aligned} \tag{8}$$

with $T_{pr,wall,\mathrm{ref},0} = T_{pr,\mathrm{ref},0} - W^{\circ}_{loss,pr}/k_{wall}$, $M_{pr,k} = M_{pc,k} - \varphi(T_{pc,k})V_{pc,0}$,

$$\begin{aligned}
m^{\circ}_{pr,k} &= m_{in,k} - m^{\circ}_{out} - \frac{1}{c_{p,pc}M_{pc,k}}\frac{\partial\varphi(T_{pc})}{\partial T_{pc}}\Big|_k V_{pc,0}\Big[c_{p,pc}m_{in,k}(T^{\circ}_{pc,i} - T_{pc,k}) + c_{p,pc}m^{\circ}_{out}\Delta^{\circ} \\
&\qquad + c_{\psi}N_k - n_{sg}k_{t,sg}(T_{pc,k} - T_{sg,k}) - W^{\circ}_{loss,pc}\Big].
\end{aligned}$$

Digital implementation of the pressurizer pressure controller (6)

$$\begin{aligned}
I_{e_{ppr},k+1} &= I_{e_{ppr},k} + \delta\left(c_0 - c_1\hat{T}_{pr} + c_2\hat{T}_{pr}^2 - p_{pr,\mathrm{ref}}\right) \\
\xi_{k+1} &= \xi_k + \delta\left(\hat{T}_{pr} - T_{pr,wall} - \frac{1}{k_{wall}}W^{\circ}_{loss,pr} - \frac{1}{k}\frac{1}{c_{p,pr}M_{pr}}C_{pr}\right) \\
\hat{T}_{pr,k} &= \kappa\left(\frac{c_{p,wall}}{k_{wall}}T_{pr,wall,k} - \xi_k\right) \\
C_{pr,k} &= \frac{c_{p,pr}M_{pr,k}}{-c_1 + 2c_2\hat{T}_{pr,k}}\left(\dot{p}_{pr,\mathrm{ref},k} - K_p\left(c_0 - c_1\hat{T}_{pr,k} + c_2\hat{T}_{pr,k}^2 - p_{pr,\mathrm{ref},k}\right) - K_i I_{e_{ppr},k}\right) \\
W_{heat,pr,k} &= k_{wall}(\hat{T}_{pr,k} - T_{pr,wall,k}) + C_{pr,k} + \delta_{pr}\left(c_{p,pr}m^{\circ}_{pr,k}\hat{T}_{pr,k} - c_{p,pc}m^{\circ}_{pr,k}(T_{pc,k} + \Delta^{\circ})\right)
\end{aligned} \tag{9}$$

where $I_{e_{ppr},0} = 0$, $\xi_0 = -\hat{T}_{pr,0}/\kappa + c_{p,wall}T_{pr,wall,0}/k_{wall}$.

2 Self Triggered Robust Strategies for Optimal Implementations of Control Laws on Digital Devices

The implementation of controllers with digital devices presents many advantages, but at the same time poses some issues. We have already mentioned about the fact that the implementation of control laws with zero order holders, commonly used for the digital implementation of control laws, introduce a delay and hence could bring to unstable behaviors. We have also mentioned the possibility of determining a region of practical stability of the control system. In this section we want to be more specific and consider a technique, called self triggered control, determining the sampling time so that the control system performances are preserved. The obvious property to be preserved is the asymptotic stability, which in turn means that the plant is operating safely.

One important aspect is the design of the digital controller so that the control system recovers, at least in first approximation, the same behavior of a system controlled with a continuous time controller. In fact, many authors propose nonlinear digital controllers reproducing the performances of a certain continuous controller, viz emulating the behavior of the continuous controller [34] [29], [30]. This a very popular technique relies on the simple consideration that when the (fixed) sampling period is short enough, one regains the continuous behavior. Other authors aim at designing the digital controller directly in the digital setting, imposing control performances on the digital model of the system, although nonlinear systems cannot be discretized exactly in closed form, in general. In both cases, a relevant problem arises: the determination of the sampling period. From a theoretical point of view, the sampling period is usually considered constant, namely the new control value is computed periodically at each sampling time. This helps, from a mathematical point of view, the analysis of the sampled nonlinear system, and gives some mathematical tools to solve the design problem. However, it is clear that a better solution should be that of calculating the controller only "when necessary". This clearly complicates the problem from an analytical point of view. But there are also practical aspects that push to deal with variable sampling. A first aspect is that a constant sampling is quite inefficient (in terms of CPU usage, communication bandwidth, energy, etc.), since it has to be considered the worst–case scenario. In fact, since the system dynamics are nonlinear, one has to ensure good performance for all the operative

points. Incidentally, this reveals the need of criteria to fix the sampling time value, that otherwise has to be fixed using (possibly extensive) simulations, or considering empirical rules (20 times the system bandwidth [20]). A further aspect is that many digital controller perform various tasks at the same time. This is quite typical, especially in the case of embedded systems. An example is the electronic central unit in an automobile, which has to manage different tasks, with different priority. Their scheduling is clearly of critical importance to prevent negative coupling effects of lower priority tasks when computing high level tasks, such as attitude control laws. Another important example are the networked systems, where not only the processor time is a resource to be optimized, but also the available communication bandwidth is limited. In wireless sensor networks, furthermore, an important issue is the minimization of the power consumption, in order to augment the life span of the network. In all these applications, the energy consumption is related to the frequency of measurements and transmission over the network. It is clear that in these cases measurement/computation/actuation data transmission should be minimized and should occur only "when necessary".

Among the various techniques proposed to face this problem, the event triggered technique seems to be promising [21], [31], [14], [22]. This technique formalize the statement "when necessary": the measurement/computation/actuation data transmission event occurs when the state of the system assumes certain values. Clearly, this technique requires the continuous measurement of the state. To circumvent this drawback, self–triggered techniques have been proposed. In this case the controller determines its next execution time, and does not require continuous measurements of the state. In particular, when the stabilization of the system origin is considered, this event is triggered only when the asymptotic stability property, as formalized by the Lyapunov approach, can be lost [14], [4], [13], [36], [37], [5], [18], [6]. This approach can be also generalized to a weaker property such as safety [6].

Self triggered control strategies have been introduced in [33], where a heuristic rule is provided to self–trigger the next execution time of a control task on the basis of the last measurement of the state. In [11], [12], a robust self triggered strategy is proposed, which guarantees that the $\mathcal{L}_2$ gain of a linear time invariant system is kept under a given threshold. In [13] a self triggered strategy distributed over a wireless sensor network is proposed for linear time invariant systems.

In [14] sufficient conditions for the existence of a stabilizing event–triggered control strategy are given for nonlinear systems. In [5] the authors propose a self–triggered emulation of the event–triggered control strategy proposed in [14]. In particular a methodology for the computation of the next execution time as a function of the last sample is presented, under a homogeneity condition.

In this deliverable, a methodology for the computation of the next execution time is proposed, based on polynomial approximations of Lyapunov functions, and relying on the assumption that the nonlinear

differential equations and the control law are C^ℓ functions, with ℓ sufficiently large. This assumption is verified in the present case. The next sampling time is also computed in the presence of bounded sensing/computation/actuation delays and of norm–bounded parameter uncertainties and disturbances. Moreover, under weaker conditions than those used in [14], it is proved the existence of a self triggered strategy keeping the state in a "safe set" arbitrarily close to the equilibrium point, and a methodology for computing the next execution time is provided. Fixed a δ–ball of the equilibrium point and a disturbance that is upper bounded in norm by a class $\mathcal{K}$ function $\nu(\delta)$, a methodology is presented for the computation of the next execution time that depends on the δ boundary defining the safe set.

2.1 Problem Formulation

Consider a generic nonlinear system

$$\dot{x} = f(x, u, \mu, d) \tag{10}$$

where $x \in \mathcal{D}_x \subset r^n$, $\mathcal{D}_x$ a domain containing the origin, $u \in \mathcal{D}_u \subset r^p$, μ is a parameter uncertainty vector varying in a compact set $\mathcal{D}_\mu \subset r^r$, with $0 \in \mathcal{D}_\mu$, d is an external bounded disturbance vector taking values in a compact set $\mathcal{D}_d \subset r^s$, with $0 \in \mathcal{D}_d$. In the following, we may refer to (10) as the perturbed system. Furthermore, we define the nominal (or "unperturbed") system associated to the "perturbed" system (10) as

$$\dot{x} = f_0(x, u) \doteq f(x, u, 0, 0). \tag{11}$$

Given a state feedback control law $\kappa: \mathcal{D}_x \to \mathcal{D}_u$, the closed loop perturbed system is

$$\dot{x} = f(x, \kappa(x), \mu, d) \tag{12}$$

and the closed loop nominal system is

$$\dot{x} = f_0(x, \kappa(x)). \tag{13}$$

We will denote by $x(t)$, $t \geq t_0$, the solution of the closed loop system (12) (or (13), according to the context), with initial condition $x_0 = x(t_0)$. Given a state feedback control law κ it is well–known that, if κ locally stabilizes the origin of system (13) and if $f_0(x, \kappa(x)) \in C^\ell(\mathcal{D}_x)$, $\ell > 1$ integer, then there exists a Lyapunov function $V(x)$ of class $C^1(\mathcal{D}_x)$ such that

$$\begin{aligned} \alpha_1(\|x\|) \leq V(x) \leq \alpha_2(\|x\|) \\ \frac{\partial V(x)}{\partial x} f_0(x, \kappa(x)) \leq -\alpha_3(\|x\|) \\ \left\| \frac{\partial V(x)}{\partial x} \right\| \leq \alpha_4(\|x\|) \end{aligned} \tag{14}$$

with $\alpha_1, \alpha_2, \alpha_3, \alpha_4 \in \mathcal{K}$ [8], [10], [35].

Moreover, given a state feedback control law κ, we say that system (12) is safe with respect to the set $\mathcal{S} \subseteq \mathcal{D}_x$ for the time interval $\mathcal{T} \subseteq r_0^+$, if $x(t) \in \mathcal{S}$, $\forall t \in \mathcal{T}$.

The feedback control signal $u(t) = \kappa(x(t))$ requires continuous measurements of the state of the system. We assume that state measurements are available at sampling times t_k, which define a sequence $\mathcal{I} = \{t_k\}_{k\geq 0}$, and that the applied control is

$$u_{\mathcal{I}}(t) = \begin{cases} 0 & \forall t \in [t_0, t_0 + \Delta_0) \\ \kappa(x_k) & \forall t \in [t_k + \Delta_k, t_{k+1} + \Delta_{k+1}),\ k \geq 0 \end{cases} \tag{15}$$

where $\{\Delta_k\}_{k\geq 0}$ is a sequence of delays, due to the transmission time from the sensor to the controller, the computation time, and the transmission time from the controller to the actuator. On the basis of this assumption, we address the following problems.

Problem 1 *Given a system (11), and a state feedback control law κ such that the origin of (13) is asymptotically stable with region of attraction $\Omega \subset \mathcal{D}_x$ containing the origin, determine a function $\tau_s : \mathcal{D}_x \to [\tau_{\min}, \infty), \tau_{\min} > 0$ and a maximum allowed delay $\Delta_{\max} \in [0, \tau_{\min}]$ such that if the sequence of sampling instants $\mathcal{I}$ is inductively defined by*

$$t_{k+1} = t_k + \tau_s(x_k) \tag{16}$$

and if the delays are such that

$$\Delta_k \in [0, \Delta_{\max}), \quad \forall\, k \geq 0, \tag{17}$$

then the origin of the closed loop system (13) with control input signal $u_{\mathcal{I}}(t)$ as in (15) is asymptotically stable with region of attraction Ω. ⋄

Problem 2 *Given a system (10) (resp. (11)), and a state feedback control law κ such that the origin of (12) (resp. (13)) is asymptotically stable with region of attraction $\Omega \subset \mathcal{D}_x$ containing the origin, and an arbitrary safe set $\mathcal{B}_\delta = \{x \in r^n \mid \|x\| < \delta\} \subset \Omega$, $\delta > 0$, determine τ_s and $\Delta_{\max}$ as defined in Problem 1 such that if $\mathcal{I}$ is inductively defined by (16) and if Δ_k satisfies (17), then the closed loop system (12) with control input signal $u_{\mathcal{I}}(t)$ as in (15) is safe with respect to $\mathcal{B}_\delta$ for the time interval $[t_0, \infty)$.* ⋄

In Problems 1 and 2, the function τ_s is used to determine the next sampling instant as a function of the current measurement of the system. The purpose is to obtain a self triggered control system that is robust with respect to delays bounded by a design parameter $\Delta_{\max}$. By choosing the next sampling instant t_{k+1} as a function of the current measurement at time t_k, we perform sampling only when needed for guaranteeing asymptotic stability or safety. The aim is to determine a sampling instant sequence $\mathcal{I}$

such that the intersampling time $t_{k+1} - t_k$ is as large as possible, in order to reduce transmission power of the sensing and actuation data transmissions, and to reduce the CPU effort due to the computation of the control.

As a comparison of the above definitions with the concepts of Maximally Allowable Transmission Interval (MATI) and Maximally Allowable Delay (MAD) introduced in [7], we can interpret $\Delta_{\max}$ as the (*global*) MAD, and $t_{k+1} - t_k = \tau_s(x_k) - t_k$ as the (*local*) MATI of the system in the time interval $[t_k, t_{k+1}]$ on the basis of the measurement $x_k = x(t_k)$ of the state $x(t)$ at $t = t_k$.

2.2 Self Triggered Stabilizing Control

The results developed in this section address Problem 1 for system (11), and are based on the following assumptions, which are weaker than those required in [5] (viz., homogeneity of the closed loop dynamics) to compute the next sampling time as a function τ_s of the current state of the system.

Assumption 1 *Assume that*

1. *$f_0 \in C^\ell(\mathcal{D}_x \times \mathcal{D}_u)$, with ℓ a positive integer sufficiently large;*
2. *There exists a nonempty set $\mathcal{U}$ of state feedback laws $\kappa: \mathcal{D}_x \to \mathcal{D}_u$, such that $\kappa \in C^\ell(\mathcal{D}_x)$ and the origin of (13) is asymptotically stable, with region of attraction a certain compact $\Omega \subset \mathcal{D}_x$ containing the origin;*
3. *The functions $\alpha_3, \alpha_4 \in \mathcal{K}$ in (14) are such that α_3^{-1}, α_4 are Lipschitz.* ◇

The assumption of existence of a stabilizing control (i.e. non–emptiness of the set $\mathcal{U}$) is not restrictive, since if the nominal system cannot be stabilized using continuous time measurement and actuation, then it is clear that the nominal system cannot be stabilized using a digital control with zero–order holders. The main limitation of Assumption 1 is the Lipschitz condition on $\alpha_3^{-1}(\cdot)$ and $\alpha_4(\cdot)$. We will show how to weaken this assumption in Section 2.3, which will be devoted to safety control.

Theorem 1 *Under Assumption 1, Problem 1 is solvable for system (11), and the function τ_s can be iteratively computed as a function of the current state of the system and the maximum allowable delay $\Delta_{\max}$.* ◇

Proof: We first prove the result for $\Delta_k = 0$. Since $\mathcal{U}$ is not empty, by Assumption 1, we pick a state feedback control law $\kappa \in \mathcal{U}$. Since $f_0(x, \kappa(x)) \in C^\ell(\mathcal{D}_x)$ with $\ell > 1$, there exists a Lyapunov candidate

$V(x)$ that satisfies (14). Choose $r > 0$ such that the ball $B_r = \{x \in \Omega : \|x\| \leq r\} \subset \Omega$. For $x_k \in B_r$,

$$\begin{aligned} \dot{V} &= \frac{\partial V}{\partial x} f_0(x, \kappa(x_k)) = \frac{\partial V}{\partial x} f_0(x, \kappa(x)) + \frac{\partial V}{\partial x}\Big(f_0(x, \kappa(x_k)) - f_0(x, \kappa(x))\Big) \\ &\leq -\alpha_3(\|x\|) + \alpha_4(\|x\|)\|d_h\| \end{aligned} \tag{18}$$

where

$$d_h = f_0(x(t), \kappa(x_k)) - f_0(x(t), \kappa(x(t)))$$

can be considered as a perturbation due to the holding.

Under Assumption 1, there exists a $\delta_k > 0$ such that $\dot{x} = f(x, \kappa(x_k))$ has a unique solution over $[t_k, t_k + \delta_k]$. Hence, we can expand the components $d_{h,i}$ of d_h in Taylor series. Consider the i^{th} component $d_{h,i}$, $i = 1, \cdots, n$, of the n–dimensional vector d_h. One can expand each component in Taylor series with respect to $t \in [t_k, t_k + \delta_k]$, on the right of t_k, up to the 2^{nd} term, with Lagrange remainder of the 3^{rd} term

$$d_{h,i} = \varphi_{1,i}(x_k)(t - t_k) + \varphi_{2,i}(\bar{x}_i, x_k)(t - t_k)^2 \tag{19}$$

where

$$\varphi_{1,i}(x_k) = d_{h,i}\Big|_{x(t)=x_k}, \ \varphi_{2,i}(\bar{x}_i, x_k) = \frac{1}{2} \left.\frac{\mathrm{d}_+^2 d_{h,i}}{\mathrm{d}t^2}\right|_{x(t)=\bar{x}_i}$$

where $\frac{\mathrm{d}_+^n(\cdot)}{\mathrm{d}t^n}$ denotes the n–th right derivative. According to Taylor theorem with Lagrange remainder, there exists $\bar{t}_i \in [t_k, t]$, with $\bar{x}_i = x(\bar{t}_i)$, $i = 1, \cdots, n$, such that the equality (19) holds. Hence,

$$\|d_h\| \leq \|\varphi_1(x_k)\|(t - t_k) + \|\varphi_2(\bar{x}, x_k)\|(t - t_k)^2$$

where $\bar{x} \doteq (\bar{x}_1, \cdots, \bar{x}_n)$ and

$$\begin{aligned} \varphi_1(x_k) &\doteq \Big(\varphi_{1,1}(x_k), \cdots, \varphi_{1,n}(x_k)\Big)^T \\ \varphi_2(\bar{x}, x_k) &\doteq \Big(\varphi_{2,1}(\bar{x}_1, x_k), \cdots, \varphi_{2,n}(\bar{x}_n, x_k)\Big)^T. \end{aligned}$$

Consider the set $\Omega_{V(x_k)} \doteq \{x \in \Omega : V(x) \leq V(x_k)\}$, and define

$$M_1(x_k) \doteq \|\varphi_1(x_k)\|, \qquad M_2(x_k) \doteq \max_{\bar{x} \in \Omega_{V(x_k)}} \|\varphi_2(\bar{x}, x_k)\|.$$

Since $f, \kappa \in C^\ell$ and $\Omega_{V(x_k)}$ is compact, then $M_1(x_k)$ is finite for any $x_k \in \Omega_{V(x_k)}$, and $M_2(x_k) \in r^+$ exists and is finite for any $x_k \in \Omega_{V(x_k)}$.

Note that there exists a time interval $[t_k, t_{k+1} < t_k + \delta_k]$ such that

$$\alpha_4(\|x\|)\|d_h\| \leq \vartheta \alpha_3(\|x\|) \tag{20}$$

is satisfied for a fixed $\vartheta \in (0, 1)$. In fact, (20) is satisfied if

$$\alpha_3^{-1}\left(\frac{1}{\vartheta}\alpha_4(\|x\|)\Big(M_1(x_k)(t - t_k) + M_2(x_k)(t - t_k)^2\Big)\right) \leq \|x\|.$$

Since α_3^{-1} and α_4 are Lipschitz, then equation (20) is satisfied if

$$\frac{1}{\vartheta} L_{\alpha_3^{-1}} L_{\alpha_4} \|x\| \Big(M_1(x_k)(t - t_k) + M_2(x_k)(t - t_k)^2 \Big) \leq \|x\|$$

where $L_{\alpha_3^{-1}}, L_{\alpha_4} > 0$ are the Lipschitz constants of α_3^{-1}, α_4, respectively. The above equation directly implies that (20) is satisfied if

$$M_1(x_k)(t - t_k) + M_2(x_k)(t - t_k)^2 \leq \frac{\vartheta}{L_{\alpha_3^{-1}} L_{\alpha_4}}. \tag{21}$$

Hence, if we define

$$\tau_s(x_k) \doteq \max \{ t - t_k : \text{ (21) is satisfied for each } t - t_k \in [0, \tau_s(x_k)] \}$$

$$\tau_{\min} \doteq \min_{x_k \in \Omega_{V(x_k)}} \tau_s(x_k)$$

and we choose $t_{k+1} = t_k + \tau_s(x_k)$, then $\dot{V}(t) \leq -(1 - \vartheta)\alpha_3(\|x\|) < 0$ for all $t \in [t_k, t_{k+1}]$ and for all $k \geq 0$. This implies that the origin is asymptotically stable. Equation (21) is a second degree inequality in the form $ay^2 + by \leq c$, where a, b are non–negative and upper bounded for each $x_k \in \mathcal{D}_x$, and c is strictly positive and upper bounded. This trivially implies that $\tau_s(x_k)$ is strictly positive for each $x_k \in \Omega_{V(x_k)}$, and thus $\tau_{\min}$ is strictly positive as well. In this way, $\tau_s(\cdot)$ remains defined iteratively for each $k \geq 0$. This completes the proof for $\Delta_k = 0$.

For the case of $\Delta_k > 0$, following the same reasoning

$$\begin{aligned} \dot{V}(t) &= \frac{\partial V}{\partial x} f_0(x(t), \kappa(x_k)) = \frac{\partial V}{\partial x} f_0(x, \kappa(x)) + \frac{\partial V}{\partial x}(d_h + d_{\Delta_k}) \\ &\leq -\alpha_3(\|x\|) + \alpha_4(\|x\|)\|d_h\| + \alpha_4(\|x\|)\|d_{\Delta_k}\| \end{aligned}$$

for $t \geq t_k + \Delta_k$ where

$$d_h = f_0(x(t), \kappa(x(t_k + \Delta_k))) - f_0(x, \kappa(x))$$

$$d_{\Delta_k} = f_0(x(t), \kappa(x_k)) - f_0(x(t), \kappa(x(t_k + \Delta_k)))$$

can be considered as perturbations due to the holding and to the sensing/computation/actuation delay. Since also the solution $x(t)$ is Lipschitz, as well as f_0 and κ, then

$$\|d_{\Delta_k}\| \leq M_3 \Delta_k, \quad M_3 = L_{f_0} L_\kappa L_x$$

where L_{f_0}, L_κ, L_x are the Lipschitz constants of f_0, κ, x. Proceeding for d_h as in the previous case, we conclude that (20) is satisfied if

$$M_1(x_k)(t - t_k) + M_2(x_k)(t - t_k)^2 + M_3 \Delta_k \leq \frac{\vartheta}{L_{\alpha_3^{-1}} L_{\alpha_4}}. \tag{22}$$

Setting $\vartheta = \vartheta_1 + \vartheta_2$, with $\vartheta_1, \vartheta_2 \in (0, 1)$, equation (22) implies that the stability condition (20) holds if

$$M_1(x_k)(t - t_k) + M_2(x_k)(t - t_k)^2 \leq \frac{\vartheta_1}{L_{\alpha_3^{-1}} L_{\alpha_4}}, \tag{23}$$

and

$$\Delta_k \leq \Delta_{\max} \doteq \frac{\vartheta_2}{M_3 L_{\alpha_3^{-1}} L_{\alpha_4}}. \tag{24}$$

Defining

$$\tau_s(x_k) \doteq \max\{t - t_k : \text{ (23) is satisfied for each } t - t_k \in [0, \tau_s(x_k)]\} - \Delta_{\max}$$

$$\tau_{\min} \doteq \min_{x_k \in \Omega_{V(x_k)}} \tau_s(x_k)$$

and if we choose $t_{k+1} = t_k + \tau_s(x_k)$, then $\dot{V}(t) \leq -(1-\vartheta)\alpha_3(\|x\|) < 0$ for all $t \in [t_k + \Delta_k, t_{k+1} + \Delta_{\max}]$ and for all $k > 0$. This ensures the asymptotic stability of the origin. $\Delta_{\max}$ is non-negative, and for ϑ_2 sufficiently small $t_{k+1} - t_k = \tau_s(x_k) > \Delta_{\max} \geq 0$ for each $x_k \in \Omega_{V(x_k)}$. Therefore, $\tau_{\min}$ is strictly positive as well. This completes the proof. ■

Remark 1 *It is worth noting that $\tau_{\min} > 0$, as shown in the proof, implies that the time interval between two sampling instants is lower bounded by the minimum sampling time $\tau_{\min} > 0$, so that undesired Zeno behaviors are avoided.* ◇

Remark 2 *The choice of $\vartheta_1 \in (0,1)$ corresponds to a simple tradeoff between larger intersampling times $\tau_s(x_k)$ and robustness with respect to larger delays $\Delta_{\max}$. As ϑ_1 decreases, $\tau_s(x_k)$ decreases and $\Delta_{\max}$ increases. This implies that we improve robustness vs delays, paid by stronger sampling requirements.* ◇

Remark 3 *When applying the self triggered rule defined in the above theorem in a real scenario, it is necessary to compute on–line the next sampling time for each time instant t_k. This computation corresponds to solving a second degree equality, which is reasonable in an embedded system. On the contrary, the functions $M_1(\cdot)$ and $M_2(\cdot)$ can be determined off–line, and then (numerically) computed on–line in x_k. However, $M_2(\cdot)$ might be still difficult to determined in closed form. In this case, one can define*

$$M_2 \doteq \max_{\bar{x}, x_k \in \Omega} \|\varphi_2(\bar{x}, x_k)\|$$

and use it in equation (21) to compute the next sampling time. This new definition clearly implies shorter sampling times. ◇

The above remarks also apply to Theorems 2 and 3 in the following Sections.

2.3 Self Triggered Safety Control

The main limitation of the results developed in Section 2.2 is the Lipschitz continuity assumption of $\alpha_3^{-1}(\cdot)$ and $\alpha_4(\cdot)$. The following example shows that even exponentially stabilizable systems do not always satisfy this assumption.

Example 1 *Consider the system* $\dot{x} = Ax + Bu + f(x,u) = f_0(x,u)$ *with*

$$f_0(x,u) = \begin{pmatrix} -x_1 + x_2 + x_1^2 \\ (1+x_1)u \end{pmatrix}.$$

Let $u = \kappa(x) = -x_2 \in \mathcal{U}$. *Consider the Lyapunov candidate* $V(x) = x^T P x$, *with* P *solution of the Lyapunov equation* $PA_c + A_c^T P = -Q$, *with* $Q = 2I$, I *the identity matrix, and* $A_c = \begin{pmatrix} -1 & 1 \\ 0 & -1 \end{pmatrix}$. *Since* $P = \begin{pmatrix} 2 & 1 \\ 1 & 3 \end{pmatrix}$, *then* $\lambda^P_{\min} \cong 1.382$ *and* $\lambda^P_{\max} \cong 3.618$ *denote respectively the minimum and the maximum eigenvalue of* P. *For* $\|x\| \leq 2/3$, *the time derivative of* V *satisfies*

$$\dot{V} = -\|x\|_Q^2 + 2|x_1|^3 + 3|x_1|x_2^2 \leq -2x_1^2 - 2x_2^2 + 2(2/3)x_1^2 + 3(2/3)x_2^2 \leq -\frac{1}{2}\|x\|^2$$

thus the origin is locally exponentially stable, with $\alpha_1(\|x\|) = \lambda^P_{\min}\|x\|^2$, $\alpha_2(\|x\|) = \lambda^P_{\max}\|x\|^2$, $\alpha_3(\|x\|) = \|x\|^2/2$, $\alpha_4(\|x\|) = \lambda^P_{\max}\|x\|$.

It is clear that Assumption 1 is not satisfied, since $\alpha_3^{-1}(\cdot)$ *is not Lipschitz. For this reason, one the basis of the previous results, one can not ensure the existence of a stabilizing self triggered strategy.* ⋄

The main technical problem is that, if $\alpha_3^{-1}(\cdot)$ is not Lipschitz, the next sampling time $\tau_s(x_k)$ goes to zero as x_k approaches the equilibrium point, and this might generate Zeno behaviors. The results developed in this section address Problem 2, both for the nominal system (11) and the generic system (10), and are based on the following assumption, that does not require $\alpha_3^{-1}(\cdot)$ to be Lipschitz.

Assumption 2 *Assume that* $f_0 \in C^\ell(\mathcal{D}_x \times \mathcal{D}_u)$, *with* ℓ *a positive integer sufficiently large. Assume that there exists a nonempty set* $\mathcal{U}$ *of state feedback laws* $\kappa : \mathcal{D}_x \to \mathcal{D}_u$, *such that* $\kappa \in C^\ell(\mathcal{D}_x)$ *and the origin of the system (13) is asymptotically stable, with region of attraction a certain compact* $\Omega \subset \mathcal{D}_x$ *containing the origin.* ⋄

For system (11) (unperturbed case) we determine a function τ_s to compute the next sampling time as a function of the current state of the system and the maximum allowable delay $\Delta_{\max}$, such that the closed loop system applying a self triggered strategy is safe. On the basis of Assumption 2, in Theorem 2 we provide a different computation of τ_s providing less conservative (less frequent) sampling instants.

For system (10) (perturbed case), given a δ boundary of the equilibrium point and a disturbance that is upper bounded in norm by a class $\mathcal{K}$ function $\nu(\delta)$, we determine a function τ_s to compute the next sampling time as a function of the current state of the system and the maximum allowable delay $\Delta_{\max}$, such that the closed loop system applying a self triggered strategy is safe with respect to δ. We remark that, according to well known results in [8], a locally stable system subject to a bounded disturbance always satisfies a safety property with respect to δ sufficiently small. Nevertheless neither the computation

of the function τ_s nor the relation among the safe boundary δ and the disturbance upper bound $\nu(\delta)$ are straightforward from the results in [8].

2.4 Unperturbed Systems

The following theorem states that, if a system (11) is asymptotically stabilizable using a continuous time state feedback control law, then it is always possible to keep the state arbitrarily close to the equilibrium point by applying a digital self triggered strategy. Note that, in order to guarantee that the state is *arbitrarily* close to the equilibrium point, we need the stabilizability assumption.

Theorem 2 *Under Assumption 2, Problem 2 is solvable for system (11), and the function τ_s can be iteratively computed as a function of the current state of the system and the maximum allowable delay $\Delta_{\max}$.* ◇

Proof: Using the same reasoning of Theorem 1 proof, and directly considering the case $\Delta_k > 0$, we conclude that the following inequality

$$\dot{V} \leq -(1-\vartheta)\alpha_3(\|x\|) + \alpha_4(\|x\|)(\|d_h\| + \|d_\Delta\|) - \vartheta\alpha_3(\|x\|) \ \leq -(1-\vartheta)\alpha_3(\|x\|)$$

holds when

$$\alpha_4(\|x\|)\Big(M_1(x_k)(t-t_k) + M_2(x_k)(t-t_k)^2 + M_3\Delta_k\Big) \leq \vartheta\alpha_3(\|x\|)$$

with $\vartheta \in (0,1)$, and d_h, d_Δ, $M_1(x_k)$, $M_2(x_k)$, M_3 defined as in Theorem 1. The above inequality holds if

$$\|x\| \geq \eta \doteq \alpha_3^{-1}\Big(\frac{\alpha_4(\delta)}{\vartheta}\Big(M_1(x_k)(t-t_k) + M_2(x_k)(t-t_k)^2 + M_3\Delta_k\Big)\Big).$$

This implies, by [8], that there exists $b := \alpha_1^{-1}(\alpha_2(\eta)) > 0$ such that $\|x(\tau)\| \leq b$, $\forall \tau \in [t_k, t]$ if $x_k \in \mathcal{B}_b$ and if the following holds

$$\alpha_4(\delta)\Big(M_1(x_k)(t-t_k) + M_2(x_k)(t-t_k)^2 + M_3\Delta_k\Big) \leq \vartheta\alpha_3\Big(\alpha_2^{-1}(\alpha_1(\delta))\Big) \tag{25}$$

where we imposed the constraint $b = \delta$. Equation (25) holds if the following inequalities hold

$$\begin{aligned} &\alpha_4(\delta)\Big(M_1(x_k)(t-t_k) + M_2(x_k)(t-t_k)^2\Big) \leq \vartheta_1\alpha_3\Big(\alpha_2^{-1}(\alpha_1(\delta))\Big) \\ &\alpha_4(\delta)M_3\Delta_k \leq \vartheta_2\alpha_3\Big(\alpha_2^{-1}(\alpha_1(\delta))\Big) \end{aligned} \tag{26}$$

where we have set $\vartheta = \vartheta_1 + \vartheta_2$, with $\vartheta_1, \vartheta_2 \in (0,1)$ and $\vartheta_1 + \vartheta_2 < 1$. Defining

$$\begin{aligned} \Delta_{\max} &\doteq \vartheta_2 \frac{\alpha_3\Big(\alpha_2^{-1}(\alpha_1(\delta))\Big)}{\alpha_4(\delta)M_3} \\ \tau_s(x_k) &\doteq \max\{t - t_k : \ (26) \text{ is satisfied for each } t - t_k \in [0, \tau_s(x_k)]\} - \Delta_{\max} \\ \tau_{\min} &\doteq \min_{x_k \in \mathcal{B}_\delta} \tau_s(x_k) \end{aligned}$$

and if we choose $t_{k+1} = t_k + \tau_s(x_k)$, then (26) holds for all $t \in [t_k + \Delta_k, t_{k+1} + \Delta_{\max}]$ and for all $k \geq 0$, with $\Delta_{\max}$ non-negative. Since $M_1(x_k)$, $M_2(x_k)$ and M_3 are non–negative and upper bounded for each $x_k \in \mathcal{B}_\delta$, and since α_4, $\alpha_3 \circ \alpha_2^{-1} \circ \alpha_1 \in \mathcal{K}$, then the first of (26) is a second degree inequality in the form $ay^2 + by - c \leq 0$, where a, b are non–negative and bounded and c is strictly positive and bounded. Therefore, for ϑ_2 sufficiently small, $t_{k+1} - t_k = \tau_s(x_k) > \Delta_{\max} \geq 0$ for each $x_k \in \mathcal{B}_\delta$, and thus $\tau_{\min}$ is strictly positive as well. This completes the proof. ■

systems with a Lyapunov function fulfilling Assumption . Hence, condition (**??**) can be used to determine a time instant $\bar{t}_k > t_k$, while the application of Theorem 2 can be used to determine a value δ_k. Hence, the next sampling time results to be $t'_k = \bar{t}_k + \delta_k$, which may be bigger those determined by applying only Theorem 2.

2.5 Perturbed Systems

that there exists a nonempty set $\mathcal{U}$ of state feedback laws $\kappa: \mathcal{D}_x \to \mathcal{D}_u$, such that $\kappa \in C^\ell(\mathcal{D}_x)$ and the origin of the system (12) is asymptotically stable. ⋄A generic system (10), subject to disturbances and parameter variations, can be seen as the nominal system (11), perturbed by the term

$$d_g \doteq g(x, u, \mu, d) = f(x, u, \mu, d) - f_0(x, u) \doteq d_g. \tag{27}$$

Hence, (10) can be rewritten as follows

$$\dot{x} = f_0(x, u) + g(x, u, \mu, d). \tag{28}$$

Definition 1 *Under Assumption 2, and given the perturbed system (10) and a safe set $\mathcal{B}_\delta$, $\delta > 0$, we say that the perturbation (27) is δ–admissible if there exists a state feedback control law $\kappa \in \mathcal{U}$ and a constant $\vartheta_g \in (0, 1)$ such that the function $g(x, \kappa(x_0), \mu, d)$ satisfies*

$$\max_{\substack{x, x_k \in \mathcal{B}_\delta \\ d \in \mathcal{D}_d \\ \mu \in \mathcal{D}_\mu}} \|g(x, \kappa(x_k), \mu, d)\| \leq \nu(\delta) \doteq \vartheta_g \frac{\alpha_3\left(\alpha_2^{-1}(\alpha_1(\delta))\right)}{\alpha_4(\delta)} \tag{29}$$

with $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ as in (14). ⋄

The δ–admissible perturbations are those for which the safety problem with respect to a ball $\mathcal{B}_\delta$ can be solved using continuous time measurement and actuation, namely it is a necessary condition to achieve safety with respect to $\mathcal{B}_\delta$ using sampled measurements and actuations. Note that in condition (1) the expression of $\nu(\delta)$ can be explicitly computed.

The following theorem states that, if a system is asymptotically stabilizable using a continuous time state feedback control law and the perturbation is δ–admissible, then it is possible to keep the state in a boundary $\mathcal{B}_\delta$ of the equilibrium point by applying a digital self triggered strategy.

Theorem 3 *Under Assumption 2, Problem 2 is solvable for system (10) for any δ–admissible perturbation (27), and the function τ_s can be iteratively computed as a function of the current state of the system and the maximum allowable delay $\Delta_{\max}$.* ⋄

Proof: Using the same reasoning as in the proof of Theorem 2, and since the perturbation is assumed δ–admissible, we conclude that the following inequality

$$\begin{aligned}\dot{V} &\leq -(1-\vartheta)\alpha_3(\|x\|) - \vartheta\alpha_3(\|x\|) + \alpha_4(\|x\|)(\|d_h\| + \|d_\Delta\| + \|d_g\|) \\ &\leq -(1-\vartheta)\alpha_3(\|x\|)\end{aligned}$$

with d_g defined in (27), and d_h, d_Δ defined as in Theorem 1, holds when

$$\begin{aligned}&\alpha_4(\delta)\Big(M_1(x_k)(t-t_k) + M_2(x_k)(t-t_k)^2\Big) \leq \vartheta_1\alpha_3\Big(\alpha_2^{-1}(\alpha_1(\delta))\Big) \\ &\alpha_4(\delta)M_3\Delta_k \leq \vartheta_2\alpha_3\Big(\alpha_2^{-1}(\alpha_1(\delta))\Big)\end{aligned} \tag{30}$$

where $\vartheta = \vartheta_1 + \vartheta_2 + \vartheta_g$, with $\vartheta_1, \vartheta_2, \vartheta_g \in (0,1)$, and $\vartheta_1 + \vartheta_2 < 1 - \vartheta_g$, and $M_1(x_k)$, $M_2(x_k)$, M_3 are as in Theorem 1. Defining

$$\begin{aligned}\Delta_{\max} &\doteq \vartheta_2 \frac{\alpha_3\Big(\alpha_2^{-1}(\alpha_1(\delta))\Big)}{\alpha_4(\delta)M_3} \\ \tau_s(x_k) &\doteq \max\{t - t_k : \text{ (30) is satisfied for each } t - t_k \in [0, \tau_s(x_k)]\} - \Delta_{\max} \\ \tau_{\min} &\doteq \min_{x_k \in \mathcal{B}_\delta} \tau_s(x_k)\end{aligned}$$

and if we choose $t_{k+1} = t_k + \tau_s(x_k)$, then (30) holds for all $t \in [t_k + \Delta_k, t_{k+1} + \Delta_{\max}]$ and for all $k \geq 0$, with $\Delta_{\max}$ non-negative. Arguing as in the proof of Theorem 2, for ϑ_2 sufficiently small, $t_{k+1} - t_k = \tau_s(x_k) > \Delta_{\max} \geq 0$ for each $x_k \in \mathcal{B}_\delta$, and thus $\tau_{\min}$ is strictly positive as well. This completes the proof. ■

As discussed in Section 2.2, the choice of ϑ_1, ϑ_2 and ϑ_g corresponds to a simple tradeoff between larger intersampling times (ϑ_1), and robustness with respect to larger delays (ϑ_2) and perturbations (ϑ_g).

Remark 4 *Theorems 2 and 3 prove the existence of a self triggered strategy characterized by the time sequence $\mathcal{I} = \{t_k\}_{k\geq 0}$, with $t_k \geq \tau_{\min} > 0$ for each $k \geq 0$, such that the closed loop system satisfies a given safety specification. Moreover, they provide a formula to explicitly compute the next sampling time t_{k+1} as a function of the state x_k at time t_k.*

Although the simulation results, illustrated in Section 2.6, show strong benefits of the proposed self triggered strategy with respect to controllers based on constant sampling, the sequence $\mathcal{I}$ might be conservative, in the sense that longer sampling times might be determined, because of the approximations used in the proof. A trivial way to obtain a less conservative sequence $\mathcal{I}$ without introducing more restricting assumptions is the use of Taylor expansions of order higher than 2.

2.6 An Example of Application of the Digital Self Triggered Robust Control

Consider the system defined in Example 1. As already shown, we can not imply the existence of a stabilizing self triggered strategy. However, since Assumption 2 holds, Theorem 2 implies the existence of a self triggered strategy that guarantees safety for an arbitrary small neighborhood of the equilibrium point. In particular, since the origin of the system is locally exponentially stabilizable for $\|x\| \leq 2/3$, we define the safe set $\mathcal{B}_\delta$ with $\delta = 10^{-4} < 2/3$. We performed simulations using Matlab, with initial condition $x_0 = (10^{-5}, 10^{-5})^T \in \mathcal{B}_\delta$.

When a discrete time control law with constant sampling time greater than 2.1 s is used, the closed loop system is unstable.

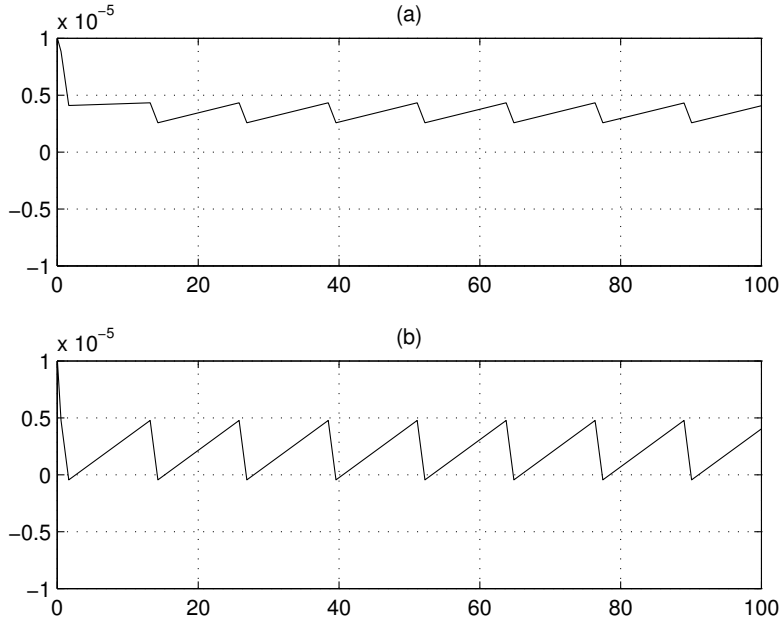

Figure 1: Self triggered control with $\vartheta_1 = 0.99$ and $\vartheta_2 = 0.009$: (a) x_1; (b) x_2 vs time

In Figure 1, the closed loop behavior is illustrated when the proposed self triggered control algorithm is used, with $\vartheta_1 = 0.99$ and $\vartheta_2 = 0.009$. The closed loop system is not asymptotically stable, but is safe with respect to $\mathcal{B}_\delta$ for the time interval $[t_0, \infty)$. It is interesting to remark that the average sampling time is

6.2 s, i.e. more than 295% longer than the constant sampling time of 2.1 s that yields an unstable control loop. Thus, using the proposed self triggered control algorithm, we achieve safety reducing of more than 295% the battery energy consumption, with respect to an unstable control strategy with constant sampling. However, since we have chosen $\vartheta_2 = 0.009$, we can only guarantee robustness with respect to delays bounded by $\Delta_{\max} = 0.17$ ms.

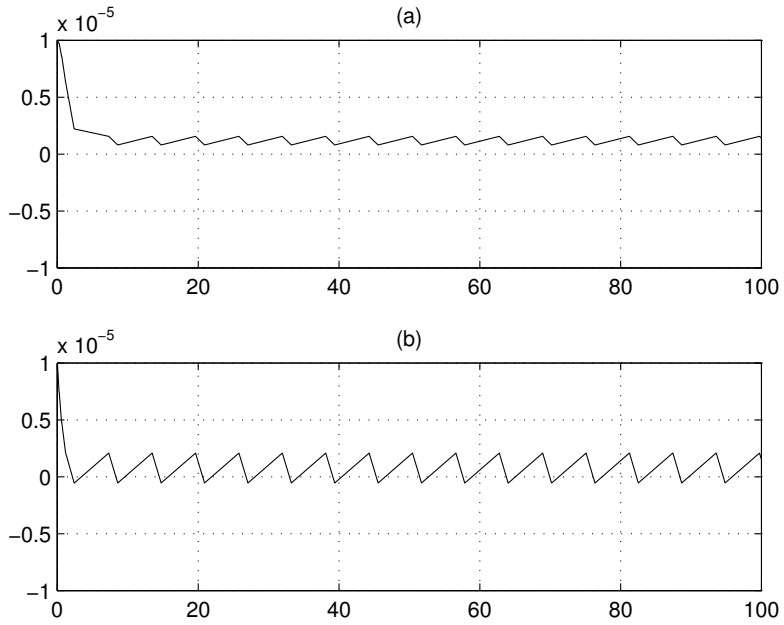

Figure 2: Self triggered control with $\vartheta_1 = 0.5$ and $\vartheta_2 = 0.499$: (a) x_1; (b) x_2 vs time

In Figure 2, the closed loop behavior is illustrated when the proposed self triggered control algorithm law is used, with $\vartheta_1 = 0.5$ and $\vartheta_2 = 0.499$. The closed loop system is not asymptotically stable, but is still safe with respect to $\mathcal{B}_\delta$ for the time interval $[t_0, \infty)$. However, since we have chosen $\vartheta_1 = 0.5$ in order to be robust with respect to delays, the average sampling time 3 s is more conservative with respect to the case $\vartheta_1 = 0.99$. Nevertheless, the average sampling time is almost 50% longer than the constant sampling time of 2.1 s, that yields an unstable control loop. Since we have chosen $\vartheta_2 = 0.009$, we can guarantee robustness with respect to delays bounded by $\Delta_{\max} = 9$ ms. Thus, using the proposed self triggered control algorithm, we achieve safety reducing of almost 50% the battery energy consumption, with respect to an unstable control strategy with constant sampling, while guaranteeing robustness with respect to delays bounded by $\Delta_{\max} = 9$ ms.

3 Self Triggered Robust Control of Nonlinear Stochastic Systems

In this section we consider the self–triggered stabilization problem for the class of stochastic systems, where the input generically enters both in the deterministic dynamics and in those affected by noise. This class of system is of particular interest since in practice various disturbances, not measurable, may affect the dynamics. We assume that the state equations are described by an Itô differential equation driven by a Wiener noise [24], [27], [28].

3.1 Problem Formulation

We consider nonlinear stochastic systems of the form

$$dx(t) = f_0(x,u)dt + \sum_{j=1}^{m} g_{0j}(x,u)d\xi_j(t) \tag{31}$$

where $x \in \mathcal{D}_x \subset r^n$, $\mathcal{D}_x$ is a domain containing the origin, $u \in \mathcal{D}_u \subset r^p$, f_0, $g_{0j} : \mathcal{D}_x \times \mathcal{D}\sqcap \to r^n$, $j = 1, \cdots, m$ are sufficiently smooth vector fields, such that $f_0(0,0) = 0$, $g_{0j}(0,0) = 0$, $j = 1, \cdots, m$. Moreover, $\{\xi(t) = (\xi_1(t), \cdots, \xi_m(t))^T, t \geq 0\}$ is a standard R^m–valued Wiener process, defined on the usual complete probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t\geq 0}, P)$, with $(\mathcal{F}_t)_{t\geq 0}$ the complete right–continuous filtration generated by ξ and $\mathcal{F}_0$ contains all P–null sets. It is worth stressing that in (31) the control appears either in the deterministic or in the stochastic terms [16], [17].

Given a continuous state feedback control law κ: $\mathcal{D}_x \to \mathcal{D}_u$, the closed loop system is

$$dx(t) = f_0(x,\kappa(x))\, dt + \sum_{j=1}^{m} g_{0j}(x,\kappa(x))\, d\xi_j(t) \tag{32}$$

and we will denote by $x(t)$, $t \geq t_0$, the solution of the closed loop system (32), with initial condition $x_0 = x(t_0)$. It is well–known that if the origin of system (32) is locally asymptotical stable in probability for a certain feedback κ, and if $f_0(x,\kappa(x)) \in C^\ell(\mathcal{D}_x)$, $\ell > 1$ integer, then there exists a Lyapunov function

$V(x)$ of class $C^2(\mathcal{D}_x)$ such that [25], [26]

$$\begin{aligned} \alpha_1(\|x\|) \leq V(x) \leq \alpha_2(\|x\|) \\ \mathcal{L}V(x) \leq -\alpha_3(\|x\|) \\ \left\| \frac{\partial V(x)}{\partial x} \right\| \leq \alpha_4(\|x\|) \\ \left\| \frac{\partial^2 V(x)}{\partial x^2} \right\| \leq \alpha_5(\|x\|) \end{aligned} \tag{33}$$

with $\alpha_i \in \mathcal{K}$, $i = 1, \cdots, 5$. The infinitesimal generator associated to (32), obtain by differentiating V in the sense of Itô, is given by

$$\mathcal{L}V(x) = \frac{\partial V(x)}{\partial x} f_0(x, \kappa(x)) + \frac{1}{2} \sum_{j=1}^{m} \mathrm{Tr}\Big(g_{0j}^T(x, \kappa(x)) \frac{\partial^2 V}{\partial x^2} \, g_{0j}(x, \kappa(x)) \Big). \tag{34}$$

Here, the matrix $\partial^2 V(t,x)/\partial x^2$ is the Hessian matrix of the second order partial derivatives, and $\mathrm{Tr}(\cdot)$ denotes the trace of a matrix.

Clearly, the feedback control signal $u(t) = \kappa(x(t))$ requires continuous measurements of the state of the system. In view of an implementation of $\kappa(x(t))$ by means of digital devices, with variable sampling intervals δ_k, in the following we consider its digital version

$$u(t) = \kappa(x_k), \qquad \forall t \in [t_k, t_{k+1} = t_k + \delta_k), \; k \geq 0 \tag{35}$$

one needs to determine these sampling instants t_k so that the stability property of the origin is preserved in probability. Following the approach developed in [14], [11], [4], [13], [5], [6], the aim is hence to determine on–line a sequence of strictly positive sampling intervals $\delta_k > 0$, i.e. a sequence $\{t_k\}_{k \geq 0}$ of sampling times, such that the origin of

$$dx(t) = f_0(x, \kappa(x_k))dt + \sum_{j=1}^{m} g_{0j}(x, \kappa(x_k)) d\xi_j(t) \tag{36}$$

is asymptotically stable in probability.

It is worth noting that to require $\delta_k > 0$ means that there exists a minimum sampling time $0 < \delta_k \leq \tau_{\min}$, $\forall \; k \geq 0$, which in turns will ensure that no Zeno behavior can occur. Hence, the time interval between two sampling instants is lower bounded by $\tau_{\min} > 0$.

The philosophy behind the self–triggered control is obvious: the control is performed only when necessary for guaranteeing the control objectives. This clearly reduces the transmission power of the sensing and actuation data transmissions, as well as the control effort of the digital device computing the control.

3.2 Self Triggered Stabilizing Control

The result developed in this section is based on the following assumption, analogous to the assumptions used in [14], [6] in the case of a deterministic systems.

Assumption 3 *Assume that*

1. $f_0, g_0 \in C^\ell(\mathcal{D}_x \times \mathcal{D}_u)$*, with* ℓ *a positive integer sufficiently large;*

2. *There exists a nonempty set* $\mathcal{U}$ *of state feedback laws* $\kappa\colon \mathcal{D}_x \to \mathcal{D}_u$*, such that* $\kappa \in C^\ell(\mathcal{D}_x)$ *and the origin of (32) is asymptotically stable in probability, with region of attraction a certain compact* $\Omega \subset \mathcal{D}_x$*;*

3. *The functions* $\alpha_3, \alpha_4, \alpha_5 \in \mathcal{K}$ *in (33) are such that* $\alpha_3^{-1}, \alpha_4, \alpha_5$ *are Lipschitz.* $\diamond$

The assumption of sufficient regularity of the functions f_0, g_0 is required in order to ensure the determination of the next sampling time, making use of a Taylor expansion, analogous to that used in [6]. The assumption of existence of a stabilizing control is not restrictive, since if the nominal system cannot be stabilized in probability using continuous time measurements and actuations, then it is clear that the nominal system cannot be stabilized using a digital control with zero–order holders. Finally, the Lipschitz assumption on $\alpha_3^{-1}, \alpha_4, \alpha_5$ is required to write a simple stability condition, as used in [14], and represents the main limitation of this approach.

Using Assumption 3, one can state the following result.

Theorem 4 *Let us consider the nonlinear stochastic system (31). Under Assumption 3, there exist a piece–wise constant state feedback control law (35), and a sequence of strictly positive sampling intervals* $\delta_k > 0$*, such that the origin of the closed loop system (36) is asymptotically stable in probability.* $\diamond$

Proof: Since $\mathcal{U}$ is not empty, by Assumption 3, we pick a state feedback control law $\kappa \in \mathcal{U}$. Since $f_0(x, \kappa(x)) \in C^\ell(\mathcal{D}_x)$ with $\ell > 1$, there exists a Lyapunov candidate (33). Let us choose $r > 0$ such that the ball $B_r = \{x \in \Omega \mid \|x\| \leq r\} \subset \Omega$.

For $x_k \in B_r$, the infinitesimal generator associated to (36) is given by

$$\mathcal{L}V(x_k) = \frac{\partial V}{\partial x} f_0(x, \kappa(x_k)) + \frac{1}{2} \sum_{j=1}^{m} \mathrm{Tr}\Big(g_{0j}^T(x, \kappa(x_k)) \, \frac{\partial^2 V}{\partial x^2} \, g_{0j}(x, \kappa(x_k)) \Big)$$

which can be rewritten as

$$\begin{aligned}\mathcal{L}V(x_k) = \frac{\partial V}{\partial x} f_0(x,\kappa(x)) + \frac{\partial V}{\partial x}\Big(f_0(x,\kappa(x_k)) - f_0(x,\kappa(x))\Big) + \frac{1}{2}\sum_{j=1}^{m} \mathrm{Tr}\Big(g_{0j}^T(x,\kappa(x))\,\frac{\partial^2 V}{\partial x^2}\,g_{0j}(x,\kappa(x))\Big) \\ + \frac{1}{2}\sum_{j=1}^{m} \mathrm{Tr}\Big(\Big[g_{0j}(x,\kappa(x_k)) - g_{0j}(x,\kappa(x))\Big]^T \times \frac{\partial^2 V}{\partial x^2}\Big[g_{0j}(x,\kappa(x_k)) - g_{0j}(x,\kappa(x))\Big]\Big) \\ + \sum_{j=1}^{m} \mathrm{Tr}\Big(g_{0j}^T(x,\kappa(x_k))\,\frac{\partial^2 V}{\partial x^2}\,g_{0j}(x,\kappa(x))\Big) - \sum_{j=1}^{m} \mathrm{Tr}\Big(g_{0j}^T(x,\kappa(x))\,\frac{\partial^2 V}{\partial x^2}\,g_{0j}(x,\kappa(x))\Big).\end{aligned} \tag{37}$$

Note that

$$\begin{aligned}&\frac{1}{2}\mathrm{Tr}\Big(\Big[g_{0j}(x,\kappa(x_k)) - g_{0j}(x,\kappa(x))\Big]^T \times \frac{\partial^2 V}{\partial x^2}[g_{0j}(x,\kappa(x_k)) - g_{0j}(x,\kappa(x))]\Big) \\ &\quad\le \frac{n}{2}\Big\|\Big[g_{0j}(x,\kappa(x_k)) - g_{0j}(x,\kappa(x))\Big]^T \times \frac{\partial^2 V}{\partial x^2}\Big[g_{0j}(x,\kappa(x_k)) - g_{0j}(x,\kappa(x))\Big]\Big\|_\infty \\ &\quad\le \frac{n\sqrt{n}}{2}\Big\|\Big[g_{0j}(x,\kappa(x_k)) - g_{0j}(x,\kappa(x))\Big]^T \times \frac{\partial^2 V}{\partial x^2}\Big[g_{0j}(x,\kappa(x_k)) - g_{0j}(x,\kappa(x))\Big]\Big\| \\ &\quad\le \frac{n\sqrt{n}}{2}\Big\|g_{0j}(x,\kappa(x_k)) - g_{0j}(x,\kappa(x))\Big\|^2\Big\|\frac{\partial^2 V}{\partial x^2}\Big\|\end{aligned}$$

for all $j = 1,\cdots,m$. Similarly, the sum of the two last terms of (37) are such that

$$\begin{aligned}&\mathrm{Tr}\Big(\Big[g_{0j}(x,\kappa(x_k)) - g_{0j}(x,\kappa(x))\Big]^T \frac{\partial^2 V}{\partial x^2} g_{0j}(x,\kappa(x))\Big] \\ &\qquad\le n\Big\|\Big[g_{0j}(x,\kappa(x_k)) - g_{0j}(x,\kappa(x))\Big]^T \frac{\partial^2 V}{\partial x^2} g_{0j}(x,\kappa(x))\Big\|_\infty \\ &\qquad\le n\sqrt{n}\Big\|\Big[g_{0j}(x,\kappa(x_k)) - g_{0j}(x,\kappa(x))\Big]^T \times \frac{\partial^2 V}{\partial x^2} g_{0j}(x,\kappa(x))\Big\| \\ &\qquad\le n\sqrt{n}\Big\|g_{0j}(x,\kappa(x_k)) - g_{0j}(x,\kappa(x))\Big\| \times \Big\|g_{0j}(x,\kappa(x))\Big\|\Big\|\frac{\partial^2 V}{\partial x^2}\Big\|.\end{aligned}$$

Using these bounds, one obtains

$$\begin{aligned}\mathcal{L}V(x_k) \le -\alpha_3(\|x\|) + \alpha_4(\|x\|)\|d_{h,f}\| + \frac{n\sqrt{n}}{2}\alpha_5(\|x\|)\sum_{j=1}^{m}\|d_{h,g_j}\|^2 \\ + n\sqrt{n}\alpha_5(\|x\|)\sum_{j=1}^{m}\|d_{h,g_j}\|\|g_{0j}(x,\kappa(x))\|\end{aligned} \tag{38}$$

where

$$d_{h,f} = f_0(x(t),\kappa(x(t_k))) - f_0(x(t),\kappa(x(t)))$$

$$d_{h,g_j} = g_{0j}(x(t),\kappa(x(t_k))) - g_{0j}(x(t),\kappa(x(t)))$$

$j = 1,\cdots,m$, are terms that can be regarded as perturbations, due to the holding, and acting on the control system (32).

Under Assumption 3, there exists a time interval $[t_k, t_k + \varepsilon_k]$ such that (36) has a unique solution $x(t)$. Hence, it is possible to expand in Taylor series the i^{th} components $d_{h,f,i}, d_{h,g_j,i}$ of $d_{h,f}, d_{h,g_j}$, $j = 1, \cdots, m$, with respect to $t \in [t_k, t_k + \varepsilon_k]$, on the right of t_k, with the Lagrange remainder. Denoting by $\mathrm{d}^n_+(\cdot)/\mathrm{d}t^n$ the n–th right derivative, and proceeding as in [6], one works out

$$\begin{aligned} d_{h,f,i} &= \varphi_{1,i}(x_k)(t - t_k) + \varphi_{2,i}(\bar{x}_i, x_k)(t - t_k)^2 \\ d_{h,g_j,i} &= \varphi_{3j,i}(x_k)(t - t_k) + \varphi_{4j,i}(\bar{x}_i, x_k)(t - t_k)^2 \end{aligned} \tag{39}$$

for $j = 1, \cdots, m$, where we have defined

$$\begin{aligned} \varphi_{1,i}(x_k) &= d_{h,f,i}\Big|_{x(t)=x_k} \\ \varphi_{2,i}(\bar{x}_i, x_k) &= \frac{1}{2} \left.\frac{\mathrm{d}^2_+ d_{h,f,i}}{\mathrm{d}t^2}\right|_{x(t)=\bar{x}_i} \\ \varphi_{3j,i}(x_k) &= d_{h,g_j,i}\Big|_{x(t)=x_k} \\ \varphi_{4j,i}(\bar{x}_i, x_k) &= \frac{1}{2} \left.\frac{\mathrm{d}^2_+ d_{h,g_j,i}}{\mathrm{d}t^2}\right|_{x(t)=\bar{x}_i} . \end{aligned}$$

The Taylor theorem with the Lagrange remainder ensures the existence of time instants $\bar{t}_i \in [t_k, t]$, with $\bar{x}_i = x(\bar{t}_i)$, $i = 1, \cdots, n$, such that the equalities (39) hold. Denoting by $\bar{x} = (\bar{x}_1, \cdots, \bar{x}_n)$ the corresponding point, one obtains

$$\begin{aligned} \|d_{h,f}\| &\leq \|\varphi_1(x_k)\|(t - t_k) + \|\varphi_2(\bar{x}, x_k)\|(t - t_k)^2 \\ \|d_{h,g_j}\| &\leq \|\varphi_{3j}(x_k)\|(t - t_k) + \|\varphi_{4j}(\bar{x}, x_k)\|(t - t_k)^2 \end{aligned} \tag{40}$$

where

$$\begin{aligned} \varphi_p(x_k) &= \Big(\varphi_{p,1}(x_k), \cdots, \varphi_{p,n}(x_k)\Big)^T \\ \varphi_q(\bar{x}, x_k) &= \Big(\varphi_{q,1}(\bar{x}_1, x_k), \cdots, \varphi_{q,n}(\bar{x}_n, x_k)\Big)^T \end{aligned}$$

for $p = 1, 3j$ and $q = 2, 4j$, $j = 1, \cdots, m$. Moreover, let us consider the level set $\Omega_{V(x_k)}$, and define

$$\begin{aligned} M_p(x_k) &= \|\varphi_p(x_k)\|, && p = 1, 3j \\ M_q(x_k) &= \max_{\bar{x} \in \Omega_{V(x_k)}} \|\varphi_q(\bar{x}, x_k)\|, && q = 2, 4j. \end{aligned}$$

Since $f_0, g_{0j}, \kappa \in C^\ell$ and $\Omega_{V(x_k)}$ is compact, then $M_p(x_k)$ is finite for any $x_k \in \Omega_{V(x_k)}$, and $M_q(x_k) \in r^+$ exists and is finite for any $x_k \in \Omega_{V(x_k)}$.

Finally, one can introduce the terms

$$C_{g_{0j}} = \max_{x \in \Omega_{V(x_k)}} \Big\|g_{0j}(x, \kappa(x))\Big\|, \quad j = 1, \cdots, m$$

which exist and are finite on the compact set $\Omega_{V(x_k)}$, so that the infinitesimal generator (38) associated to (36) can be written as follows

$$\begin{aligned}\mathcal{L}V(x_k) &\leq -\alpha_3(\|x\|) + \alpha_4(\|x\|)\|d_{h,f}\| + \frac{n\sqrt{n}}{2}\alpha_5(\|x\|)\sum_{j=1}^{m}\left(\|d_{h,g_j}\| + 2C_{g_{0j}}\|d_{h,g_j}\|\right)\\ &= -\alpha_3(\|x\|) + \alpha_4(\|x\|)\|d_{h,f}\| + \frac{n\sqrt{n}}{2}\alpha_5(\|x\|)\sum_{j=1}^{m}\Big(\left(\|d_{h,g_j}\| + C_{g_{0j}}\right)^2 - C_{g_{0j}}^2\Big).\end{aligned} \tag{41}$$

In what follows we will show that, for each term in (41) which is not negative definite, one can consider a negative definite term that ensures the negativity of the whole $\mathcal{L}V$, at least for small (but nonzero) time intervals. For, let us consider $\vartheta = \sum_{j=0}^{m}\vartheta_j < 1$, with $\vartheta_j \in (0,1)$, $j = 0, \cdots, m$, and let us require that

$$\begin{aligned}\alpha_4(\|x\|)\|d_{h,f}\| &\leq \vartheta_0\alpha_3(\|x\|)\\ \frac{n\sqrt{n}}{2}\alpha_5(\|x\|)\Big(\left(\|d_{h,g_j}\| + C_{g_{0j}}\right)^2 - C_{g_{0j}}^2\Big) &\leq \vartheta_j\alpha_3(\|x\|)\\ j &= 1, \cdots, m\end{aligned} \tag{42}$$

are satisfied. Conditions (42) will determine a time instant $t_{k+1} = t_k + \delta_k < t_k + \varepsilon_k$, and hence a positive time interval δ_k in which the infinitesimal generator is negative definite. In fact, using (40), equations (42) are satisfied if

$$\begin{aligned}\alpha_3^{-1}\Big(\frac{1}{\vartheta_0}\alpha_4(\|x\|)\Big(M_1(x_k)(t-t_k) + M_2(x_k)(t-t_k)^2\Big)\Big) &\leq \|x\|\\ \alpha_3^{-1}\Big(\frac{n\sqrt{n}}{2\vartheta_j}\alpha_5(\|x\|)\Big[\left(M_{3j}(x_k)(t-t_k) + M_{4j}(x_k)(t-t_k)^2 + C_{g_{0j}}\right)^2 - C_{g_{0j}}^2\Big]\Big) &\leq \|x\|\end{aligned}$$

for $j = 1, \cdots, m$. Since α_3^{-1}, α_4 and α_5 are Lipschitz, then equations (42) are satisfied if

$$\begin{aligned}\frac{1}{\vartheta_0}L_{\alpha_3^{-1}}L_{\alpha_4}\|x\|\Big(M_1(x_k)(t-t_k) + M_2(x_k)(t-t_k)^2\Big) &\leq \|x\|\\ \frac{n\sqrt{n}}{2\vartheta_j}L_{\alpha_3^{-1}}L_{\alpha_5}\|x\|\Big[\Big(M_{3j}(x_k)(t-t_k) + M_{4j}(x_k)(t-t_k)^2 + C_{g_{0j}}\Big)^2 - C_{g_{0j}}^2\Big] &\leq \|x\|\end{aligned}$$

for all $j = 1, \cdots, m$, where $L_{\alpha_3^{-1}}, L_{\alpha_4}, L_{\alpha_5} > 0$ are the Lipschitz constants of $\alpha_3^{-1}, \alpha_4, \alpha_5$, respectively. These equations imply that (42) are satisfied under the sufficient conditions

$$\begin{aligned}M_1(x_k)(t-t_k) + M_2(x_k)(t-t_k)^2 &\leq \frac{\vartheta_0}{L_{\alpha_3^{-1}}L_{\alpha_4}}\\ M_{3j}(x_k)(t-t_k) + M_{4j}(x_k)(t-t_k)^2 &\leq \left(\sqrt{1 + \frac{2\vartheta_j}{n\sqrt{n}}\frac{1}{C_{g_{0j}}^2 L_{\alpha_3^{-1}}L_{\alpha_5}}} - 1\right)C_{g_{0j}}\end{aligned} \tag{43}$$

for $j = 1, \cdots, m$. Defining

$$\delta_k = \min\max\left\{t - t_k \mid \text{(43) are satisfied, } j = 1, \cdots, m\right\}$$

and choosing $t_{k+1} = t_k + \delta_k$, then

$$\mathcal{L}V(x) \leq -(1-\vartheta)\alpha_3(\|x\|)$$

for all $t \in [t_k, t_{k+1}]$ and for all $k \geq 0$. This implies that the origin is asymptotically stable in probability.

Equations (43) are second degree inequalities in the form $a(x_k)y^2 + b(x_k)y \leq c$, where $a(x_k), b(x_k)$ are non-negative and upper bounded for each $x_k \in \mathcal{D}_x$, and c is strictly positive and upper bounded. This trivially implies that δ_k, $\forall\, k \geq 0$, are strictly positive for each $x_k \in \Omega_{V(x_k)}$, and thus a minimum dwell time does exists, so ensuring that δ_k does not go to zero as $k \to \infty$. ■

Remark 5 *From the proof of the previous result, it is clear that the main difference between the deterministic and the stochastic case consists of the fact that the sampling period has to satisfy extra conditions [6]. In fact, while in the deterministic case one can determine a sampling sequence $\{\delta_k\}$ solving only the first of conditions (43), in the stochastic case one needs to satisfy m additional conditions given by the second of (43). Therefore, in the stochastic case the self–triggered control strategy will determine, in general, more restrictive (shorter) sampling times.* ◇

3.3 Self Triggered Safety Control

The main limitation of the results developed in Section 3.2 is the Lipschitz continuity assumption of $\alpha_3^{-1}(\cdot)$. In fact, if $\alpha_3^{-1}(\cdot)$ is not Lipschitz, the next sampling time $t_k + \delta_k$ goes to zero as x_k approaches the equilibrium point, and this might generate Zeno behaviors. Hence, in the spirit of the self triggered safety control addressed in [6], in this section we will show that it is possible to keep the state arbitrarily close to the equilibrium point by applying a self triggering strategy. The solution of this problem will not require the Lipschitz assumption on α_3^{-1}.

In the following definition, an invariant property is used to define that a system is almost surely (a.s.) safe with respect to a given subset of the state space.

Definition 2 *Given a state feedback control law κ, system (32) is a.s. safe with respect to the set $\mathcal{S} \subseteq \mathcal{D}_x$ for the time interval $\mathcal{T} \subseteq r^+$, if $x(t) \in \mathcal{S}$, $\forall t \in \mathcal{T}$ a.s.* ◇

Given the system (31), a stabilizing state feedback control law κ, and an arbitrary safe set $\mathcal{B}_\delta = \{x \in r^n \mid \|x\| < \delta\} \subset \mathcal{D}_x$, the objective is to determine a sequence of strictly positive sampling intervals $\delta_k > 0$ and a piece–wise constant state feedback control law, as in the previous section, such that the closed loop system (32) is a. s. safe with respect to $\mathcal{B}_\delta$, for the time interval $[t_0, \infty)$. ◇

The results developed in this section are based on the following.

Assumption 4 *Assume that $f_0, g_{0j} \in C^\ell(\mathcal{D}_x \times \mathcal{D}_u)$, $j = 1, \cdots, m$, with ℓ a positive integer sufficiently large. Assume that there exists a nonempty set $\mathcal{U}$ of state feedback laws $\kappa: \mathcal{D}_x \to \mathcal{D}_u$, such that $\kappa \in C^\ell(\mathcal{D}_x)$ and the origin of the system (32) is asymptotically stable in probability.* ⋄

The following theorem states that if a system is almost surely asymptotically stabilizable, using a continuous time state feedback control law, then it is always possible to keep the state arbitrarily close to the equilibrium point by applying a digital self triggering strategy. Note that, in order to guarantee that the state is arbitrarily close to the equilibrium point, we still need the stabilizability assumption.

Theorem 5 *Given the system (31) and a safe set $\mathcal{B}_\delta$, $\delta > 0$, under Assumption 4 there exist piece–wise constant state feedback control law (35) and a sequence of strictly positive sampling intervals $\delta_k > 0$ such that the closed loop system (32), (35) is almost surely safe with respect to $\mathcal{B}_\delta$, for the time interval $[t_0, \infty)$.* ⋄

Proof: The proof of Theorem (5) follows the same arguments of Theorem (4).

3.4 A Simple Illustrative Example

In order to illustrate the propose approach, let us consider the following system

$$\begin{aligned} dx &= (Ax + Bu + f(x,u))dt + Cxdw \\ &= f_0(x,u)dt + g_0(x,u)dw \end{aligned}$$

with

$$f_0(x,u) = \begin{pmatrix} -x_1 + x_2 + x_1^2 \\ (1 + x_1)u \end{pmatrix}, \qquad C = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}.$$

Let us consider the continuous control $u = \kappa(x) = -x_2 \in \mathcal{U}$, and the Lyapunov candidate $V(x) = x^T P x$, with P solution of the Lyapunov equation

$$PA_c + A_c^T P + C^T P C = -R$$

with R a symmetric positive definite matrix, and

$$A_c = \begin{pmatrix} -1 & 1 \\ 0 & -1 \end{pmatrix}.$$

If

$$R = \begin{pmatrix} -2 & 0 \\ 0 & -3 \end{pmatrix}$$

the matrix

$$P = \begin{pmatrix} 3 & 1 \\ 1 & 4 \end{pmatrix}$$

is a solution of the Lyapunov equation, with $\lambda^P_{\min} \cong 2.382$, $\lambda^P_{\max} \cong 4.618$ the minimum and the maximum eigenvalue of P, respectively.

The infinitesimal generator associated to the previous system, for $\|x\| \leq 1/3$ satisfies

$$\begin{aligned} \mathcal{L}V &\leq -2x_1^2 - 3x_2^2 + 6|x_1|^3 + 8|x_1|x_2^2 \\ &\leq -2x_1^2 - 3x_2^2 + 6(1/3)x_1^2 + 8(1/3)x_2^2 \leq -\frac{1}{3}\|x\|^2. \end{aligned}$$

Thus, the origin of the system is almost surely exponentially stable in probability, with

$$\alpha_1 = \lambda^P_{\min}\|x\|^2, \qquad \alpha_2 = \lambda^P_{\max}\|x\|^2, \quad \alpha_3 = \|x\|^2/3$$

$$\alpha_4 = \lambda^P_{\max}\|x\|, \qquad \alpha_5 = \|2P\|.$$

It is clear that Assumption 3 is not satisfied, since α_3^{-1} is not Lipschitz. For this reason, we can not imply the existence of a stabilizing self triggered strategy. However, Theorem 5 implies the existence of a self triggered strategy that guarantees almost surely safety for an arbitrary small neighborhood of the equilibrium point. Since the origin is locally stabilizable in probability for $\|x\| \leq 1/3$, we can define the safe set as the ball $\mathcal{B}_\delta$ with $\delta = 10^{-4} < 1/3$.

4 Implementation Features about Control Algorithms in Digital Logic Devices

In the digital logic system scenario, there are many architectures suitable for realizing different kinds of control algorithms considering performances in terms of timing, power consumption and resources availability.

In this section a brief introduction about microprocessors and custom devices for digital processing is presented, in order to obtain an evaluation about benefits and drawbacks. DSP processors are described in their features and logic functioning, focusing on how they tend to be integrated in wider systems including FPGAs, memories, peripherals, etc. Moreover, FPGA technology is depicted in the control environment explaining how designers are trying to exploit their potentialities in applications of parallel computation, control and digital signal processing.

4.1 General Description of DSP Systems

The informal definition of digital signal processing is the application of mathematical operations to digitally represent and elaborate signals. Often, samples are obtained from physical signals (for example, audio signals) through the use of transducers (such as microphones) and analog-to-digital converters. After mathematical processing, digital signals may be converted back to physical signals via digital-to-analog converters. In some systems, the use of DSP is crucial for the operation of the system. For example, modems and digital cellular telephones rely very heavily on DSP technology. In other products, the use of DSP is less central, but often offers important competitive advantages in terms of features, performance, and costs. For example, manufacturers of primarily analog consumer electronics devices like audio amplifiers are beginning to employ DSP technology to provide new features.

4.1.1 DSP Advantages

Digital signal processing enjoys several advantages over analog signal processing. The most significant of these is that DSP systems are able to accomplish tasks inexpensively that would be difficult or even impossible using analog electronics. Examples of such applications include speech synthesis, speech recognition, and high-speed control techniques. DSP systems also enjoy two additional advantages over

analog systems:

1. Low sensitivity to the environment. Digital systems, by their nature, are considerably less sensitive to environmental conditions than analog systems. For example, an analog circuit's behavior depends strictly on its temperature. In contrast, barring catastrophic failures, a DSP system's delivers the same response for little temperature changes.

2. Insensitivity to component tolerances. Analog components are manufactured to particular tolerances- a resistor, i.e., might be guaranteed to have a resistance within 1 percent of its nominal value. The overall response of an analog system depends on the actual values of all of the analog components used. As a result, two analog systems of exactly the same design will have slightly different responses due to slight variations in their components. In contrast, correctly functioning digital components always produce the same outputs given the same inputs.

These two advantages combine synergistically to give DSP systems an additional benefit over analog systems:

3. Predictable, repeatable behavior. Because a DSP system's output does not vary due to environmental factors or component variations, it is possible to design systems having exact, known responses.

Finally, some DSP systems may also have two other advantages over analog systems:

4. Reprogrammability. If a DSP system is based on programmable processors or programmable logic devices (PLD) in general, it can be reprogrammed, even in the field, to perform other tasks. In contrast, analog systems require physically different components to perform different tasks.

5. Size. The size of analog components varies related to their values.

These advantages, coupled with the fact that DSP can take advantage of the rapidly increasing density of digital integrated circuit manufacturing processes, increasingly make DSP the solution of choice for signal processing.

4.1.2 DSP Systems Features

In this section a number of characteristics common to all DSP systems are described, such as algorithms, sample rate, clock rate, and arithmetic types.

1. Algorithms

   DSP systems are often characterized by algorithms. The algorithm specifies the arithmetic operations to be performed but does not specify how that arithmetic is to be implemented. It might be implemented in software on an ordinary microprocessor or in a programmable signal processor, or it might be implemented in custom integrated circuits. The selection of an implementation technology is determined in part by the required speed and arithmetic precision.

2. Sample Rates

   A key characteristic of a DSP system is its sample rate: the rate at which samples are consumed, processed, or produced. Combined with the complexity of the algorithms, the sample rate determines the required speed of the implementation technology. A familiar example is the digital audio compact disc (CD) player, which produces samples at a rate of 44.1 kHz on two channels. Of course, a DSP system may use more than one sample rate; such systems are said to be multirate DSP systems. An example is a converter from the CD sample rate of 44.1 kHz to the digital audio tape (DAT) rate of 48 kHz. Because of the awkward ratio between these sample rates, the conversion is usually done in stages, typically with at least two intermediate sample rates. Another example of a multirate algorithm is a filter bank, used in applications such as speech, audio, and video encoding and some signal analysis algorithms. Filter banks typically consist of stages that divide the signal into high and low frequency portions. These new signals are then downsampled (i.e., their sample rate is lowered by periodically discarding samples) and divided again. In multirate applications, the ratio between the highest and the lowest sample rates in the system can become quite large, sometimes exceeding 100,000. The range of sample rates encountered in signal processing systems is huge. Sample rates for applications range over 12 orders of magnitude. Only at the very top of that range is digital implementation rare. This is because the cost and difficulty of implementing a given algorithm digitally increases with the sample rate. DSP algorithms used at higher sample rates tend obviously to be simpler than those used at lower sample rates. Many DSP systems must meet extremely rigorous speed goals, since they operate on lengthy segments of real world signals in real–time. Where other kinds of systems (like databases) may be required to meet performance goals on average, real–time DSP systems often must meet such goals in every instance. In such systems, failure to maintain the necessary processing rates is considered a serious malfunction. Such systems are often said to be subject to hard realtime constraints.

3. Clock Rates

   Digital electronic systems are often characterized by their clock rates. The clock rate usually refers to the rate at which the system performs its most basic unit of work. In mass-produced, commercial products, clock rates of up to 100 MHz are common, with faster rates found in some high–performance products. For DSP systems, the ratio of system clock rate to sample rate is one of the most important characteristics used to determine how the system will be implemented. The relationship between the clock rate and the sample rate partially determines the amount of hardware needed to implement an algorithm with a given complexity in real–time. As the ratio of sample rate to clock rate increases, so does the amount and complexity of hardware required to implement the algorithm.

4. Numeric Representations

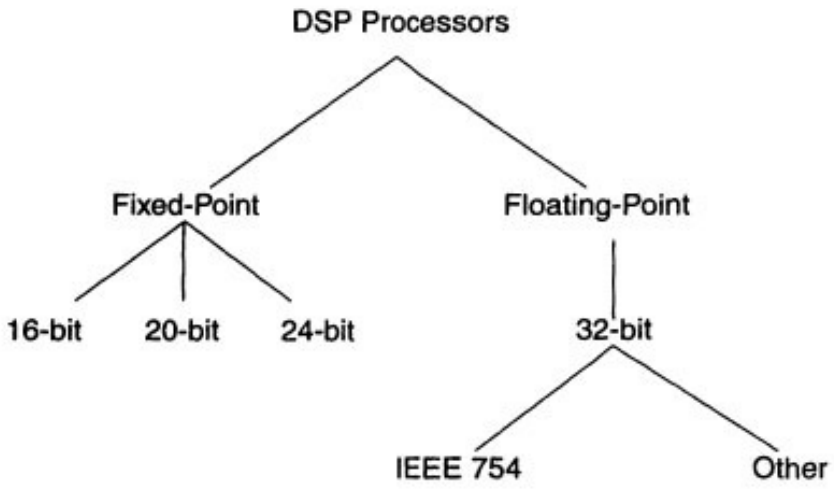

Figure 3: Numerical representation in DSP processors

Arithmetic operations such as addition and multiplication are at the heart of DSP algorithms and systems [39]. As a result, the numeric representations and type of arithmetic used can have a profound influence on the behavior and performance of a DSP system. The most important choice for the designer is between fixed–point and floating–point arithmetic. Fixed-point arithmetic represents numbers in a fixed range (e.g., -1.0 to $+1.0$) with a finite number of bits of precision (called the word width). For example, an eight-bit fixed–point number provides a resolution of 1/256 of the range over which the

number is allowed to vary. Numbers outside of the specified range cannot be represented; arithmetic operations that would result in a number outside this range either saturate (that is, are limited to the largest positive or negative representable value) or wrap around (that is, the extra bits resulting from the arithmetic operation are discarded).

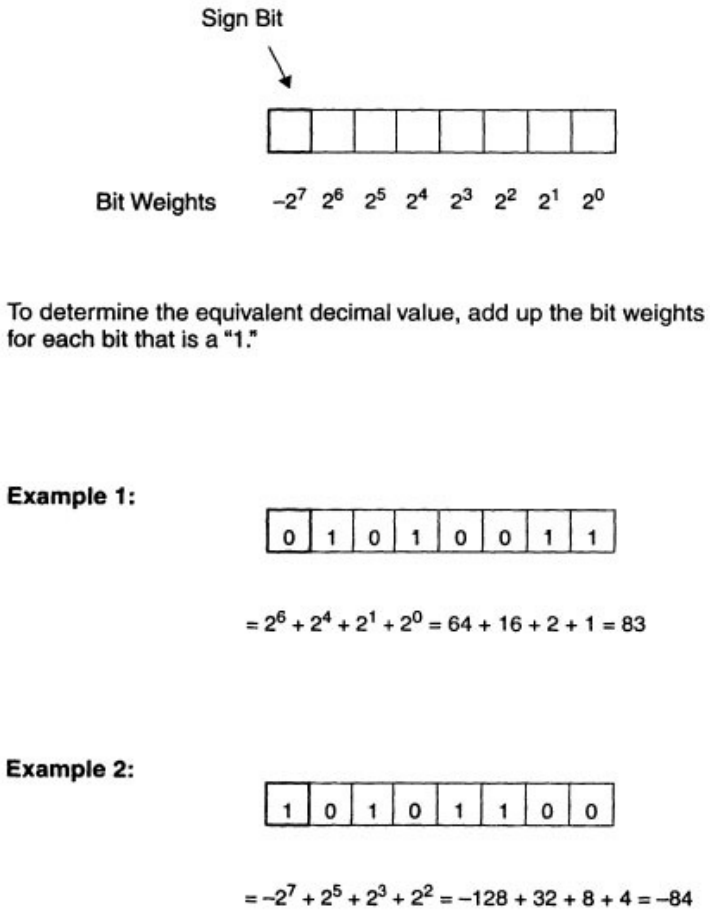

Figure 4: Simple binary integer representation

Floating–point arithmetic greatly expands the representable range of values. Floating–point arithmetic represents every number in two parts: a mantissa and an exponent. The mantissa is, in effect, forced to lie between -1.0 and $+1.0$, while the exponent keeps track of the amount by which the mantissa must be scaled (in terms of powers of two) in order to create the actual value represented.

That is: value = mantissa $\times$ base$^{\text{exponent}}$. Floating–point arithmetic provides much greater dynamic range (that is, the ratio between the largest and smallest value that can be represented) than fixed–point arithmetic. Because it reduces the probability of overflow and the necessity of scaling, it can considerably simplify algorithm and software design. Unfortunately, floating–point arithmetic is generally slower and more expensive than fixed–point arithmetic, and is more complicated to implement in hardware than

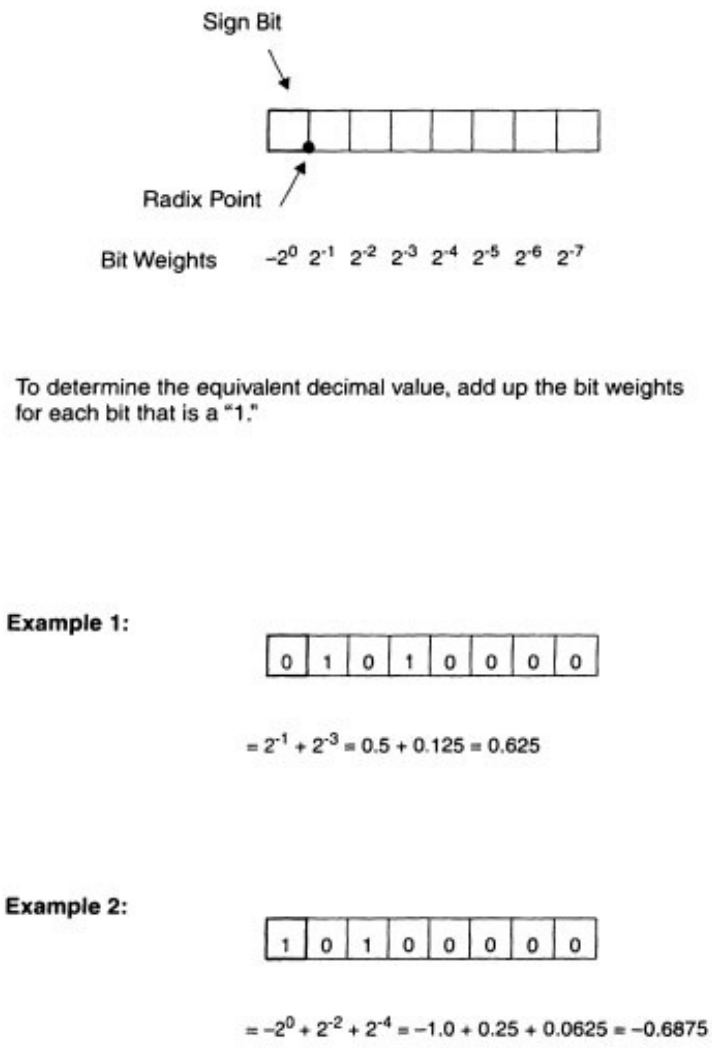

Figure 5: Simple binary fractional representation

fixed point arithmetic.

4.1.2.1 Fixed–Point Versus Floating–Point

The earliest DSP processors used fixed–point arithmetic, and in fact fixed–point DSPs still dominate today. The algorithms and hardware used to implement fractional arithmetic are virtually identical to those used for integer arithmetic. The main difference between integer and fractional arithmetic has to do with how the results of multiplication operations are handled. In practice, most fixed–point DSP processors support fractional arithmetic and integer arithmetic. The former is most useful for signal processing algorithms, while the latter is useful for control operations, address calculations, and other

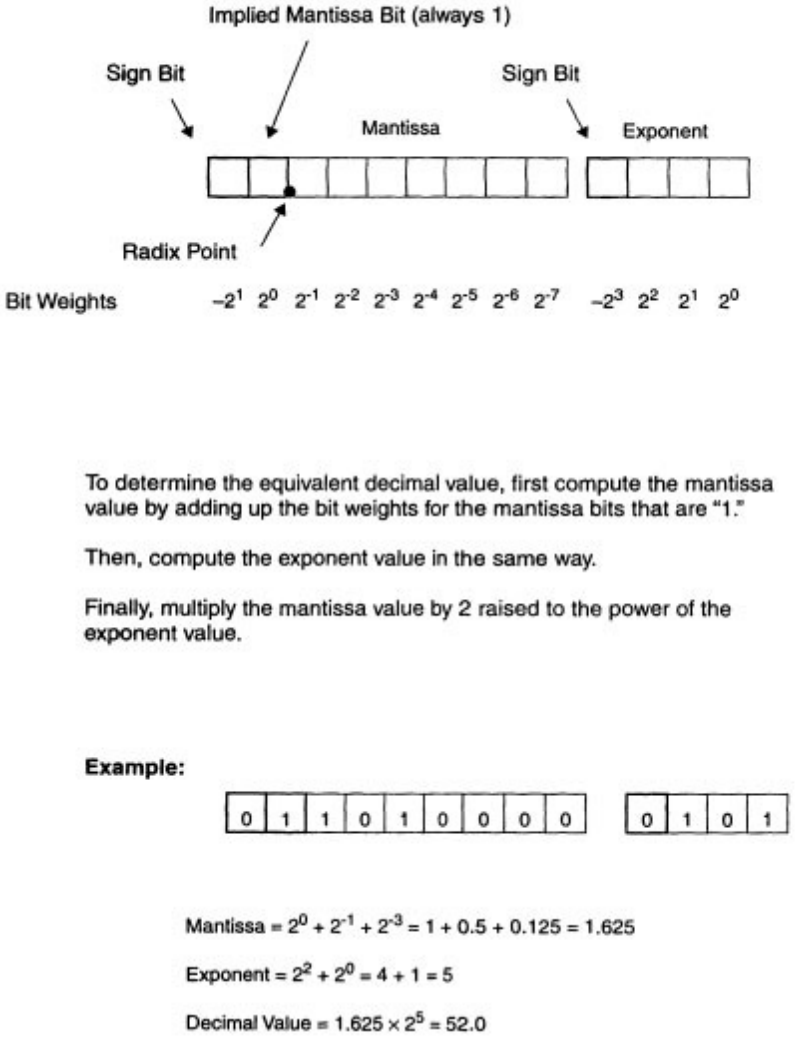

Figure 6: Simplified binary floating–point representation, comprised of a mantissa (fraction part) and an exponent

operations that do not involve signals. With the floating–point representation instead, system designers have access to wider dynamic range (the ratio between the largest and smallest numbers that can be represented) and in many cases better precision.

Our definition of precision is based on the idea of the quantization error. This is the numerical error introduced when a longer numeric format is converted to a shorter one. The greater the possible quantization error relative to the size of the value represented, the less precision is available. For a fixed–point format, we define the maximum available precision to be equal to the number of bits in the format. For example, a 16–bit fractional format provides a maximum 16 bits of precision. This definition is based on computing the ratio of the size of the value represented to the size of the maximum quantization error

that could be suffered when converting from a more precise representation via rounding. Formally stated

$$\text{maximum precision (in bits)} = \log_2 (\text{maximum value} / \text{maximum quantization error}).$$

For a 16–bit fractional representation, the largest-magnitude value that can be represented is -1.0. When converting to a 16–bit fractional format from a more precise format via rounding, the maximum quantization error is 2^{-16}. Using the relation above, we can compute that this format has a maximum precision of $\log_2(1/2^{16})$, or 16 bits, the same as the format's overall width.

Note that if the value being represented has a smaller magnitude than the maximum, the precision obtained is less than the maximum available precision. This underscores the importance of careful signal scaling when using fixed–point arithmetic. Scaling is used to maintain precision by modifying the range of signal values to be near the maximum range of the numeric representation used.

Using this same definition for a floating–point format, the maximum available precision is the number of bits in the mantissa, including the implied integer bit. Because floating–point processors automatically scale all values so that the implied integer bit is equal to 1, the magnitude of the mantissa is restricted to be at least 1.0. This guarantees that the precision of any floating–point value is no less than half of the maximum available precision. Thus, floating–point processors maintain very good precision with no extra effort on the part of the programmer.

In practice, floating–point DSPs generally use a 32–bit format with a 24–bit mantissa and one implied integer bit, providing 25 bits of precision. Most fixed–point DSPs use a 16–bit format, providing 16 bits of precision. So, while in theory the choice between fixed and floating–point arithmetic could be independent of the choice of precision, in practice floating–point processors usually provide higher precision.

As mentioned above, dynamic range is defined as the ratio between the largest and smallest number representable in a given data format. It is in this regard that floating–point formats provide their key advantage. So, while using the same number of bits as the fixed–point format, the floating–point format provides dramatically higher dynamic range. In applications, dynamic range translates into a range of signal magnitudes that can be processed while maintaining sufficient fidelity. Different applications have different dynamic range needs. For telecommunications applications, dynamic range in the neighborhood of 50 dB is usually sufficient. For high-fidelity audio applications, 90 dB is a common benchmark. It's often helpful, though, if the processor's numeric representation and arithmetic hardware have somewhat more dynamic range than the application demands, as this frees the programmer from some of the painstaking scaling that may otherwise be needed to preserve adequate dynamic range.

Floating–point DSP processors are generally costlier than their fixed–point cousins, but easier to program. The increased cost results from the more complex circuitry required within the floating–point processor, which implies a larger chip. In addition, the larger word sizes of floating–point processors often means that off-chip buses and memories are wider, raising overall system costs.

The ease–of–use advantage of floating–point processors is due to the fact that in many cases the programmer does not have to be concerned about dynamic range and precision. On a fixed–point processor, in contrast, programmers often must carefully scale signals at various stages of their programs to ensure adequate numeric performance with the limited dynamic range and precision of the fixed–point processor.

Most high–volume, embedded applications use fixed–point processors because the priority is low cost. Programmers and algorithm designers determine the dynamic range and precision needs of their application, either analytically or through simulation, and then add scaling operations into the code if necessary. For applications that are less cost–sensitive, or that have extremely demanding dynamic range and precision requirements, or where ease of programming is paramount, floating–point processors have the advantage.

4.1.2.2 Native Data Word Width

The native data word width of a processor is the width of data that the processor's buses and data path can manipulate in a single instruction cycle. The size of the data word has a major impact on processor cost because it strongly influences the size of the chip and the number of package pins required as well as the size and number of external memory devices connected to the DSP. Therefore, designers try to use the chip with the smallest word size that their application can tolerate.

As with the choice between fixed–point and floating–point chips, there is often a trade-off between word size and development complexity. An application that appears to require 24–bit data for adequate performance can sometimes be coaxed into a 16–bit processor at the cost of more complex algorithms and/or programming.

4.1.2.3 Extended Precision

Extended precision means the use of data representations that provide higher precision than that of a processor's native data format. Extended precision can be obtained in two ways. First, many fixed and floating–point processors provide built-in support for an extended precision format for operations taking place exclusively within the data path of the processor. This means that as long as a series of arithmetic operations is carried out exclusively within the processor's data path and does not involve transferring

intermediate results to and from memory, a data word width larger than the native data word width is available.

This allows a series of arithmetic operations to be performed using extra precision and/or dynamic range, with a final rounding operation performed when the result is stored to memory. Second, it's generally possible, though often painful, to perform multiprecision arithmetic by constructing larger data words out of sequences of native-width data words. For example, with a 16–bit fixed–point processor, a programmer can form 32–bit data words by stringing together pairs of 16–bit words. The programmer can implement multiprecision arithmetic operations by using the appropriate sequences of single-precision instructions. Of course, because each multiprecision arithmetic operation requires a sequence of single–precision instructions, multiprecision arithmetic is much slower than single-precision.

However, some processors provide features that ease multiprecision arithmetic. These include the ability to preserve the carry bit resulting from a single-precision addition operation for use as an input into a subsequent addition, and the ability to treat multiplication operands as signed or unsigned under program control. If the bulk of an application can be handled with single-precision arithmetic, but higher precision is needed for a small section of the code, then the selective use of multiprecision arithmetic may make sense. If most of the application requires higher precision, then a processor with a larger native data word size may be a better choice, if one is available.

4.1.2.4 Floating–Point Emulation and Block Floating–Point

Even when using a fixed–point processor, it is possible to obtain the precision and dynamic range of general-purpose floating–point arithmetic by using software routines that emulate the behavior of a floating–point processor. Some processor manufacturers provide a library of floating–point emulation routines for their fixed–point processors. If a library is not available, then the emulation routines must be written by the user. Floating–point routines are usually very expensive to execute in terms of processor cycles. This implies that floating–point emulation may be appropriate if only a very small part of the arithmetic computations in a given application require floating–point. If a significant amount of floating–point arithmetic is needed, then a floating–point processor is usually the appropriate choice. Another approach to obtaining increased precision and dynamic range for selected data in a fixed–point processor implementation is a block floating–point representation. With block floating–point, a group of numbers with different mantissas but a single, common exponent is treated as a block of data. Rather than store the exponent within part of each data word as is done with general purpose floating–point, the shared exponent is stored in its own separate data word. For example, a block of eight data values might share a common exponent, which would be stored in a separate data word. In this case, storage of an

entire block of eight data values would require nine memory locations (eight for the mantissas and one for the exponent). Block floating–point is used to maintain greater dynamic range and precision than can be achieved with the processor's native fixed–point arithmetic formats. The conversion between the processor's native fixed–point format and block floating–point format is performed explicitly by the programmer through software. Some processors have hardware features to assist in the use of block floating–point formats. The most common of these is an "exponent detect" instruction. This instruction computes the shift needed to convert a high-precision intermediate result (for example, a value in an accumulator) to block floating–point format.

4.1.2.5 IEEE–754 Floating–Point

In 1985, the Institute of Electrical and Electronics Engineers released IEEE Standard 754 [IEE85], which defines standard formats for floating–point data representations and a set of standard rules for floating–point arithmetic. The rules specify, for example, the rounding algorithms that should be provided in a floating–point processor and how the processor should handle arithmetic exception conditions, such as divide by zero or overflow. A few commercial DSP processors provide partial hardware support for IEEE–754 floating–point formats and arithmetic. The Motorola DSP96002 features hardware support for single precision floating–point arithmetic as specified in IEEE–754. The Analog Devices ADSP–210xx family processors provide nearly complete hardware support for single-precision floating–point arithmetic as specified in the standard. Some other floating–point processors, such as the ATT DSP32xx, do not internally conform to IEEE–754, but do provide special hardware for fast conversion of numbers between the processor's internal floating–point representation and IEEE–754 representation. Hardware support for format conversion can be important in applications that require a non-IEEE–754-compliant DSP to interface with other processors that use the IEEE–754 representation. Without hardware conversion support, the noncompliant floating–point DSP must use software routines to convert between the different floating–point formats, and this software conversion can be quite time consuming. Therefore, developers of applications that require a DSP to interface with other processors that use the IEEE–754 representation should evaluate the practicality of software conversion carefully, or choose a processor with hardware conversion capabilities (or one that uses IEEE floating–point formats internally).

4.2 Custom Hardware

There are two important reasons why custom-developed hardware is sometimes a better choice than a DSP processor-based implementation [38]: performance and production costs. Just as DSP processors are more cost-effective for DSP applications than general-purpose processors because of their specializa-

tion, custom hardware has the potential to be even more cost-effective due to its more specialized nature. In applications with high sampling rates (for example, higher than 1/100th of the system clock rate), custom hardware may be the only reasonable approach.

For high volume products, custom hardware may also be less expensive than a DSP processor. This is because a custom implementation places in hardware only those functions needed by the application, whereas a DSP processor requires every application to pay for the full functionality of the processor, even if it uses only a small subset of its capabilities. Of course, developing custom hardware has some serious drawbacks in addition to these advantages. Most notable among these drawbacks are the effort and expense associated with custom hardware development, especially for custom chip design.

Custom hardware can take many forms. It can be a simple, small printed circuit board using off-the-shelf components, or it can be a complex, multiboard system, incorporating custom integrated circuits.

One of the most common approaches for custom hardware for DSP applications is to design custom printed circuit boards that incorporate a variety of off-the-shelf components. These components may include standard logic devices, fixed-function or configurable arithmetic units, field-programmable gate arrays (FPGAs), and function or application-specific integrated circuits (FASICs). As their name implies, FASICs are chips that are designed to perform a specific function, perhaps for a single application. Examples of FASICs include configurable digital filter chips, which can be configured to work in a range of applications, and facsimile modem chips, which are designed specifically to provide the signal processing functions for a fax modem and aren't useful for anything else.

As tools for creating custom chips improve and more engineers become familiar with chip design techniques, more companies are developing custom chips for their applications. Designing a custom chip provides the ultimate flexibility, since the chip can be tailored to the needs of the application, down to the level of a single logic gate. Of course, the benefits of custom chips and other hardware-based implementation approaches come with important trade-offs. Perhaps most importantly, the complexity and cost of developing custom hardware can be high, and the time required can be long. In addition, if the hardware includes a custom programmable processor, new software development tools will be required.

It is important to point out that the implementation options discussed here are not mutually exclusive. In fact, it is quite common to combine many of these design approaches in a single system, choosing different techniques for different parts of the system. One such hybrid approach, DSP core-based ASICs, was mentioned above. Others, such as the combination of an off-the-shelf DSP processor with custom ICs, FPGAs, and a general-purpose processor, are very common.

4.2.1 FPGA Architecture and Technology

FPGAs are a group of digital and user–programmable blocks (Gate Array), which are programmable in their functionalities and routing. As the acronym suggests, FPGAs grow out from gate arrays: which represent a particular digital technology that allows designers to realize tailored circuits on the basis of their needs, beginning from a standard architecture. This kind of technology called semi-custom differs from the full-custom one (like ASICs) where every single element is user-defined. Gate arrays are composed by a uniform logic gate matrix. Designers act on the final circuit, realized on the gate array, editing the last metal levels that link the defined logic gates. An FPGA device maintains some of the gate array features but its programming technique is totally different, indeed they are in-system programmable by users, directly on their workbench. High performances, low costs and a limited development time have decreed FPGA success in many applications. Most of FPGA applications reside in military projects, image processing, high–performance Digital Signal Processing (DSP) and other vector or matrix processing. The final result is a circuit suitable for designer necessities whose performances are very close to an ASIC development.

Figure 7: Example of an FPGA–based application

A first classification of FPGAs is done on the specific distribution of programmable elements and on the routing resources and logic: symmetrical FPGAs have the logic block distributed in a matrix and the routing logic passes horizontally and vertically between the programmable blocks. Otherwise row-based FPGAs are organized in parallel rows, with the logic gates along them, and the routing logic that

horizontally crosses programmable block rows. Lastly, another group called 'hierarchical' is organized connecting wide programmable blocks through routing resources.

Blocks and connections are carried out using advanced VLSI technologies, so that reliability issues are pre-emptively solved. The lapse of time needed to implement a first prototype is very short, because it requires only a software programming. The FPGA design flow is not trivial and it is very similar to design flows for VLSI technology, but with the benefit of being short and stable. One of the fundamental advantages is the possibility of modifying design errors in a little while because of the different test simulations for every design layer thus resulting in a simplification in designers' work to verify immediately the efficiency of a particular solution. Digital circuits production, implemented through programmable devices, exhibits an economic benefit: in fact when an application has been validated and released on the market, every FPGA realization cost is constant for the producer, only the software development environment expense is amortized. FPGA and PLD design is very profitable in every kind of project that foresees few units production. On the other hand, the expense for an ASIC device is amortized when many units are implemented, because photolithographic masks are produced only once. For these reasons, the industry of prototypes is the main beneficiary of the FPGA' economic benefits: it is worthwhile to realize a first prototype using an FPGA and moving towards the production through an ASIC device.

4.2.2 Advantages of FPGAs

In recent years a particular improvement in FPGA size and performance has been noted thanks to a number of factors, including technological advancement of finer chip geometries, higher level of integration, the use of faster serial and communication links, specialized cores, enhanced logic and innovative design. Meanwhile, the overall performance growth curve of traditional microprocessors has flattened because of power density hurdles. At the same time, the number of processor cores has increased bringing the new issue of coping with optimal use of parallelism between processes while operating systems and automatic parallelization tools lag far behind. FPGAs' first benefit derives from their ability to handle massively parallel processing. FPGAs are able to operate at a modest clock rate as hundreds of megahertz, but they can realize tens of thousands of computations per clock cycle while operating in tens of watts range of power. A similar microprocessor may run at 1-2 GHz, but it would be very limited in the number of clock-cycle operations, typically four or eight operations per cycle. This means that an FPGA can provide a 50 to 100 times the performance per watt of power consumed by a microprocessor. Nevertheless FPGA computational strength, there are three key factors that determine the utility of FPGAs for a specific application. These factors are algorithm suitability, floating point vs. fixed point number representation and general FPGA software difficulty.

The first mention to raise is for which kind of algorithms FPGAs are best suited. FPGAs are thought to be used in problems that can be easily and efficiently divided into many parallel, often repetitive, computational tasks. On the other hand, there are some specific cases where FPGAs are not well exploited, such as target classification and moving target indication problems. Indeed repetitive operations are FPGAs strong points and generally they are used in predictable and static problems.

A second issue is the fact that FPGAs are not suited to floating point calculations, which microprocessors on the contrary address with well developed vector math engines. FPGAs are able to cope with this kind of calculation, but they require an undue amount of logic to implement: this causes a limit in calculation density of the FPGA and decrease its main computational benefits.

The last key factor is the degree of technical capabilities involved in software development and the talent and resources available for the work. On one hand the challenges of designing with traditional microprocessors are well established and familiar. On the other hand even if FPGA development tools have improved significantly over the last few years, it always takes a skilled user to develop code for an FPGA. This aspect is on the overtaking: in fact more and more often a lot of development environments provide a simple way of FPGA programming. This is the case of LabVIEW programming language that does not require a hardware description language (HDL) to develop FPGA set-up.

In addition to these three main arguments, other considerations can be taken into account for example which sensor interfaces are required by an application, that can be directed towards FPGAs or microprocessors. Often FPGAs cover all the different kind of communication standards, not thinkable for microprocessor limitations.

4.2.2.1 A Forward–Looking Architecture

In the past, systems tended to be homogeneous that is composed of one type of processing element. Today, designers have a wide range of products to choose from and can often mix–and–match computing components to get just the right mix of computing and I/O to meet their application specs. Using a hybrid FPGA-microprocessor architecture, it is possible to customize a system, providing FPGA components where they are useful with more DSPs or general-purpose microprocessors for the portions of application for which they are more suitable.

As FPGAs have grown larger and faster over the last decade, they have assumed a more central role in embedded processing applications. System–On–Chip is becoming a complete novel discipline able to let designers create new system solutions using the state of the art about what markets propose. An embedded system has some designed tasks well–known during its development, these will be executed through a hardware and software combination studied for that specific application. This is an important

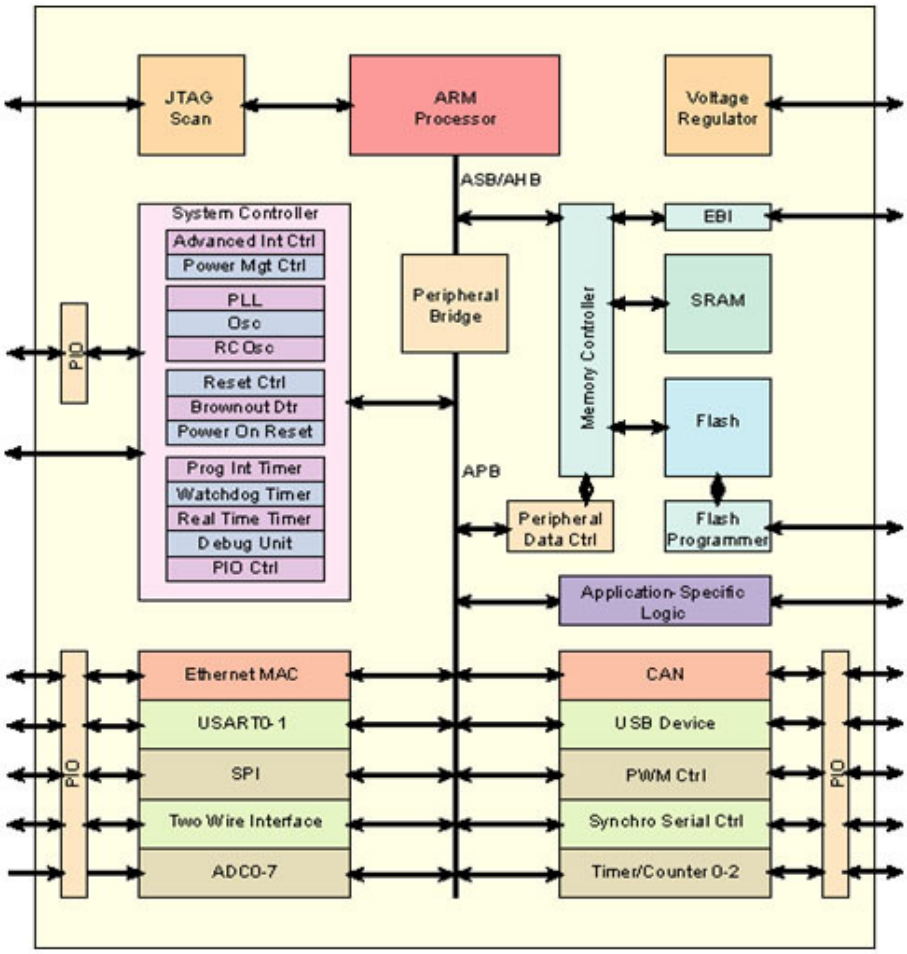

Figure 8: Base structure of a System–On–Chip architecture

benefit because the hardware resources can be heavily reduced to optimize the circuit occupation, consumption and costs. Furthermore the software counterpart execution is often real–time to allow users to reach a deterministic control of the system evolution.

A SoC implementation in FPGA technology permit to include the processors and a DSP cores inside the FPGA architecture, in order to manage a complete integrated instrument able to satisfy every requirement in several applications. The most important vendors of FPGA provide hard-core and soft-core devices easy to be introduced in the FPGA programming citeart:Minev-2007. The former are real physical areas where a microprocessor or a DSP core resides that are realized as a layout level, and it depends on the FPGA technology; the latter are a general hardware decription totally indipendent from the adopted technology, that are recognized by the FPGA programmer tool and automatically synthesized and inserted during the place and route phase.

Designers customize these soft cores and the surrounding logic to the task at hand. Recently, FPGA

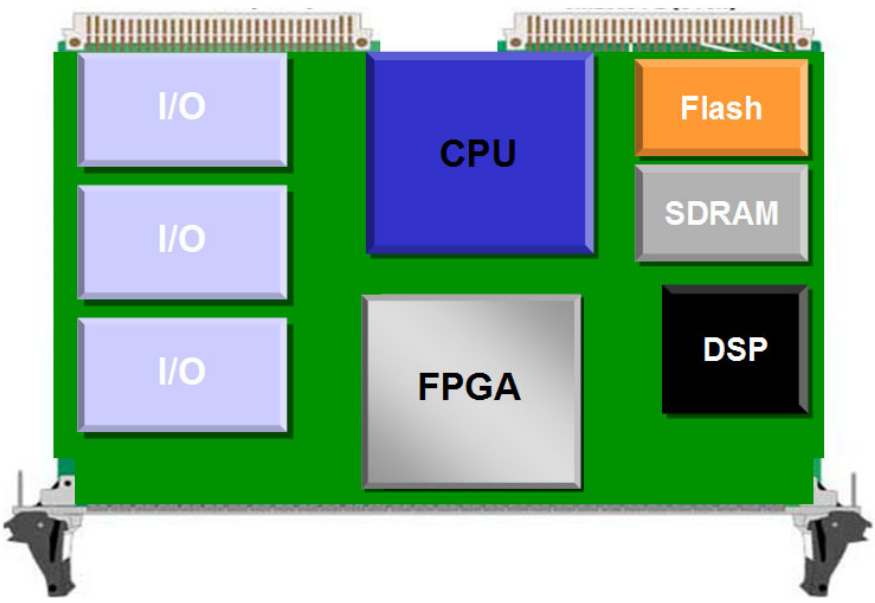

Figure 9: Plethora of components in a typical industrial PCB

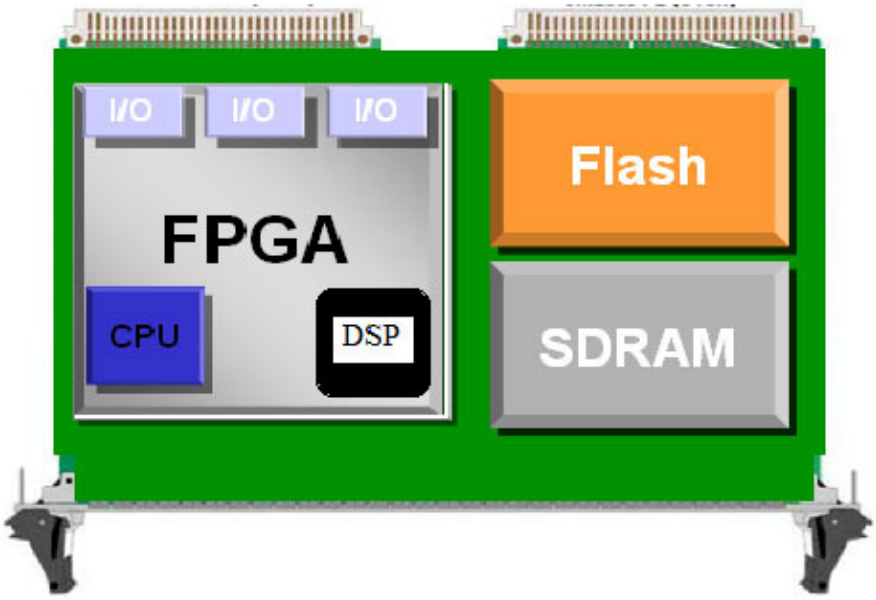

Figure 10: Novel FGPAs including DSP and CPU cores

vendors have taken this idea a step further, developing ICs that now include full ARM processing subsystems along with hardened peripherals, memory controllers, etc. all tightly integrated with the FPGA fabric. This marriage of hardened processor cores within the fabric gives designers the ease of programming with a familiar real time operating system (RTOS), yet has opened up new doors for customization of the overall processing system, with much tighter linkage between data and controlling processing.

4.3 FPGA in Control Schemes

Embedded control systems are found in a wide range of applications such as consumer electronics, medical equipment, robotics, automotive products, and industrial processes. For such systems, control algorithms are implemented as software programs that execute on a fixed architecture hardware processor.

The question that we must answer before we proceed is, with a plethora of embedded devices available for digital control: 'Why must one go in for embedded control using FPGA'? This can be answered by looking at the following advantages that FPGA possess. Most of computations in control involves the use of 2 operations. The first one being the Multiply operation and the second one being the accumulate operation. Together these operations are called Multiply ACcumulate (MAC) operations. The computational overhead is the maximum when any kind of digital controller is performing these operations. Hence the sampling rate and hence speed is limited by the rate at which the device performs these computations. In a general purpose microprocessor the processors resources are held up while it is busy performing these MAC operations and the speed or the sampling rate is decided by the latency of these instructions. In addition to this important feature, FPGAs can exploit other benefits in order to offer excellent parallelism, reconfigurable configuration and rapid prototyping. FPGAs are also fundamentals to implement PWM generators whose signals of high frequencies and precise duty–cycle resolution.

4.4 FPGA Versus DSP Processor Developing Control Loops

In order to ensure fair and square comparison between FPGA and general purpose processors, let us examine the operation of implementing a digital filter. It is a well known fact that many of the controllers that are designed are ultimately implemented as digital filters. Hence in order to illustrate the power of the FPGA, let us look at the specific implementation of a 256 tap filter on a typical DSP processor and an FPGA. The conventional DSP processor is a general purpose programming device that typically has 1–4 MAC units along with barrel shifters and other circuits optimized for efficient computations.

The conventional DSP is a serial device. Let us assume that it has got a single MAC unit. A 256 tap filter involves 256 MAC operations per sample. Hence with a single MAC unit, it takes 256 clock cycles for the output to be computed in a typical DSP processor. In order to improve the system throughput, we have to look at other options like using a high frequency clock generator. This increases the system complexity and the cost. Moreover the chances for clock skew occurring with high frequency clocks is also high. On the other hand, let us look at the same filter implemented on a typical FPGA.

Let us consider the most important feature of a FPGA–parallelism. The FPGA contains a large number of gates and millions of transistors. Hence we can implement the filter in a parallel manner.

The implementation consists of 256 registers and 256 multiplier units along with the addition of the final partial product. Hence what took 256 clock cycles in a DSP can be completed in a single clock cycle in an FPGA. This results in a tremendous improvement in the latency of each instruction.

Now let us look at some of the other features that FPGA based embedded control offers to us. The speed of a control system impacts its performance, stability, robustness and disturbance rejection characteristics. Faster control systems are typically more stable, easier to tune, and less susceptible to changing conditions and disturbances. To provide stable and robust management, a control system must be able to measure the process variable and set an actuator output command within a fixed period of time. The computational performance of the FPGA is so fast that the control loop rate is limited only by the sensors, actuators, and I/O modules. This is a stark contrast to traditional control systems, where the processing performance was typically the limiting factor. One of the most important parameters that is involved in performance measurement of digital control systems is loop cycle time. Loop cycle time is the time taken to execute one cycle of the control loop. It is the time that elapses between sampling the output, computing the controller output according to the control algorithm and sending the control signal to the actuator. Because of the inherent parallelism present in the FPGA, very low loop cycle times are possible.

Another common measure of control system performance and robustness is jitter, which is a measure of the variation of the actual loop cycle time from the desired loop cycle time. In general purpose operating systems such as Windows, the jitter is unbounded so closed loop control system stability cannot be guaranteed. Processor-based control systems with real–time operating systems are commonly able to guarantee control loop jitter of less than 100 microseconds. In FPGA based systems the control loop does not need to share hardware resources with other tasks and control loops can be precisely timed using the FPGA clock. The jitter for FPGA–based control loops depends on the accuracy of the FPGA clock source. It typically ranges in the order of picoseconds. The FPGA can effectively be used as a prototyping device in order to get the control algorithms fine tuned and running correctly. The wide of design tools available for FPGA's make it very easy in order to build a prototype of the control algorithm that we wish to implement and understand and refine the various issues like timing and signal integrity. One can even design the controller in a control systems design package like MATLAB or LabVIEW environment and use the VHDL or Verilog descriptions of the controller thus generated to fuse it on to the FPGA prototyping board. The FPGA thus plays a very important role in prototyping the controller even if the ultimate goal is the creation of an Application Specific Integrated Circuit (ASIC) controller for the application at hand. FPGA has another advantage in the fact that the design cycle time for the controller is less in an FPGA rather than an ASIC. In some cases it may be economical for the controller to be implemented in a FPGA rather than an ASIC. The FPGA also consumes lesser power than the

microprocessor based or ASIC based controllers. The FPGA design now consists of the steps of creation, simulation, verification, synthesis, placement and routing of the design. A lot of computer based tools are available for this purpose, which is yet another argument in FPGA's favor. Hence we can safely arrive at a justification for the use of FPGA in control applications.

4.4.1 FPGA–Based PID Controller

In this section we give an example of how a controller can be implemented using FPGAs. The PID controller is the most used algorithm in industry. Also the controllers (8), (9) have PI terms. In a continuous domain the output is computed as follows

$$u(t) = k_p \left[e(t) + \frac{1}{T_i} \int_0^t e(t)\, dt + T_d \frac{de(t)}{dt} \right]$$

where k_p is the proportional gain, T_i is the reset time and T_d is the derivative time. In the FPGA technology, it is possible to realize two different typologies of PID controller, a serial design or a parallel one. In this paragraph a first comparison between parallel and serial structure has been done considering the resource utilization, speed and power consumption. The former equation is discretized, obtaining

$$u_k = k_p e_k + k_i \sum_{j=0}^{k-1} e_j + k_d (e_k - e_{k-1}) \tag{44}$$

where $\kappa_i = \kappa_p T / T_i$ is the integral coefficient and $\kappa_d = \kappa_p T_d / T$ is the derivative coefficient. This form is known as the position form of the PID algorithm. An alternative would be to compute u_k based on past output u_{k-1} and correction term Δu_k. This approach is often called as the velocity form of the PID algorithm. The first step in this regard would be to calculate u_{k-1} based on equation (44)

$$u_{k-1} = k_p e_{k-1} + k_i \sum_{j=0}^{k-1} e_j + k_d (e_{k-1} - e_{k-2}).$$

Then, one calculates the correction term as

$$\Delta u_k = u_k - u_{k-1} = k_0 e_{k-1} + k_1 e_{k-2} + k_2 e_{k-3}$$

where

$$k_0 = k_i + k_p + k_d, \qquad k_1 = -k_p - 2k_d, \qquad k_2 = k_d.$$

Hence, the current control output is calculated as

$$u_k = u_{k-1} + \Delta u_k = u_{k-1} + k_0 e_k + k_1 e_{k-1} + k_2 e_{k-2}.$$

The above equation is decomposed into its basic operations. Here p and p_d refers to the controlled variable and its desired value (set point) respectively. Moreover, p_0, p_1, p_2, s_1, s_2 are temporary variables.

$$e_k = p + (-p_d)$$

$$p_0 = k_0 e_k$$

$$p_1 = k_1 e_{k-1}$$

$$p_2 = k_2 e_{k-2}$$

$$s_1 = p_0 + p_1$$

$$s_2 = p_2 + u_{k-1}$$

$$u_k = s_1 + s_2.$$

For parallel design, each basic operation has got its own arithmetic unit either an adder or a multiplier. In serial design, which is mainly composed of sequential logic. All operations share only one adder and one multiplier.

4.4.2 Parallel Design

The parallel implementation uses 4 adders and 3 multipliers corresponding to the basic operations. The architecture diagram is shown in the following figure. The other circuitry includes registers for latching initial and intermediate values of error and output signals. The implementation also includes value limitation logic that keeps the signals generated by the control logic within limits that the physical device can bear.

4.4.3 Serial Design

In order to minimize the area and the resources consumed for the design, the serial design consists of only one adder and one multiplier. The other parts in the implementation include registers, multiplexers and circuits for arithmetic operations. They are commonly refereed to as the data–path circuits. Registers are used to store intermediate results. Because of the fact that the single adder multiplier unit is used in a time shared manner, there is the necessity of a control unit which is a finite state machine that sets the select lines of the multiplexers; thereby changing the input to the circuits. The results of those tests that have relevance to the problem are presented.

1. Resource Utilization: it was found that the serial implementation consumed far less resources on the FPGA than the parallel implementation. Even though the serial implementation includes a

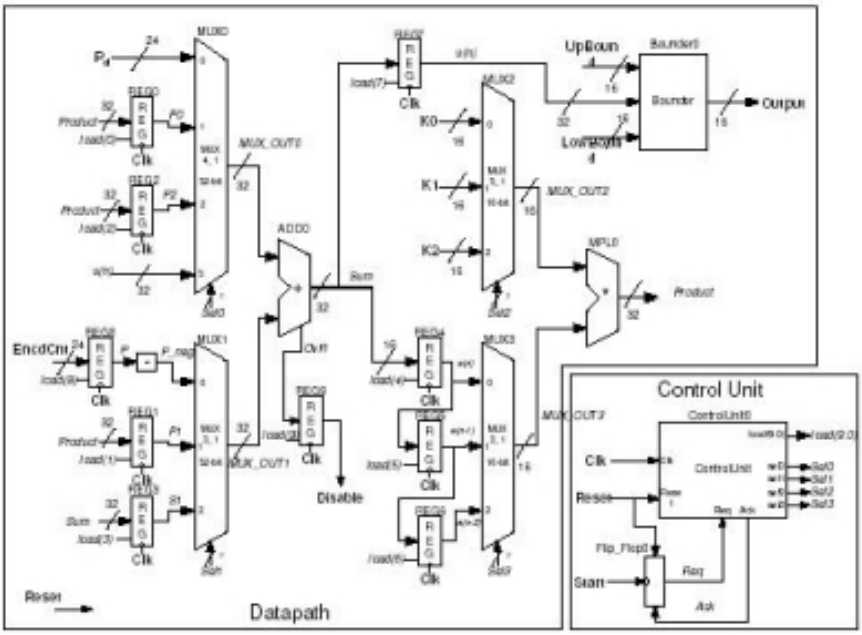

Figure 11: Serial Implementation of PID in FPGA

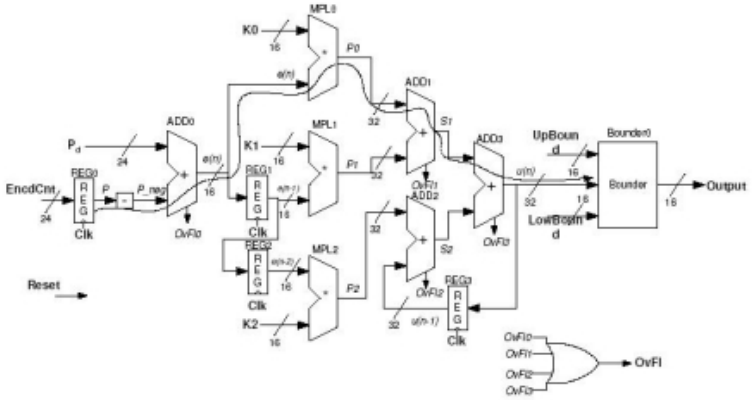

Figure 12: Parallel Implementation of PID in FPGA

control unit, it was found to consume far lesser number of CLBs to implement.

2. Speed: Test have been led with the Xilinx timing analyzer and it was found that in each design there were two timing concerns. The first one was the control clock frequency. This controlled the timing cycles of the PID algorithm. The next is the sampling frequency. This corresponds to the rate at which the control algorithm generates control signals; this is dependent on whether the implementation is a serial one or a parallel one. For the parallel implementation which is essentially

a combinational logic implementation, the sampling frequency and the control clock frequency are the same. This is a result of the inherently parallel nature of such an implementation. On the other hand, the serial algorithm requires four clock cycles to compute all the four basic operations specified in equations (3.9) – (3.14). Hence the sampling frequency for the serial implementation would be 1/4 of the control clock frequency.

3. Power Dissipation: The power dissipation increased as the sampling frequency was increased. At reasonable sampling frequencies, there wa no difference between the parallel and serial designs, even though the parallel design was expected to be more power efficient because of much lower sampling frequency.

4.4.4 A More Efficient PID

In the previous section we had looked at an implementation of a PID controller based on multipliers and adders. But when we are implementing PID controllers in LUT rich FPGA's, any design that does not make use of the memory rich characteristics of the FPGA is not an optimal implementation. It should however, be mentioned that this type of PID implementation is more efficient only in those kinds of FPGA that are rich in LUT's. An improved implementation of a PID Controller is based on Distributed Arithmetic (DA) concepts. The continuous PID equation (3.1) is modified as follows in order to avoid problems of spikes in the output because of the derivative term. These spikes occur when the user tries to change the set point abruptly. If the derivative term acts on the set point, then a sudden change in the set point would result in spikes in the output.

$$U(s) = K\left[bU_c(s) - Y(s) + \frac{1}{sT_i}(U_c(s) - Y(s)) - \frac{sT_d}{1 + \frac{sT_d}{N}}Y(s)\right]. \tag{45}$$

In (45), it is advantageous to allow only a fraction of the command signal act on the proportional part. Here k_i is the integral gain, k_d is the derivative gain, K is the proportional gain, U_c is the set point and Y is the process value. U is the controller output. Discretizing equation (45) by using the forward differences for the derivative term and backward differences for the integral term one has

$$u(kT) = P(kT) + I(kT) + D(kT)$$

where k denotes k–th sampling instant and

$$P(kT) = K(bu(kT) - y(kT))$$

$$I(kT) = I((k-1)T) + \frac{kT}{T_i}u((k-1)T) - y((k-1)T)$$

$$D(kT) = \frac{T_d}{T_d + NT}(D(k-1)T) - \frac{KT_dN}{T_d + NT}(y(kT) - y((k-1)T))$$

where $y_k = y(kT)$ is the output at the current instant, $y_{k-1} = y((k-1)T)$ is the output at the previous instant, u_c is the desired output of the system, $I((k-1)T$ is the value of the integral term at the previous instant, $D((k-1)T)$ is the value of the derivative at the previous instant, K, b, T_i, T_d, N are controller parameters, T is the sampling time. The direct implementation of the above equation requires 5 multipliers, 5 adder subtractors and 4 delay elements. The multiplier based design is not efficient for FPGA implementation because of the fact that the FPGA has got limited number of CLB's for implementing the above logic circuits. A better implementation would be the DA Based implementation. Assuming that $u(kT)$, $u((k-1)T)$, $y(kT)$, $y((k-1)T)$ are m bit numbers and $[j]$ represents the j^{th} bit of these numbers, we obtain the following equations

$$P(kT) = \sum_{j=0}^{m-1}(k_b * u(kT)[j] - k * y(kT)[j]) * 2^j$$

$$I(kT) = \sum_{j=0}^{m-1}(I((k-1)T)[j] + \frac{kT}{T_i}(u((k-1)T)[j] - y(((k-1)T)[j]) * 2^j$$

$$D(kT) = \sum_{j=0}^{m-1}(\frac{T_d}{T_d + NT}D((k-1)T)[j] - \frac{kT_dN}{T_d + NT}((y(kT)[j] - y((k-1)T)[j])) * 2^j.$$

The results of

$$(k_b * u(kT)[j] - k * y(kT)[j])$$

$$(I((k-1)T)[j] + kT/T_i(u((k-1)T)[j] - y(((k-1)T)[j])$$

$$(T_d/T_d + NTD((k-1)T)[j] - kT_dN/T_d + NT((y(kT)[j] - y((k-1)T)[j]$$

are precomputed and stored in various look up tables. Using the three LUT's and corresponding shift add accumulators. The $P(kT)$, $D(kT)$, $I(kT)$ terms can be computed in m clock cycles. The main advantage of this method is the fact that it utilizes the LUT rich feature of the FPGA for computing the control effort.

The DA implementation for this particular implementation will consists of four delay blocks, 3 LUT's, 3 accumulators, 2 adders. Delay blocks are used to obtain $U((k-1)T)$ and $y(k-1)T$ respectively, whereas delay blocks are used to compute $D(k-1)T$ and $I(k-1)T$. Three LUT's and ACC's are used to provide the terms $P(kT)$, $I(kT)$, $D(kT)$ respectively. The ACC consists of an accumulator and an adder subtractor pair. Finally two adders produce the sum of $P(kT)$, $I(kT)$, $D(kT)$. The throughput of this implementation is $m + 1$ clock cycles, i.e. m clock cycles to compute U and one more clock cycle to update $I((k-1)T)$ and $D((k-1)T)$. Thus we find that the DA based implementation consumes far less number of logic resources than the parallel multiplier based design. Hence the design using DA would require 14 clock cycles to implement in comparison to the design based on multipliers that would take

just a single clock cycle. Since power saving is dependent upon the clock frequency, the reduction in power consumption and the reduction in clock frequency would be advantageous in those applications which can tolerate the increased loop cycle time, resulting form the predominantly serial implementation of the DA based controller.

Conclusions

In this deliverable some aspects of the digital implementation of a control law on a physical device have been studied in order to reduce the deterioration of the control performances once implemented on a digital device, possibly bringing to unstable behaviors. Two aspects have been studied. The first deals with a self–triggered implementation, determining the sampling times necessary to implement the controller preserving the desired performance. The second deals with the physical device on which the control is implemented. Among the various possible solutions, we concentrated on the FPGA technology, which shows to be particularly interesting, especially in terms of parallel computation, control and digital signal processing.

References

[1] S. Di Gennaro and B. Castillo–Toledo, Comparative Study of Controllers for the Supervision, Control and Protection Systems in Pressurized Water Reactors of Evolutive Generation, Deliverable 2, PAR2010 Project, 2011.

[2] S. Di Gennaro and B. Castillo–Toledo, Performance Study of the Control Systems in the presence of Faults and/or Reference Accidents in Pressurized Water Reactors of Evolutive Generation, Deliverable 2, PAR2010 Project, 2011.

[3] I. F. Akyildiz, and I. H. Kasimoglu, *WirelessHART: Wireless sensor and actor networks: research challenges, Ad Hoc Networks*, Vol. 2, No. 4, pp. 351–367, 2004.

[4] A. Anta, and P. Tabuada, Self–Triggered Stabilization of Homogeneous Control Systems, *Proceedings of the 2008 American Control Conference – ACC 2008*, pp. 4129–4134, 2008.

[5] A. Anta, and P. Tabuada, To Sample or not to Sample: Self–Triggered Control for Nonlinear Systems, *IEEE Transactions on Automatic Control*, Vol. 55, No. 9, 2010.

[6] M. D. Di Benedetto, S. Di Gennaro, and A. D'Innocenzo, Digital Self Triggered Robust Control of Nonlinear Systems, *Proceeding of the* 50^{th} *Conference on Decision and Control and European Control Conference*, Orlando, FL, USA, pp. 1674–1679, 2011.

[7] W.P.M.H. Heemels, A.R. Teel, N. van de Wouw and D. Neöic. Networked Control Systems with Communication Constraints: Tradeoffs between Transmission Intervals, Delays and Performance. *IEEE Transactions on Automatic Control*, Vol. 55, No. 8, pp. 1781-1796, 2010.

[8] H. K. Khalil, *Nonlinear Systems*, Third Edition, Prentice Hall, Upper Saddle River, New Jersey, U.S.A., 2002.

[9] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks*, Wiley, 2005.

[10] J. Kurzweil, On the Inversion of Liapunov's Second Theorem on Stability of Motion, *Translation of American Mathematical Society*, Vol. 24, pp. 19–77, 1963. Originally appeared on *Czechoslovak Mathematica Journal*, Vol. 81, pp. 217–259, 1956.

[11] M. Lemmon, T. Chantem, X. Hu, and M. Zyskowski, On Self–Triggered Full Information H–infinity Controllers, in *Hybrid Hybrid Systems: Computation and Control*, 2007.

[12] X. Wang, and M. Lemmon, Self–Triggered Feedback Control Systems With Finite–Gain $\mathcal{L}_2$ Stability, *IEEE Transactions on Automatic Control*, No. 3, Vol. 54, pp. 452-467, 2009.

[13] M. Mazo, and P. Tabuada, On Event–Triggered and Self–Triggered Control over Sensor/Actuator Networks, *Proceedings of the* 47^{th} *Conference on Decision and Control*, Cancun, Mexico, pp. 435–440, 2008.

[14] P. Tabuada, Event–Triggered Real–Time Scheduling of Stabilizing Control Tasks, *IEEE Transactions on Automatic Control*, Vol. 52, No. 9, pp. 1680–1685, 2007.

[15] M. Velasco, P. Marti, and J. Fuertes, The Self Triggered Task Model for Real–Time Control Systems, *Work–in–Progress Session of the* 24^{th} *IEEE Real–Time Systems Symposium – RTSS03*, pp. –, 2003.

[16] W. Aggoune, Contribution to the Stabilization of Stochastic Nonlinear Systems with Time Delays, *Proceedings of the* 18^{th} *IFAC World Congress* Milano, Italy, August 28–September 2, pp. 3885–3890, 2011

[17] W. Aggoune, On Feedback Stabilization of Stochastic Nonlinear Systems with Discrete and Distributed Delays, *Proceedings of the* 50^{th} *IEEE Conference on Decision and Control, and European Control Conference (CDC–ECC)*, Orlando, FL, USA, December 12–15, pp. 6296–6301, 2011.

[18] A. Anta and P. Tabuada, Exploiting Isochrony in Self–Triggered Control, *Submitted for publication, Preprint available on arXiv:1009.5208*, September 2010.

[19] L. Arnold, *Stochastic Differential Equations: Theory and Applications*, Wiley, 1972.

[20] K. J. Åström and B. Wittenmark, *Computer Controlled Systems*, Prentice Hall, Englewood Cliffs, NJ, USA, 1990.

[21] K. E. Årzén, A Simple Event Based PID Controller, *Proceedings of the* 14^{th} *IFAC World Congress*, Vol. 18, pp. 423–428, 1999.

[22] W. P. Heemels, J. H. Sandee, and P. P. Bosch, Analysis of Event–Driven Controllers for Linear Systems, *International Journal of Control*, Vol. 81, No. 4, pp. 571–590, 2008.

[23] R. Z. Khasminskii, *Stochastic Stability of Differential Equations*, Sijthoff and Noordhoff, Alphen aan den Rijn, 1980.

[24] V.B. Kolmanovskii and A. Myshkis, *Applied Theory of Functional Differential Equations, Mathematics and Its Application*, Kluwer Academic Publishers, Dordrecht, 1992.

[25] H. J. Kushner, *Stochastic Stability and Control*, Academic Press, 1967.

[26] H. J. Kushner, Converse Theorem for stochastic Lyapunov functions, *SIAM Journal of Control and Optimization*, Vol. 5, pp. 228–233, 1967.

[27] X. Mao, *Stochastic Differential Equations and Applications*, Horwood, 1997.

[28] S.–E.A. Mohammed, *Stochastic Functional Differential Equations*, Longman, 1986.

[29] D. Nesic and L. Gruene, Lyapunov Based Continuous–Time Controller Redesign for Sampled–Data Implementation, *Automatica*, Vol. 41, No. 7, pp. 1143Ð1156, 2005.

[30] D. Nesic and A. R. Teel, Stabilization of Sampled–Data Nonlinear Systems via Backstepping on their Euler Approximate Model, *Automatica*, Vol. 42, No. 10, pp. 1801Ð1808, 2006.

[31] P. G. Otanez, J. R. Moyne, and D. M. Tilbury, Using Deadbands to Reduce Communication in Networked Control Systems, *Proceedings of the American Control Conference 2002*, Vol. 4, pp. 3015–3020, 2002.

[32] P.E. Protter, *Stochastic Integration and Differential Equations*, 2^{nd} Ed., Springer, 2003.

[33] M. Velasco, J. Fuertes, and P. Marti, The Self Triggered Task Model for Real–Time Control Systems, *Proceedings of the Real–Time Systems Symposyum – RTSS03*, Work Progress Track, pp. 67–70, 2003.

[34] Y. Yamamoto, B. D. O. Anderson, and M. Nagahara, Approximating Sampled–Data Systems with Applications to Digital Redesign, *Proceedings of the IEEE Conference on Decision and Control*, Las Vegas, NV, 2002, Vol. 4, pp. 3724Ð3729, 2002.

[35] T. Yoshizawa, *On the Stability of Solutions of a System of Differential Equations*, Memoirs of the College of Sciences, University of Kyoto, Ser. A, Vol. 29, pp. 27–33, 1955.

[36] X. Wang, and M. Lemmon, State Based Self–triggered Feedback Control Systems with L_2 Stability, *Proceedings of the* 17^{th} *IFAC World Congress*, pp. 15238–15243, 2008.

[37] X. Wang, and M. Lemmon, Self–triggered Feedback Control Systems with Finite–Gain L_2 Stability, *IEEE Transactions on Automatic Control*, Vol. 45, No. 3, pp. 452-467, March 2009.

[38] S. Edwards, Microprocessor or FPGAs? Making the Right Choice, *RTC Magazine*, 2011.

[39] D. Strenski, P. Sundararajan, and Ralph Wittig, The Expanding Floating–Point Performance Gap Between FPGAs and Microprocessors, *HPC Wire*, 2010.

[40] P. B. Minev, and V. Stoianova Kukenska Implementation of Soft–Core Processors in FPGAs, *UNITECH'07*, 2007.