

Feedback Control Learning Model for QoS, Power & Performance Management of Reconfigurable Embedded Systems

Jean-Philippe Diguët, Yvan Eustache, Milad El Khodary
LESTER, CNRS/UBS University, Lorient, France, (Jean-Philippe.Diguët@univ-ubs.fr)

Abstract

This paper focuses on a usually ignored issue, the auto-configuration modelling and decision in the context of reconfigurable embedded systems. We propose an original and generic method based on control theory including stability analysis. Our approach addresses the question of local vs global reconfiguration decision at hardware, software and RTOS levels. We tackle the run-time uncertainty conditions with a learning system model that balances trade-offs between accuracy and complexity using sensors and run-time light estimators based on signal processing theory. Then we figure out how our method can be applied to a smart camera.

1 Introduction

1.1 Problem formulation

Mobile multi-media systems are characterized by scarce energy, memory and computation resources, which must be used efficiently. However power, performance and quality of service (QoS) optimizations can only be partially performed at design time since the system environment can be very uncertain. The CPU and communications loads vary with tasks' changeable execution time. This can be due to telecommunication conditions (e.g. channel signal-noise ratio), to the multimedia contents (object speed and complexity) and the user choices. Thus a part of design decisions should be dynamically performed at run time in order to systematically optimize resource usage to meet the application constraints in terms of QoS, throughput and lifetime. To face the run-time design challenges, the system must be flexible and reactive, it means that a real-time operating system (RTOS) must be in charge of resource management. The design of the resource manager should validate three properties. First, the addition of the control logic should be rentable i.e. additional overhead due to the manager added logic must be smaller than the expected absolute gain. Secondly, the system must be adaptive so that it can estimate and apply on-line candidate configurations. Finally the system must be stable. In this paper we present our approach to deal with these antagonist constraints. Our method combines the usual control theory with a learning-based simple modelling of the system behavior.

The rest of the paper is organized as follows, in section 1.2 we present relevant relative work and formalize our approach in section 2. Our global regulation global algorithm is detailed in section 3, then we introduce the smart camera case study, finally we conclude.

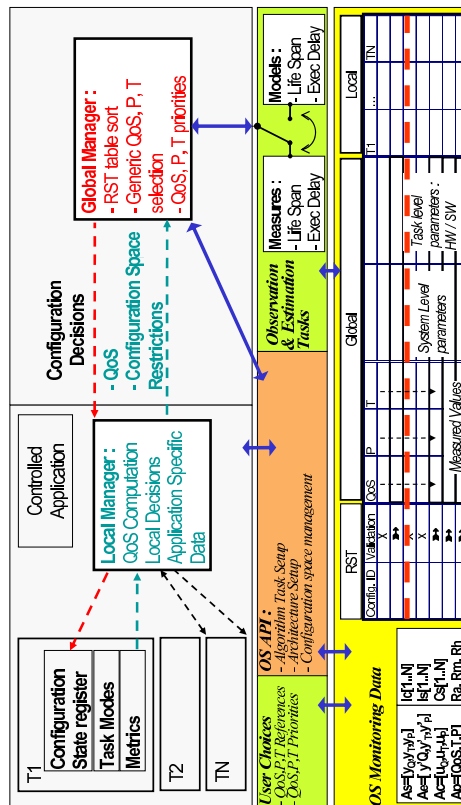


Figure 1: System Control Overview

1.2 Related work

In the domain of adaptive embedded systems four issues are related to our work. Firstly, a lot of work has been produced in the domain of adaptive architectures. Different techniques have been introduced for clock and voltage scaling [1], pipeline control [2], cache resource allocation [3], functional units [4] and Network on Chip (NoC) bandwidth adaptation [5]. These approaches can be classified in the category of local configurations based on specific aspects. Our aim is to add a global configuration management including both algorithmic and architectural aspects. In [6] dynamic algorithm selection combined with an architecture parameterization for MPEG-4 is detailed. Here, the association between algorithmic and architectural views is pertinent, however the hardware controller still remains local. Secondly, even though dynamically reconfigurable coarse-grain architectures have been already proposed, the conditions for reconfiguration decisions remain ignored.

Thirdly, RTOS for hardware reconfiguration manage-

ment have been recently introduced. Proofs of concepts are demonstrated in [7] and [8]. These experiments show that RTOS level management of reconfigurable architectures can be considered as available from a research perspective. Finally, if we assume that specific adaptive and reconfigurable architectures are also mature, we can conclude the need for decision strategies. The objective is to take benefit from the reconfiguration space in order to adapt embedded systems to application and user demands.

From the software point of view, QoS management has been deeply explored for web services applications. In [9] a prototype operating system (OS) can handle different QoS dimensions and allocation strategies in the context of server / mobile video application. Finally, feedback control has been recently studied in the area of soft RTOS to handle the uncertainty of worst case execution time (WCET). In [10] authors present a complete model for feedback control real-time scheduling. It is based on CPU utilization and deadline miss ratio as references. The QoS is strongly related to real-time deadlines and architectural aspects and power are not considered. In [11] a relevant two step approach is proposed. The target application is PC-based video server that can select frame QoS depending on its workload. However this kind of technique doesn't fit for embedded low power systems.

2 Concept formalization

2.1 Local/global separation of concerns

The reconfiguration issue encompasses different aspects that drive the implementation of the reconfiguration decision control. The first relevant point is the locality. A reconfiguration can be decided at the application level or at the system level. Based on application specific data, a local decision provides a short reaction delay and metrics to compute the QoS. However some decisions must be considered globally, in that case a trade-off can be decided over the complete system including the power consumption and the computation efficiency of the whole system. An application is a set of dependent tasks so the choice also impacts the application specification by means of task and intra-task control definitions. Another relevant point is the complexity of the decision implementation. In the context of embedded systems only low cost solutions must be considered. The third point is the reconfiguration type. Actually it can be i) a hardware reconfiguration (FPGA, dynamic, partial, coarse-grain), ii) an algorithmic reconfiguration regarding the different versions for a given task, iii) a NoC management including guaranteed traffic and best effort paths and slot table definitions and iv) an OS policy (priority assignment, scheduling techniques, time slot allocation). To cope with these issues we implement a two step configuration management.

2.2 Two step configuration strategy

Our Space Configuration Model is based on a two stages structure. The Local Configuration Manager (LCM) is the lowest level, located near the applications tasks. Its role is to supervise all tasks, to read the metrics of each task to compute them and to provide the desired configuration to the upper configuration level. This is equivalent to the factory management where the executive office needs to manage the production of factories and to inform the office head about the states of production and

desired configurations. The head office is in our system the Global Configuration Manager (GCM). This task has a global sight of the system. The System Parameters are global parameters including the battery power consumption, the Quality of Service and the execution time of the application. Respectively, in the industry, the office head gives some information from an audit commission ensuring that resources are used efficiently and that products will be correctly designed and provided in time. The GCM takes care of algorithmic configuration LCM wishes and user constraints. In our hierarchical configuration model, the GCM is standardized and application independent. On the contrary, the LCM is developed for each application and some tasks have a metric layer to describe the actual state of the application and to allow the choice of a new configuration. Regarding, the application the LCM can be designed as a hardware task to unload the processor from the configuration tasks.

The LCM is in charge of the algorithm selection for all the tasks of the application it controls. Given results from all the application tasks, it first computes the QoS as a kind of "application sensor" for the global manager and secondly it restricts the global configuration space regarding local algorithmic decisions.

The GCM is in charge of global system parameters (e.g. I/O data rates) and hardware/software implementations decisions. It receives data from sensors (gas gauge, system cpu load, application specific QoS) and from estimators when no measures are available. The GCM decides the new system configuration according to user requirements (system references) and configuration restrictions from the local managers.

2.3 OS API requirements

Global reconfiguration means to add new OS services in order to facilitate a generic implementation of reconfiguration control.

The uplink APIs provide the OS with i) application status namely the local configuration decisions, the priority choices (QoS, Power, Real-Time), ii) system status (QoS, Power, Real-time features) and iii) global reconfiguration decisions to be handled by the OS.

The downlink APIs enables the OS to apply algorithmic and architectural global reconfiguration decisions, these transformations are applied conjointly with task setup (e.g. period specification).

2.4 Definitions

Reconfiguration space dimensions At a system level we consider three generic dimensions:

- QoS means the processing quality for one iteration of the application;
- T is the execution time for one application, namely the Real-Time constraint;
- P is the average power consumption. In practice this term is measured with a battery gauge [12] and related to the system lifespan.

From a system view, an application is defined as :

- $Ac = [u_Q, u_T, u_P]$ represents the user choice constraints (i.e. references); in practice P is derived from the user required span life.

- $As = [y_Q, y_T, y_P]$ is the controlled application status, i.e. the real performances;
- $Ap = [QoS, T, P]$ is the priority order chosen by the user, i.e. $Ap = [0, 1, 2]$ means that the constraint order is QoS, T, P.
- $Ae = [\hat{y}_Q, \hat{y}_T, \hat{y}_P]$ is the estimation of system outputs. In practice, we implement the estimator only for the main priority magnitude (e.g. P).

Each task of the controlled application can be implemented with various algorithms or algorithm parameter choices (LCM). Those solutions can be implemented with different hardware modules or software version that may use specific coprocessors (i.e. wired set of instructions). Finally, if a NoC is available, various NoC configurations can extend the solution space.

- $Cs[1..N]$ is the task status register where $Cs[i]$ is the algorithm version for task i . 0 means that the task is not activated, 1 is the simplest version with guaranties regarding the QoS and so on. N is the size of the controlled application task set. $Cs[]$ is exclusively controlled by the LCM.
- $Is[1..N]$ is the current task implementation, $Is[i]$ is the number of the implementation configuration for task i .
- $Ic[1..N]$ is the next task implementation decided by the controller to be handled by the OS.

Reconfiguration space table As previously mentioned, we face two kinds of problems. The first one is the trade-off between accuracy, search space complexity and low cost implementation in the context of embedded systems. The second one is the fluctuating nature of data. Actually for a given configuration, $[QoS, T, P]$ can have important variations depending on the data flows within the system.

To cope with these issues we have implemented a solution based on a sorted reconfiguration space table (RST) where values are regularly updated with online measures or estimations. As indicated in Fig.1, each line L_j characterizes a configuration solution for the system i.e. values affected to the $Ic[i]$ for each i such as $Cs[i] \neq 0$. We note L_c the current RST line.

For each configuration line L_j , $L_j(QoS)$, $L_j(T)$, and $L_j(P)$ values are available. These values are regularly measured from sensors or counters or estimated, as explained in section 3.2. Thus we benefit from a low cost flexible learning system aware of its own behavior in a changeable environment. In a context of multiple controlled applications, multiple tables are maintained.

Regulation parameters Three different rates are used to control regulation :

- R_a is the rate of algorithmic reconfigurations, when measures are not available, estimations are also computed with this rate.
- $R_m \leq R_a$ is the sensor acquisition rate for task execution delays and power consumption for $As[QoS, T, P]$ updates.
- $R_h \leq R_m$ is the hardware reconfiguration rate.

These different rates must be adaptive to consider the tradeoff between the regulation cost and the expected gains provided by reconfiguration choices. This point is detailed in section 3.4.

3 Regulation

3.1 Close-loop modelling

Control theory methodology requires first to settle an analytical model close the real system to be controlled. In our case, the system is composed of a reconfigurable SoC containing an applicative set of tasks that can be implemented with various versions on different HW/SW resources and control, estimation and configuration tasks. Our model, depicted in Fig.2 is based on three elements. R is the control function and S the configuration adaptation. O is the system observer, which provides estimates for the next time slot. The observer implements a system model that is updated when measures are available. In the following we consider P as the controlled magnitude for clarity sake.

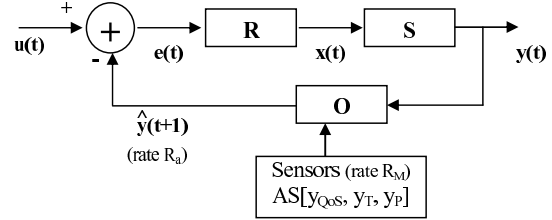


Figure 2: Close-loop model

$u(t)$ is the power reference, which is derived from the user time-life constraint.

$e(t) = u(t) - \hat{y}(t)$ is the difference between the reference and the observer's prediction output i.e. the expected average power consumption of the system in the next time slot.

$x(t) = ke(t)$ is the output of the proportional regulator (R). We don't use the derivative effect, which can lead to sudden output variations with this kind of very fluctuating input nor the integral one since it appears in the system modelling as explained hereafter.

$y(t) = y(t-1) + x(t)$ is the output of the system after the reconfiguration adaptation (S). This function introduce an integrator effect that can in theory annihilate the steady state error.

$\hat{y}(t+1) = a_0y(t) + a_1y(t-1) + a_2y(t-2)$ produces an estimate of the next average power consumption. We detail in 3.2 the estimation model i.e. how the coefficients $\{a_i\}$ are updated.

$y(t)$ is a noisy signal, since the power and delay in the configuration table are approximated. The first error source ϵ_T is the load of non controlled sporadic tasks that can run on the processor simultaneously. The second source is the data dependency ϵ_D , which implies that the execution delay and power consumption fluctuate with the data values. This aspect is particularly true with applications based on video frame analysis or 3D graphics. Thus in practice $\hat{y}(t) = y(t-1) + x(t) + \epsilon_D + \epsilon_T$. Close-loop feedback control suits well for controlling such a changeable environment.

3.2 Online estimation / prediction

The observer regularly updates a model that estimates the system behavior. The aim is firstly to predict the magnitude evolution in order to anticipate the right decision for reconfiguration. Secondly sensors acquisition introduce delay and power overheads when a model-based approach

enables to rapidly estimate the system behavior even when new measures are not available. Basically it enables to tune the tradeoff between accuracy and cost. $y(t)$ can be considered as a noisy signal, the model objective is to predict $y(t+n)$ representing for instance the average power consumption of the system at time $t+n$. Various digital signal processing techniques have been defined to solve this kind of problem. However in the context of embedded systems the aim is to save power, so the control technique overhead must be lower than the expected gains. It means that only low cost solutions can be implemented. For these reasons methods like adaptive least square or Kalman filter are prohibited. The same kind of tradeoff has already been considered in the domain of audio coding (e.g. echo cancellation), where simplified solutions have been successfully implemented. The LMS (Least Median Square) algorithm is equivalent to the recursive least square (RLS) algorithm when the autocorrelation matrix is diagonal. In the context of QoS, P and T signals such a model provides a relevant tradeoff for embedded system modelling. Moreover, the model coefficients are updated at a sampling rate (R_m), which is dynamically modified to tune the cost/accuracy tradeoff. For instance if $R_m = 4R_a$, it means that 3 over 4 $\hat{y}(t)$ values are estimated with constant a_i coefficient without any new acquisition, this downsampling is particularly efficient when the considered magnitude evolution is stabilized. In Fig.3 simulations are given with a 10db SNR ratio, the black (.) line is the reference with a gaussian noise (namely $\epsilon = \epsilon_D + \epsilon_T$), the red (+) line is the estimation of $\hat{y}(t+1)$ with the LMS algorithm and the blue line (+) is obtained with the RLS algorithm with only two taps in order to produce a relatively simple matrix inversion. The dotted lines are the prediction errors. Considering the algorithm complexity for adaptation and estimation (13 mult, 1 div, 9 +/- for a 2 taps RLS and 9 mult, 6 +/- for a 3 taps LMS) and the filter length we have opted for a 3 taps (N_{Taps}) LMS. Under the assumption that ϵ is a gaussian noise, the algorithm stability is guaranteed if K is such as given in eq.2.

$$\begin{aligned} e_p(t) &= y(t) - \hat{y}(t) \\ \forall i \in \{0, 1, 2\} \quad a_i &= K * y(t)y(t-i)e_p(t) \\ \hat{y}(t+1) &= \sum_{i=0}^2 a_i y(t-i) \end{aligned} \quad (1)$$

$$0 < K < \frac{2}{3 * N_{Taps} * \sigma_c^2} \quad (2)$$

3.3 Stability analysis

Equation 2 indicates the stability condition for the estimation algorithm, however the close-loop system still remains potentially instable. Conditions of stability must be guaranteed for all a_i possible values, hereafter we develop the close-loop function in the Z domain and set the conditions of stability for the feedback control.

Open loop function

$$\begin{aligned} G(z) &= (a_0 + a_1 z^{-1} + a_2 z^{-2}) \frac{k}{1 - z^{-1}} \\ G(z) &= \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - z^{-1}} \text{ where } b_i = ka_i \end{aligned} \quad (3)$$

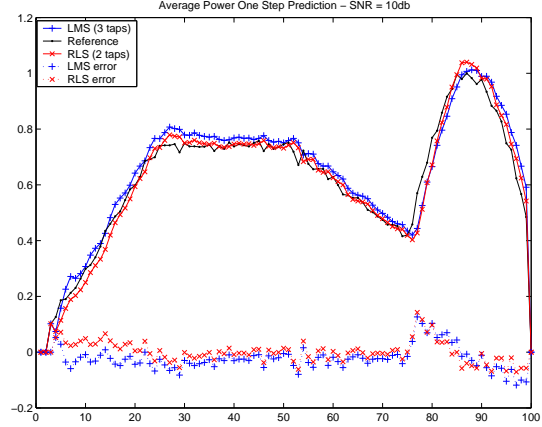


Figure 3: Signal modelling for QoS/P/T prediction.

Close-loop function

$$H(z) = \frac{G(z)}{1 + G(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{D(z)} \quad (4)$$

$$D(z) = (b_0 + 1) + (b_1 - 1)z^{-1} + b_2 z^{-2} \quad (5)$$

From 5 we derive necessary conditions for stability :

$$\text{case 1 : } \quad b_0 = -1; b_1 \neq 1; \quad |b_2| < |1 - b_1| \quad (6)$$

$$\text{case 2 : } \quad b_0 \neq -1; \quad |b_2| < |1 + b_0| \quad (7)$$

3.4 Control of regulation parameters

One of the main issue is the cost of configuration, actually a reconfiguration choice is inefficient if the power and delay gain obtained are lower than power and delay due to the reconfiguration management (observation, decision, configuration processing).

This control is organized in two steps, first the upper bounds of R_a , R_m and R_h are computed off-line, the aim is to compute the maximal values for R_a , R_m and R_h for which power and delay gains can potentially be obtained. Secondly these values are updated dynamically regarding the system behavior.

Two kinds of stability metrics are used for control rate management as described in Algo.1. The first one addresses the system pertinent magnitudes (QoS,T,P) evolution, it is used to adapt the basic control rate R_a namely when the system is stable R_a can slow down. The second one is the prediction error used to control the measurement rate. When the prediction model is correct the R_m can slow down down. Hereafter R_a and R_m controls are based on $y_P(i)$ and $\hat{y}_P(i)$ the average power at time i measured and estimated respectively. We set $R_m = R_a/L$ with $L \leq 1$, this model is applicable identically to QoS and T while respecting $R_m = R_a/L < R_m^{max}$.

3.5 Global configuration control

The global configuration control is described in Algo.2. The inputs are the current configuration and algorithm choices. The algorithm can be decomposed into four parts.

(i) (lines 1-2,15) **Configuration space restriction** (i.e. legal RST lines). The current system status is loaded,

Alg. 1 Dynamic rate management (P case)

 R_a **dynamic management**If $\left| \frac{y_P(i) - y_P(i-1)}{y_P(i-1)} \right| < Th1$ then $R_a \leftarrow R_a/2$ If $\left| \frac{y_P(i) - y_P(i-1)}{y_P(i-1)} \right| > Th2$ then $R_a \leftarrow \max\{R_a^{max}, 2R_a\}$ R_m **dynamic management**If $\left| \frac{\hat{y}_P(i) - y_P(i)}{y_P(i)} \right| < Th3$ then $L \leftarrow 2L$ If $\left| \frac{\hat{y}_P(i) - y_P(i)}{y_P(i)} \right| > Th4$ then $L \leftarrow \min\{1, L/2\}$ $Th1 < Th2 < Th3 < Th4$ are thresholds to be set with regards to accuracy vs cpu load tradeoff.

$Cs[1..N]$ provides the configuration controller with local decisions that delimit the configuration space. Moreover, at R_h rate, moves between RST including HW reconfiguration lines are authorized (line 15).

(ii) (lines 6-13,31) **Configuration characteristics update** At R_m sensor acquisition rate the LMS coefficients are updated and errors (e_{RST}) between the QoS,P and T values in L_c RST line and real values are computed. The control rates are updated.

(iii) (line 17-30) **Configuration decision**, $\hat{y}(t+1)$ prediction is computed to obtain the error $e(t)$ between the references and the expected values. QoS, P and T estimator can possibly be implemented, however we made the choice of using the estimator for the first priority magnitude for simplicity reason. In Algo.2 the priority is the power reference (i.e. time-life constraint). The configuration decision is implemented with fast sort algorithms. Firstly a \mathcal{L}_0 set is built with configurations that guaranty the power constraint at $t+1$. The RST errors are only known for the current L_c configuration. Without any other information and to avoid complex estimation for the whole configuration space, we assume that these errors, due to data dependency and non controlled sporadic tasks, are the same for all configurations ($L_i \neq L_c$). This assumption fits also for the concept of a learning system where data are regularly updated. Then a second set \mathcal{L}_1 is built with \mathcal{L}_0 configurations respecting the second priority (QoS in Algo.2) and finally a third set \mathcal{L}_2 contains configurations according to the last priority (T in Algo.2). Finally the best solution is selected and $Ic[1..N]$ is updated.

(iv) **OS controled configuration** $Ic[1..N]$ is the next configuration computed by the decision task. The configuration process is handled by the OS for synchronisation and data integrity purpose, it is applied when the application is idle.

4 Object tracking case study

4.1 Overview

The object tracking (OT) application (see table in Fig.4) is designed for an embedded smart camera. We are currently implementing it on a Stratix platform enhanced with different peripherals : battery gauge, camera and VGA controller. This platform has been selected for the NIOS configuration facilities (SOPC) and the availability of μ COSII that can be modified to integrate control API described in 2.1. Altera technology unfortunately doesn't provide dynamic reconfiguration capabilities, however for proving the concept of our feedback configuration control this as-

Alg. 2 Global control algorithm, case Power priority.

```

1 Inputs :  $Cs[1..N], Is[1..N]$ ,
2    $As[QoS, T, P], Ap[QoS, T, P], L_c$ ,
3    $\forall i = \{P, T, QoS\} e_{RST}(i) = 0$  %% RST errors
4 For each ( $t = nT_a$ ),  $n \in \mathbb{N}$  do %% adaptation base rate
5   If ( $t = mT_m$ ),  $m \in \mathbb{N}$  %% measures available
6     Update  $As[y_{QoS}, y_T, y_P]$ 
7      $e_p(t) = y(t) - \hat{y}(t)$ 
8     Model  $\{a_i\}$  LMS adaptation
9      $R_a$  dynamic management Algo.1
10     $R_m$  dynamic management Algo.1
11     $e_{RST}(P) = y_P(t) - L_c(P)$ 
12     $e_{RST}(T) = y_T(t) - L_c(T)$ 
13     $e_{RST}(Q) = y_{QoS}(t) - L_c(QoS)$ 
14  If ( $t = pT_h$ ),  $p \in \mathbb{N}$ 
15    HW reconfiguration allowed from  $Is[]$  to  $Ic[]$ 
16  %%% Configuration decision; power priority
17   $\hat{y}_P(t+1)$  computation
18   $e(t) = u_P(t) - \hat{y}(t+1)$ 
19  %% system output error
20  Sort RST according to  $Ap[QoS, T, P]$  priorities
21  %% for simplicity we assume PQoS order :  $Ap = [2, 1, 0]$ 
22   $\mathcal{L}_0 = \{RST L_i | L_i(P) + e_{RST}(P) + e(t) < u_P(t)\}$ 
23  If  $\mathcal{L}_0 \neq \emptyset$ 
24     $\mathcal{L}_1 = \{L_i \in \mathcal{L}_0 | L_i(QoS) + e_{RST}(Q) < u_Q(t)\}$ 
25  If  $\mathcal{L}_1 \neq \emptyset$ 
26     $\mathcal{L}_2 = \{L_i \in \mathcal{L}_1 | L_i(T) + e_{RST}(T) < u_T(t)\}$ 
27  If  $\mathcal{L}_2 \neq \emptyset$   $Ic[] \leftarrow 1^{st}$   $\mathcal{L}_2$  wrt increasing T
28  Else  $Ic[] \leftarrow 1^{st}$  line from  $\mathcal{L}_1$  wrt increasing QoS
29  Else  $Ic[] \leftarrow 1^{st}$  line from  $\mathcal{L}_0$  wrt increasing P
30  Else  $Ic[] \leftarrow 1^{st}$  line from RST wrt increasing P
31  Update line  $L_c$  in RST with  $As[QoS, T, P]$ 
32 Output :  $Ic[1..N]$ 

```

pect can be emulated. This application fetches numerous sources of uncertainty from data like light variations, object count, shape and speed, and the number of white pixels after threshold processing. We have also added uncontrolled sporadic tasks with random release times.

OT is an image processing balanced with a Kalman prediction filter for object location. When a new frame is available, OT first performs an average from 2,3 or 4 precedent images and suppresses the background before comparing the result to a threshold. The threshold value can be computed with an adaptive or static threshold. The resulted binary image contains white objects in motion on a black background. An erosion dilatation processing is computed followed by a reconstruction which can be done or not depending on a QoS / performance trade-off. Finally a task labels the detected objects and computes their center of gravity. Based on their position, two tasks manage and include the static objects in the background image. The results are displayed on a VGA screen according to the user choice (camera image, binary objects, with or without gravity center marks). Finally, the two last tasks are the Kalman prediction and QoS calculation processing. This QoS is the result of the variation between predictions and measures. When the QoS is greater than a given threshold, the image acquisition and processing are not launched but replaced by Kalman predictions.

The video rate and image size can be adapted (1) regarding the QoS, which is the relative location prediction error. Conjointly different other combinations of configurations can be adopted: the number of frames for the average frame computation (2), the use of adaptive or fixed

threshold (3) and the Kalman filter adaptation (4). The configuration space includes tens of HW/SW implementations offering various parallelism exploitation solutions (5) according to different algorithm configurations. Choices (2-4) are local decisions based on specific information while (1,5) are global and based on decisions and QoS computations from LCM. Fig.4 gives the OT local configurations and metrics.

4.2 Example of configuration schemes

We discuss three kinds of configuration schemes in the context of the OT application.

Let's consider first the Byte2bit and the erosion-dilatation tasks that compute the number of white pixels after the threshold and after the erosion processing respectively. The erosion processing suppresses the isolated white pixels from the threshold image. Those metrics represent the image noise. Then, the LCM retrieves and compares the metrics with a threshold and provides the new configuration to the GCM. For instance, the Byte2bit task is in the "two images averaging" configurations, the image is noisy, the difference of metrics are greater than the threshold. The new Byte2bit configuration is set to the "three images averaging". The LCM provides also the QoS value from the QoS tasks to the GCM. In the RST table all configurations that allowing "three images averaging" are valid. Finally, the GCM sorts and selects the best future configuration according to the system parameters priority. The new configuration is then sent to the tasks by the OS. The Byte2bit configuration decision needs metrics from two tasks.

The next example describes the configuration of one task from the metric of a second task. The Adaptive Threshold task depends on the labeling task metric. This task provides the LCM with the number of detected objects. The LCM compares it with the previous value. The number of detected objects may increase abnormally due to a bad threshold value. It's then important to compute a new one. Consequently, the GCM selects the best configuration among the valid configurations that allow the Adaptative Thresholding processing.

Those two previous examples take care of task metrics to configure a task. Another case of configuration is given by the LCM internal timer to select the measure or prediction processing. It provides regularly the GCM with the desired configuration of prediction. In this configuration, all tasks have the "bypass" configuration except the Kalman task which processes the prediction.

5 Conclusion

This paper presents a new approach for a self-reconfiguration control in the context of embedded reconfigurable systems with fluctuating working conditions. We propose a generic feedback control modelling with stability conditions, the magnitudes considered are QoS, Power and Real-Time constraints according to a priority order fixed by the user. Based on a learning system with prediction capabilities, our global control algorithm offers a completely tunable tradeoff between accuracy and complexity, moreover it enables to handle local and global configuration decisions. The smart camera implementation is progressing on Altera Stratix, we conjointly study the implementation on a Xilinx Virtex II to get online dynamic reconfiguration.

Task ID	Task Name	Function	Configuration	Metrics
T0	Image acquisition	acquisition of the actual image from an external SDRAM	bypass	
T1	Byte2bit	Averaging, background suppression and thresholding processing	bypass number of image to average (0,2,3only,SM or 4)	Number of white pixels after the threshold processing
T2	Erosion and dilatation	erosion and dilatation processing	bypass	Number of white pixels after the erosion processing
T3	Reconstruction	reconstruction processing	bypass	no change after the first reconstruction processing
T4	Label and center	Objects labelling and calculation of centers of gravity	bypass	Number of detected objects
T5	Adaptive Thresholding	Calculate the threshold value	bypass	Threshold variation
T6	Objects mobility	Calculate if the detected objects have moved	bypass	Yes/ No
T7	Background update	Updating the background image	bypass	
T8	Image display	display the actual image or the objects with or without center of gravity marks	bypass cam, center-of-gravity marks, object-center of gravity marks	
T9	Kalman	Prediction of the centers of gravity positions or coefficient update	prediction of coefficient update	
T10	QoS	Calculate the QoS value from the predictions and the measures	bypass	QoS

Figure 4: Object tracking local parameters

References

- [1] J.L.Wong, G.Qu, and M.Potkonjak, "An on-line approach for power minimization in qos sensitive systems," in *ASP-DAC*, 2003.
- [2] S.Manne and A.Klauser D.Grunwald, "Pipeline gating: speculation control for energy reduction," in *25th Int. Symp. on Computer Architecture*, Spain, 1998, pp. 132-141.
- [3] D.H.Albonesi, "Selective cache ways: On-demand cache resource allocation," in *32nd Annual International Symposium on Microarchitecture*, 1999.
- [4] R.Maro, Y.Bai, and R.I.Bahar, "Dynamically reconfiguring processor resources to reduce power consumption in high-performance processors," in *Work. on Power-Aware Computer Systems*, 2000.
- [5] V.Nollet, T.Marescaux, D.Verkest, J-Y.Mignolet, and S.Vernalde, "Operating-system controlled network on chip," in *41th ACM/IEEE Design Automation Conf.*, USA, 2004.
- [6] J.Liang, A.Laffely, S.Srinivasan, and R.Tessier, "An architecture and compiler for scalable on-chip communication," *IEEE Trans. on VLSI Systems*, vol. 12, no. 7, pp. 711-726, July 2004.
- [7] J-Y.Mignolet, V.Nollet, P.Coene, D.Verkest, S.Vernalde, and R.Lauwereins, "Infrastructure for design and management of relocatable tasks in a heterogeneous reconfigurable system-on-chip," in *Design, Automation and Test in Europe conf. (DATE)*, Munich, Germany, Mar. 2003.
- [8] H.Walder and M.Platzner, "Reconfigurable hardware operating systems: From design concepts to realizations," in *Int. Conf. on Engineering of Reconfigurable Systems and Algorithms, ERSA'03*, Las Vegas, USA, June 2003.
- [9] B.Noble, M.Satyanarayanan, D.Narayanan, J.E.Tilton, J.Flinn, and K.R.Walker, "Agile application-aware adaptation for mobility," in *ACM Symp. on Operating Systems Principles*, 1997.
- [10] C.Lu, J.Stankovic, G.Tao, and S.Son, "Feedback control real-time scheduling: Framework, modeling and algorithm," *Jour. of Real-Time Systems*, vol. 23, no. 1/2, pp. 85-126, jul./sep. 2002.
- [11] B.Li and K.Nahrstedt, "A control-based middleware framework for quality of service adaptation," *IEEE Journal on Selected Areas in Communication*, Sept. 1999.
- [12] TI, "Gas gauge bq 2084," <http://www.ti-estore.com/>.