行政院國家科學委員會專題研究計畫　期中進度報告

子計畫二：多媒體通訊數位基頻 SoC 加速架構及嵌入式作業系統界面的研究(2/3)

計畫類別： 整合型計畫
計畫編號： NSC93-2220-E-009-008-
執行期間： 93 年 08 月 01 日至 94 年 07 月 31 日
執行單位： 國立交通大學資訊工程學系(所)

計畫主持人： 蔡淳仁

計畫參與人員： 王岳宜、游雅惠、邱正男、王志鵬、楊植峻

報告類型： 完整報告

處理方式： 本計畫可公開查詢

中　華　民　國 94 年 10 月 31 日

行政院國家科學委員會專題研究計畫成果報告

MPEG-4/21 SOC 設計及新世代行動通訊之研究-
子計畫二：多媒體通訊數位基頻 SoC 加速架構及嵌入式作業系統界面的研究(2/3)

計畫編號：NSC 93–2220–E–009–008
執行期限：自民國 93 年 08 月 01 日起至 94 年 07 月 31 日止
主持人：蔡淳仁　教授　國立交通大學資訊工程系所
共同主持人：
計畫參與人員：王岳宜、游雅惠、邱正男、王志鵬、楊植峻
國立交通大學資訊工程系所

一、中文摘要

關鍵詞： 多媒體壓縮、多媒體通訊、嵌入式作業系統、單晶片系統、數位基頻晶片

今年度本計畫延續上一年度的計畫，除了針對去年發展的 H.264 視訊壓縮移動量偵測模組進行改進外，並增加了 CABAC 模組的設計。同樣是架構在去年所發展的多媒體視訊加速器平台之上。在 H.264 視訊壓縮移動量偵測模組改進方面，最主要是引進了一個全新的觀念：在進行移動量偵測（ME）時所用到的 sub-pixel interpolator 不一定要使用標準制定的 interpolator，這樣的設計有許多好處：首先，我們可以在 ME 時用一個較簡單的 interpolator 以節省記憶量。另外，我們可以設計一個 on-the-fly 的 interpolator，如此可以大量節省 on-chip memory 的大小。只要在最後找出最佳移動量時再用符合標準的 interpolator 計算 error residual，產生的 bitstream 就可以完全符合標準。這部份的研究已發表在 IEEE ISCAS' 05 [1]。在 CABAC 的部份，目前已經完成 C-Model 的設計和硬體架構的大體設計，RTL coding 則還在進行中。

另外，今年新增的一個研究重點是嵌入式作業系統的設計。隨著無線網路與手機的快速發展，未來的頻寬與使用著的需求越來越高，多媒體手機的應用往往需要大量的運算，因此單一處理器的系統無法有效支援這些需求。

通常這類平台的處理核心都是非同質性（heterogeneous）的，在單一晶片上整合了一個一般目的處理核心（General-Purpose Processors），如 ARM、MIPS 等，及一或數顆專門用來處理多媒體任務龐大運算的處理核心，如數位訊號處理器（DSP）、可程式化的硬體加速器（FPGA）、或者是特定對象的硬體加速器（如 H.264 video accelerator）。我們特別針對異質性雙核心平台設計了一個程序排程器，可以動態根據 RISC core 和 DSP core 的負荷，來決定要叫用那一個版本的執行碼（RISC 或 DSP）來完成工作。

另外，在 tightly-coupled 的應用上，我們延續去年的計畫，繼續開發 dual-core video encoder，今年的重點則是 H.264。

最後，我們也針對 MPEG-21 的 Scalable Video Coding (SVC) rate-control mechanism 進行研究，不過由於過去一年來，在 MPEG 組織中所進行的 SVC 的發展有需多變化，目前 MPEG 已經把這方面的標準制定移到 MPEG-4 Part 10，採用的技術也不是當初預期的 Wavelet-based approach，不過，我們仍舊把我們所設計適合 multiple-adaptation 的 wavelet-based rate-control mechanism 發表在 IEEE ISCAS' 2005 [2]。

Abstract

Keywords：multimedia compression, multimedia communication, embedded OS, System-on-Chip, digital baseband processor

This report summarizes the progress of the second year project on the accelerator architecture for a multimedia media communication digital baseband processor. Two major tasks are covered during the second year of the subproject. First of all, the design of H.264 sub-pixel motion estimator and CABAC hardware logic are executed. Secondly, the implementation of a tightly-coupled AMP scheduling kernel plug-in for embedded Linux OS for the TI OMAP 5912 platform. In addition, we also studied the rate control mechanism for a wavelet-based video codec and proposed a new algorithm for multiple-adaptation applications.

二、緣由與目的

For our first year project, we have implemented an SoC platform for video codec accelerators. We have also designed an H.264 in-loop filter logic and an integer pixel motion estimator. This year, we continue with the improvement of the SoC platform. In addition to revise the motion estimator so that sub-pixel accuracy can be handled, we also begin with the design of a H.264 CABAC hardware logic since this is typically regarded as one of the most complex module for H.264 main profile.

The most innovator part of our sub-pixel motion estimator design is that the interpolator used for sub-pixel motion estimation is different from the interpolator for error-residual coding. By doing so, we can significantly save both memory and computation complexity with very little performance degradations. Another major work done in the second year project is to design a OS kernel scheduler which realized a tightly-coupled programming model for heterogeneous multi-core architecture. Traditionally, for heterogeneous multi-core platforms, the application are developed using a loosely-coupled approach. That is, the programmer statically partitions the application into RISC code and DSP code. As soon as the partition is done, it cannot be repartitioned at runtime. Although this is optimal for a single task, the approach losses it optimality when there are many independently partitioned tasks executing simultaneously at runtime. We have proposed a new tightly-coupled programming model that enables runtime dynamic task partitioning. We have also developed a OS kernel scheduler based on embedded Linux to support this model.

Finally, we have studied a rate control algorithm for wavelet-based video codec to efficiently achieve rate-distortion (RD) optimality for multiple-adaptations. The RD side information is summarized using a cubic model in order to reduce bitstream size and, at the same time, speed up the optimal bitstream truncation point search process. Although, in the past year, MPEG has decided to use DCT-based FGS coding for its new scalable video coding standard, the proposed wavelet rate control algorithm is based on the general RD theory and can be adapted to other frameworks.

三、結果與討論、計畫成果自評

The second year project continues with the work from the first year project on H.264 codec accelerator architecture for multimedia handsets and OS support for heterogeneous multi-core architecture. In this intermediate report, we highlight our research in these two directions.

3.1 H.264 Dual-Interpolator ME

Block matching motion estimation (ME) has been widely adopted by many video codecs such as MPEG-1/2/4, H.264 and WMV9 to eliminate the temporal redundancy among motion pictures[1][4]. Due to its high complexity nature, fast ME scheme has always been the focus of efficient encoder implementation. With the advances of new video codecs, highly sophisticated motion-compensation (MC) model has been

adopted. For example, in H.264, quarter-pel MC can improve the coding efficiency by 30% at the expense of large amount of memory and/or computational complexity on the encoder side[8][9].

A typical fast ME implementation computes interpolated images (requires large amount of memory) before ME. During the ME loop, it only searches the best candidate among integer-pixel search positions and then refines the motion vector to sub-pixel accuracy around the best integer-pixel candidate. Obviously, this technique limits the efficiency of sub-pixel MC. In addition to large memory storage size, it may also requires more arithmetic operations than necessary since only a small portion of the complete interpolated images will be searched for sub-pixel motion vectors. In short, computation of the complete interpolated image may not be the best policy for sub-pixel ME.

Another complexity/quality dilemma of a high quality codec is the computation of ME measure. It is suggested that using a transformed Sum-of-Absolute-Differences (SAD), instead of SAD, as the measure for ME gives better coding efficiency since transformed SAD is a better approximation of the entropy of the residual block [1] than SAD. Hadamard transform is adopted in H.264 for this purpose (as a simple approximation to DCT) [5][7]. However, the extra computational complexity increases the difficulty for real-time video coding systems.

In order to save the memory space used to store the interpolated images and the overhead to compute them, it makes sense to perform on-the-fly sub-pixel interpolation. However, the long-tap filters commonly used in advanced codecs may make this approach too computationally expensive for ME. A key observation in the proposed scheme is that the sub-pixel interpolator used in the ME-loop does not have to be the same as the interpolator used for coding. The later must be standard-compliant while the former can be designed arbitrarily. This paper proposes an efficient method of on-the-fly sub-pixel ME with a joint architecture for Hadamard transformed-SAD computation. The design can be easily mapped to VLSI architecture.

3.1.1 Proposed Dual-interpolator ME

In this section, the proposed on-the-fly sub-pixel ME with a joint architecture for Hadamard transformed-SAD computation is presented.

A. Application of two different interpolators for ME and coding

Sophisticated sub-pixel interpolation is adopted by modern video coding standards to reduce error residuals of predictive coder. Take H.264 as an example. 6-tap and bilinear filters are applied to the video data to generate video samples at half-pixel and quarter-pixel positions, respectively. In our proposed method, different interpolation filters are used for ME and for coding. During the ME process, an on-the-fly bilinear filter is used to generate the sub-pixel samples (including half-pel and quarter-pel samples). The filter can be activated at every candidate (integer-pixel) search positions. Therefore, this proposed scheme has less chance of missing a sub-pixel target than the conventional approach (namely, search for sub-pixel target only at the best integer-pixel position). After the best motion vector is determined for certain macroblock, an on-the-fly standard compliant interpolation filter is used for MC coding. Even though this filter is more expensive, we only compute it once for each motion vector. Therefore, the overall interpolation computations should be less than the convention method of pre-computation of the entire interpolated image. Furthermore, it does not require extra storage space for interpolated images.

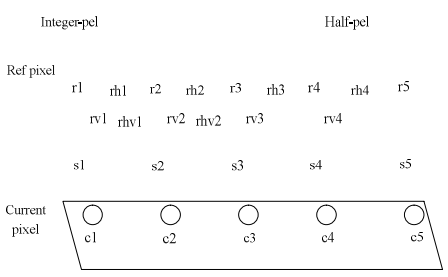


Figure 1. Bilinear interpolation of 1/2-pel reference

Figure 1 shows the bilinear interpolation by one-dimension linear weighted interpolation among the reference pixels. Notation cn's are the integer current pixels while rn's are the upper row of reference pixels and sn's are the next row of reference pixels. The horizontal, vertical, diagonal half-pel pixels are notated as rhn's, rvn's, and rhvn's respectively. Eq. (1) and Eq. (2) are the linear weighted interpolation equations. The notation, ω1 and ω2 are the weightings (distance ratio) between the two integer-pel and half-pel points. As well, the quarter-pel can also be computed by two integer pixels with different weighting value, ω1 and ω2.

$$\omega_1 \cdot \alpha + \omega_2 \cdot \beta = \gamma \qquad (1)$$

$$\omega_1+\omega_2 = 1,\ 0 \leqq \omega_1 \leqq 1,\ 0 \leqq \omega_2 \leqq 1 \qquad (2)$$

Eq. (3) calculates the difference between a current pixel and the reference bilinear-interpolated half-pixel. With this method, either half-pel or quarter-pel samples can be computed from integer sample points.

$$\Delta\rho\eta 1 = \chi_1 - \rho\eta_1 = [2\chi_1 - (\rho_1+\rho_2)] / 2 \qquad (3)$$

B. SATD RSME

Hadamard transform performs simple bit estimation generated after transform. SATD-ME is to use the SAD of the Hadamard transform (HT) coefficients as a matching measurement. It's found that DCT and HT were almost identical in energy compaction although the HT with no multiplication has a much lower computational complexity. Therefore, HT can be used to estimate the bits for coding. On the other hand, the access and computational complexity are largely required than using SAD as ME cost function. In Fig. 2, the circled (boxed) expressions can be computed by dedicated data paths of VLSI implementation.

C. Combining operations of sub-pel ME and SATD-ME

The operations of sub-pixel ME and SATD-ME can be combined and reorganized. The formulas shown in Figure 2 are the derivation of half-pel bilinear interpolation and Hadamard transforms in horizontal direction for one 4x4 sub-block. On the other hand, three half-pel bilinear candidates, H, V, and HV at half-pel positions accompanied with each integer-pel candidate are generated to calculate the residual SATD and compared with each other during the motion search. By observing the formula, the arithmetic operations with blocks are identical. Thus, they can be easily mapped to h/w design.

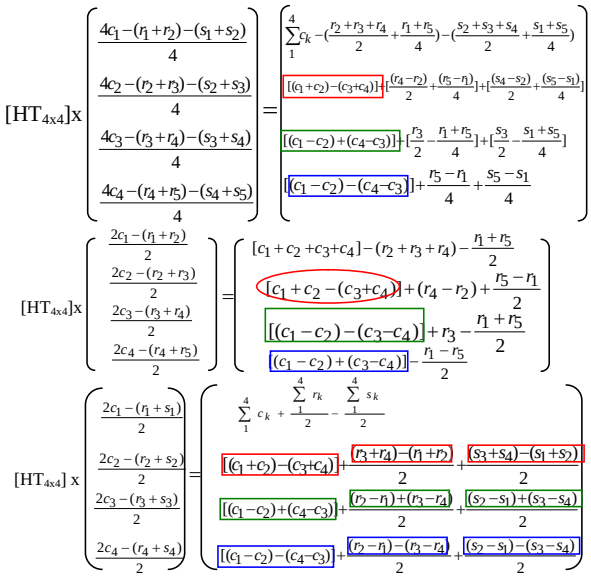


Figure 2. Formulas of three subpixel, H, V, and HV's Hadamard Transform matrice.

3.1.2 The Proposed VLSI Architecture

Due to the increased complexity and variations of multimedia systems and standards, many platform designs today adopt a hardware/software co-design approach. In this framework, instead of hard-wiring the complete system, integrated hardware accelerators (or IPs in an SoC) are used to assist coding tasks. Since ME occupies major portion of computation load among coding tools, a h/w design is presented in this section for the proposed method of dual-interpolator/sub-pixel ME to implement H.264 video coding.

A. Overall architecture

Figure 3 shows the overall architecture for the proposed bilinear-interpolated SATD ME. In the figure, BINTSATD AGU block generates the corresponding address to access current MB and reference data. There are two

register arrays for saving 4×4 current MB and n×n search window reference data. HTCtl controls the flow of HT computation and the input/output of two buffers. Before storing corresponding reference pixels into its buffer, the Aligner aligns the pixels from search window memory. The bilinear-interpolated sub-pixels are calculated by subpel_gen block while the accurate standard-complaint interpolated sub-pixels are generated by codingsubpelgen block. Both of the two blocks generate the pixels in Half-H, Half-V, and Half-HV positions respectively. The 2-D HT block calculates the Hadamard transformed 4x4 data from the result of differences between current and updated reference pixels. After the transformed data is calculated by MBSATD, SATDcomp block compares the best SATD result among integer-pel and sub-pels and also stores the result of SATD to acquire the final summation SATD value of the m×n-block. Based on the final SATD result, not only the MV for the block can be determined but also the address generator will determine the next address of sample points. Please note that this architecture is a general design for the idea. Additional hardware logic such as adding search window space, and other main blocks paralleled might be needed while timing requirement can not be met on implementing real-time video coding system.

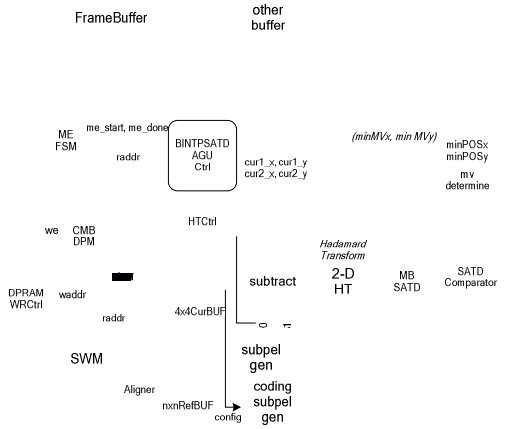
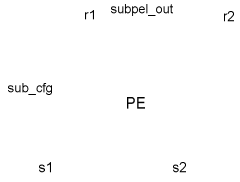


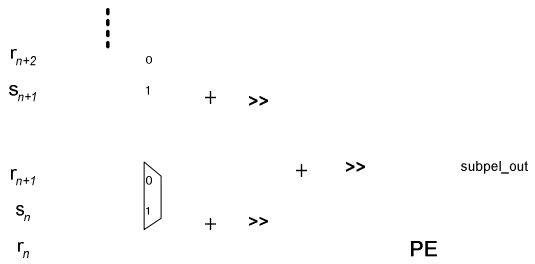
Figure 3. Formulas of three subpixel, H, V, and HV's Hadamard Transform matrice.

B. Interpolators Design

The major idea of our proposed method is to adopt two different interpolators for motion search and coding, respectively. Fig. 4 describes one unit of processing element (PE) in the bilinear interpolator on reference pixels. Each PE can generate different Half-H, Half-V, and Half-HV pixels which the output, subpel_out, is controlled (MUXed) by config from HTCtrl block. The second interpolator, the 6-tap plus bilinear filters used in H.264, can be done either in software or hardware. Because this complex interpolator is only used after the motion vector has been determined, it is actually an exact duplication of the code (logic) used in a decoder where on-the-fly computation is often used without large memory buffers for the complete interpolated images.



(a) sub-pel PE



(b) Logic design of one PE

Figure 4. Bilinear-interpolated PE design

3.1.3 Experimental Results

Software simulation of the performance of the proposed algorithm is presented in this section. In this paper, we use H.264 to evaluate the performance. There are many fast ME algorithms. Here we adopt a complexity-reduced ME, RSME [12]. The encoder is based on H.264 reference software JM 7.3 with RD optimization turned on, three reference frames, IPPP… coding pattern, and CAVLC. The standard test sequences FOREMAN, STEFAN, and MOBILE are used to show the quality performance of the proposed algorithms. All

sequences are in CIF resolution, 30 fps. Motion search range is ±32, and all sub-block types are used. In the ME, a step size of 8 is used for the TWSS step. Only the best candidate selected by the TWSS module is selected for the next level TSS refinement. The step sizes for TSS are 3, 2, and 1 respectively for the 1st, 2nd, and 3rd steps. Fig. 5, Fig. 6, and Fig. 7 show the quality performance on the proposed ME vs. FSME. From the results, we can see there is little degradation on quality while less than 4% on average for bit rate increase comparing the proposed method with FSME depicted in Table I. However, the timing required by the proposed method can be largely reduced than FSME.

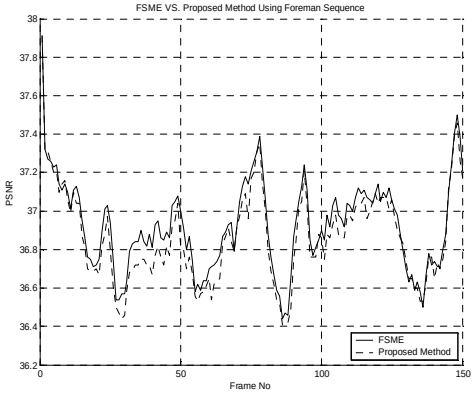


Figure 5. PSNR Plot of FOREMAN

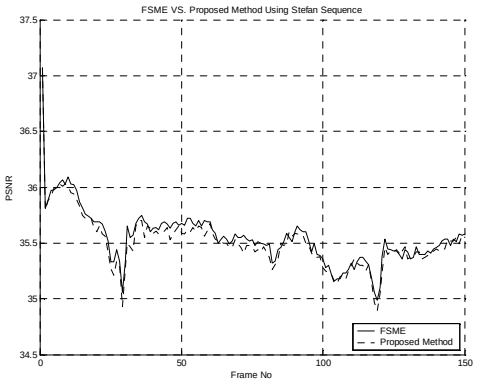


Figure 6. PSNR Plot of STEFAN

Video Sequence	Performance Analysis			
	Average Bit Rate (Kbps) (%)		*Average Quality (PSNR) (%)*	
	FSME	***Proposed ME***	***FSME***	***Proposed ME***
Foreman	342.98 (100%)	359.39 (**104.7**%)	36.92 (100%)	36.86 (**99.8**%)
Stefan	1016.0 (100%)	1054.3 (**103.7**%)	35.55 (100%)	35.51 (**99.9**%)
Mobile	1455.28 (100%)	1506.55 (**103.5**%)	34.07 (100%)	34.02 (**99.9**%)

Table I. Average RD Table

3.2. Unified AMP OS Kernel Design

The complexity of embedded system grows rapidly due to new mobile multimedia applications. Uni-processor platforms are not suitable for these applications since they require high core frequency in order to handle massive multimedia data processing tasks. However, higher core frequency consumes more power and produces more heat, which is inapt for small form factor embedded systems. Therefore, a common practice for mobile devices is to adopt multiprocessor solutions to increase system performance.

In particular, asymmetric multiprocessor architecture has been widely used for embedded systems development (for example, for cell phones). In this architecture, a general purpose RISC processor (GPP) core and a digital signal processor (DSP) core are integrated into a system-on-chip (SoC), which can handle embedded system tasks efficiently, especially for multimedia applications. However, existing real-time operating systems for such architecture typically adopt a loosely-coupled approach. Task partitions between the two cores are typically done offline and two separate schedulers are employed to perform task scheduling for the two cores independently. This paradigm works properly for traditional mobile applications where the GPP core are typically slow and functionally limited and the application tasks can be put into a simple foreground/background working model.

New generations of multimedia applications and devices make this kind of loosely-coupled system design obsolete. There are at leas three reasons that call for a new approach for real-time scheduler designs. First of all, new GPPs today are much more powerful than old ones. Many of them even include special instructions for DSP tasks. Secondly, multimedia applications has become so complicated and dynamic that run-time load balance between the GPP core and the DSP core are crucial for system performance and power consumption

reduction. Thirdly, many multimedia applications are more memory-centric than computation-centric. In addition, many embedded systems use distributed memory banks to increase memory bandwidth. Therefore, when inter-processor communication overhead is taken into account, optimal task assignment can only be done at run-time, instead of offline.

3.2.1 Previous Work

There are many researches on schedulers for symmetric multiprocessor (SMP) architecture in last twenty years. A symmetric multiprocessor system can provide better overall system performance than a uniprocessor system [18]. With the gaining popularity of multimedia devices in recent years, the focus has been shifted to asymmetric multiprocessor (AMP) systems. The main reason why AMP systems are used for embedded devices is because that they provide the best performance/clock ratio for the execution of a wide variety of tasks.

Wendorf et al. [16] proposed a number of scheduling policies, ranging from asymmetric master/slave scheduling to symmetric scheduling, for multiprocessor platforms. According to their experiments, "OS Preempt" policy provides the best performance in almost all situations for AMP systems. Moreover, an AMP system using the OS Preempt scheduling policy can perform as good as a fully symmetric system. Their results also indicate that the overhead of context switching and shared resource contention in asymmetric systems are relatively minor factors in overall system performance.

A simple model of master/slave architecture is presented by Greenberg and Wright in [13] along with two scheduling algorithms. In this proposal, a subset of the system calls, which are referred to as the kernel calls, can only be executed on the master. The remaining system calls are referred to as the user calls. When a slave process makes a kernel call, the slave processor returns the process to the master, rather than services the call by itself. The kernel calls are serialized and may not be independent since these calls may update data that influence the whole system. In the proposed design, jobs not running on any processors are waiting in one of the two queues, the master queue or the slave queue. Jobs in the master queue are all in kernel mode and jobs in the slave queue are all in user mode. A slave processor can take jobs from the slave queue only and the master processor can take jobs from either queue. Two scheduling algorithms are proposed to balance between queue-switching overhead reduction and scheduling flexibility. They also proposed a way to find P*, the optimal number of slave processors in a single-master processor environment.

In [14], Avritzer et al. developed an analytical performance modeling approach for load sharing policies in highly asymmetric systems that schedule jobs based on global system state. In the system described in [14], hosts have many different speeds which are subject to heterogeneous workloads. They also introduced a threshold type load-sharing algorithm for distributed asymmetric systems, the algorithm varies the thresholds dynamically, adjusting them to the load in order to keep an optimal number of tasks in each hosts. In this paper, they modeled the job routing algorithms by building a global state Markov chain and computing upper and lower bounds on the total system average delay. They concluded that carefully tuned algorithms for load sharing in the asymmetric environment provide a significant improvement in performance over simpler algorithms.

For cooperation between resources, Saewong and Rajkumar [22] proposed the use of a Cooperative Scheduling Server (CSS), which is a dedicated server that manages one specific controlled resource while using a controlling resource, to control multiple resources access from a single CSS. A CSS is created on a controlling resource (such as a CPU) to handle all local requests for a controlled resource (such as disk access). The CSS reserves a sufficient amount of capacity for controlling resources as needed to fulfill the obligations it has for accessing controlled resources. Because there are

scheduling policies for both controlling and controlled resources, co-scheduling design must be employed. Some important considerations of the co-scheduling design in [22] are as follows: 1) scheduling mismatch due to heterogeneity of resource scheduling policies, 2) conjunctive admission control, 3) resource synchronization, and 4) efficient resource utilization.

For embedded multimedia applications, such as 3G mobile phones, both control operations and massive data processing operations are very important. There are some architecture proposals ([17][20]) efficiently integrate these two different types of computing units into one AMP SoC. However, most of these systems are designed in a loosely-coupled manner. For example, in [21], they discussed the problem of multiprocessors scheduling for asymmetric architectures composed by a general purpose processor (GPP) and a digital signal processor (DSP). Two task queues are used in their design, one for regular tasks (for GPP) and the other for DSP tasks. When the DSP is idle, the scheduler always selects the task with higher priority between the tasks at the head of the two queues. When the DSP is active, the scheduler only selects the highest priority task from the regular queue.

3.2.2 The Proposed AMP Scheduler

The key concept of the proposed AMP scheduler is to facilitate a tightly-coupled working model. Without loss of generality, assume that there is a\one GPP core and one DSP core in the target system. In the tightly-coupled model, a task can be assigned to either the GPP or DSP at runtime. When a new task arrives, the unified scheduler will oversee the runtime status of both processor cores and decide which core is more suitable of executing the new task. In our design, the scheduler computes a cost function based on power consumption, computation complexity, deadline fulfillness, and loading balance in order to make a decision for task dispatching.

Since different processor cores execute different binaries, to enable the proposed tightly-couple model, a new programming practice must be adopted. The new programming model is somewhat similar to single thread vs. multi-thread programming. In the OS, new system service calls are provided for the application to register dual-core versions of executable images into the kernel at runtime. Note that registration of a dual-core executable image does not create a task and enter it into the task queues. Another API must be called explicitly to start a (dual-core) task, which will enter the single universal task queue. This is similar to explicitly calling a system service to start a new thread of a process. The unified scheduler will then dispatch the task either to the GPP or the DSP based on a cost function.

Figure 7 is the proposed AMP scheduler. It is composed of a cost function evaluator, an AMP scheduler, a version registrar, a resource monitor, and the task interface.

The AMP scheduler dispatches a task based on the cost function value and manages running tasks, the version registrar records available executable images (refer to as services in this paper) in the GPP version table and the DSP version table, and the resource monitor watches GPP-side and DSP-side status and provides information for the cost function evaluator.

The task interface is an interface between the proposed AMP scheduler and the processing cores. In the proposed design, there are three types of task nodes, the GPP task node, the DSP task node, and the system task node. The GPP task node provides APIs for managing tasks running on the GPP, the DSP task node provides APIs for managing tasks running on the DSP, and the system task node provides APIs for retrieving and monitoring system status.

Eq. (4) is the cost function used by the proposed scheduler to choose the target processor for a task.

$$C = \lambda_1 C_{power} + \lambda_2 C_{comp} + \lambda_3 C_{deadline} + \lambda_4 C_{load} \quad (1)$$

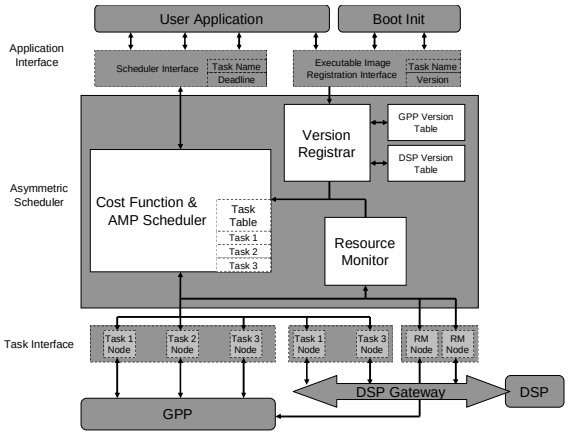


Figure 7. AMP Scheduler Architecture

In Eq. (4), *Cpower* is the power consumption cost of computation and data accessing on the GPP or the DSP, *Ccomp* is the task execution time on the GPP or the DSP, *Cdeadline* is deadline fulfillness based on the task execution time and the deadline, and *Cload* is the load balance factor based on the task queue lengths on the GPP-side and the DSP-side.

By selecting different values of λ's in Eq. (4), the cost function can adapt to different system requirements. For example, if the remaining power capacity is low, we can increase λ in order to save more power at the cost of slower response time and poor deadline fulfillness.

3.2.3 Implementation of the Scheduler

The proposed scheduler is implanted on an OMAP 5912 platform running Linux. In this section, we will first give an introduction to the OMAP 5912 development board used for the implementation, followed by an introduction to the DSP gateway package used for communication between the GPP core and the DSP core. Note that the overhead of the DSP gateway package is quite high and are not suitable for a tightly-coupled system for practical applications. The reason it is used in the implementation is merely for fast prototyping of the proposed system.

OMAP 5912 is a single-chip dual-core processor developed by Texas Instruments. The architecture integrates a TMS320C55x DSP core and an ARM926EJ-S RISC core. The C55x DSP core provides high performance and low power consumption for digital signal processing tasks and the ARM9 RISC core is very popular for embedded systems. For task dispatching from the ARM core to the DSP core under Linux, DSP Gateway is used. DSP Gateway ([26]) is an open source project for inter-processor communication mechanism on Linux for OMAP family. The DSP Gateway consists of a Linux device driver on the ARM side and a DSP-side kernel library. The Linux device driver provides a convenient interface so that an application on GPP-side can communicate with DSP through normal device system calls. The DSP-side kernel library provides multi-task environment and APIs for user tasks.

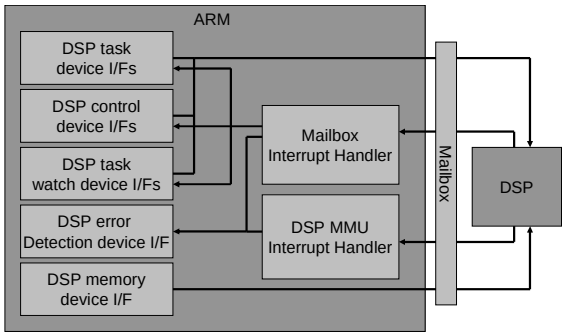


Figure 8. DSP Gateway Driver Block

Figure 8 shows GPP-side functionalities provided by DSP Gateway Linux device driver. The Linux device driver communicates with DSP through two Mailboxes, one for GPP to DSP and another for DSP to GPP.

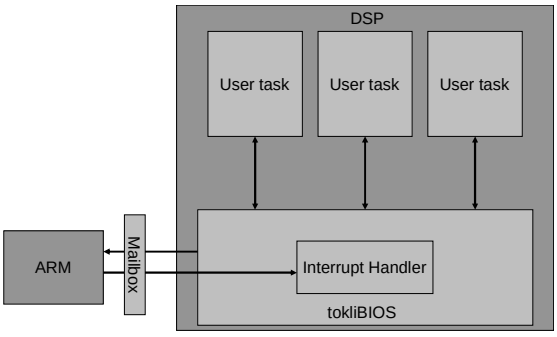


Figure 9. DSP Software Block Chart

Figure 9 shows the DSP Software Block Chart. When a Linux user application accesses the DSP task device, the Linux device driver generates a Mailbox command to DSP. On the DSP side, the system kernel

receives the Mailbox command and registers it into the corresponding queue of the DSP/BIOS TSK.

The implementation of the proposed system is based on Linux 2.6.11 kernel patched by [23] and [25]. We use [24] in DSP Gateway package to load DSP applications to the DSP core from an application (or system service) running under Linux.

Figure 10 shows how DSP dynamic loader daemon (dsp_dld) works.

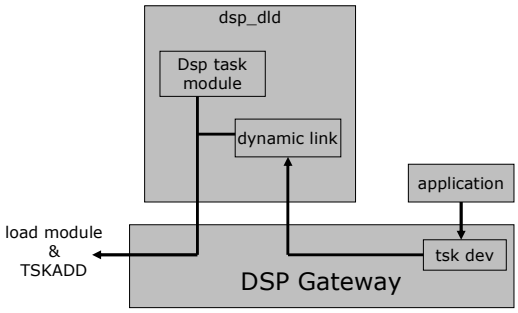


Figure 10. DSP Dynamic Loading Mechanism

We embedded our proposed design into dsp_dld as show in Figure 11.

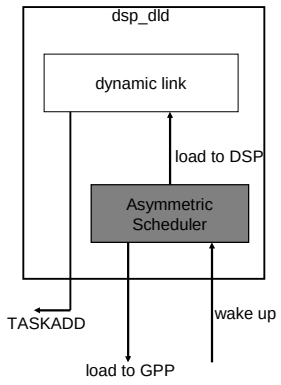


Figure 11. Dsp_dld with AMP Scheduler

When an access event passed to dsp_dld, it will wake up our proposed asymmetric scheduler first. The cost function in the asymmetric scheduler will calculate processing cost according to factors described in section 3.2.2 and then the proposed asymmetric scheduler will dispatch the task to the processing core which has lower cost.

3.2.3 Experiments and Analysis

In this section, the computation components from an MPEG-4 Simple Profile encoder application are used to test the concept of a tightly-coupled AMP system. Table 2 shows computing time for different computing units, including motion estimation (ME), interpolation, and discrete cosine transform (DCT), under ARM/DSP/Dual mode. The test sequence used is the QCIF version of the STEFAN sequence and its length is 150 frames. These tests were conducted on a TI OMAP 1510 platform (the PSI Innovator).

Under ARM mode, the computing units in Table II are processed on ARM core and didn't use any DSP hardware extension. Under DSP mode, the computing units are processed on DSP core using C55x hardware extension. Under dual mode, the computing units are processed on both cores. Table III shows time per computing unit ratio under all modes.

Unit: us

	ARM	DSP	Dual
ME	3883	357	349
Interpolation	441	282	227
DCT	764	342	276

Table II. Time per Computing Unit

	ARM	DSP	Dual
ME	11.12	1.02	1
Interpolation	1.94	1.24	1
DCT	2.77	1.24	1

Table III. Time per Computing Unit Ratio

The next experiment tests the data transfer time between two cores. Table IV shows the data size and iterations of transfer of our experiment. In group 1, for example, a Linux application transfers 12672 bytes of data three times to the DSP core through SDRAM/SARAM/DARAM. All groups in our experiment transferred same amount of data (38016 bytes) at different number of iterations.

Group	1	2	3	4	5	6	7	8
Transfer Size (bytes)	12672	6336	3168	1584	792	396	198	176
Transfer Iterations	3	6	12	24	48	96	192	216

Table IV. Data Transfer Tests Setting

Due to the limitation of SARAM and DARAM capacity, there are no group 1 result for SARAM and group 1, 2, and 3 results for DARAM experiments. Figure 12 and Figure 13 show transfer rates from GPP to DSP and from GPP to DSP to GPP.

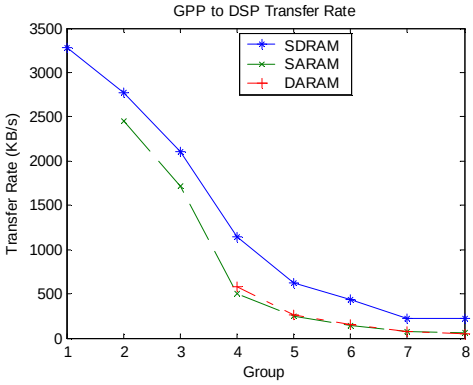


Figure 12. GPP to DSP Transfer Rate

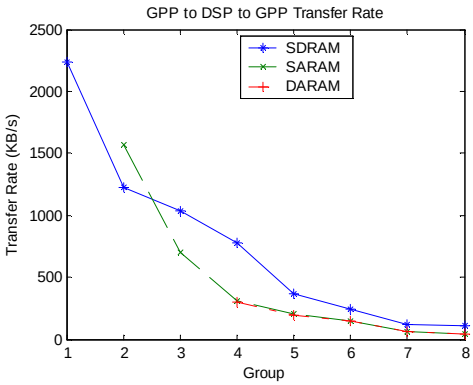


Figure 13. Round-Trip Transfer Rate

GPP writes data to SDRAM through EMIFF Interface in Memory Interface Traffic Controller (TC), and to SARAM and DARAM through MPU Interface (MPUI). It is obvious that transfers data through SDRAM is faster than through SARAM and DARAM. Besides Linux kernel system calls overhead, every GPP to DSP and GPP to DSP to GPP transfer bring 2 and 5 mailbox interrupts. This mechanism results in considerable impact on transfer rate and overall system performance.

3.3 計畫成果自評

For the second year of the projects, we have improved our SoC platform for video codec accelerator. We are expected to complete the RTL design and verification of an sub-pixel motion estimator for H.264 on the ARM Integrator platform. The architecture design of a H.264 CABAC module is done and the RTL coding will be completed. However, the verification of this module will probably be executed in the third year of the project. About the tightly-coupled dual-core embedded kernel scheduler, an implementation based on embedded Linux is done. We are currently working on debugging, testing, and application porting for the scheduler. We have also designed a rate-control mechanism for wavelet-based codec that fulfills rate-distortion optimality for multiple adaptation applications.

In summary, the direction of the project closely matches the original proposals. And the execution of the majority of the project has been going well except that the RTL coding of the CABAC module is a little bit behind schedule.

五、參考文獻

[1] Yueh-Yi Wang and Chun-Jen Tsai, “An Efficient Dual-Interpolator Architecture for Sub-pixel Motion Estimation,” *Proc. IEEE Int. Symp. Circuit And System*, Kobe, May 2005.

[2] Ya-Hui Yu and Chun-Jen Tsai, “A Model-based Rate Allocation Mechanism for Wavelet-based Embedded Image and Video Coding,” *Proc. IEEE Int. Symp. Circuit And System*, Kobe, May 2005.

[3] ISO/IEC 14496-10:2003, "Coding of Audiovisual Objects-Part 10: Advanced Video Coding," 2003, also ITU-T Recommendation H.264 "Advanced video coding for generic audiovisual services."

[4] SMPTE Technology Committee C24 on Video Compression Technology, "Proposed SMPTE Standard for Television: VC-9 Compressed Video Bitstream Format and Decoding Process," 7 Oct, 2003.

[5] Gary Sullivan, Thomas Wiegand, and Keng-Pang Lim, *Joint Model Reference*

Encoding Methods and Decoding Concealment Methods, *JVT-I049*, JVT meeting document, San Diego, Sep. 2003.

[6] X. Qiu, W. Zhang, H. Chen, and R. Zhou, "Low entropy block matching algorithm for motion estimation," ASIC, 2001 Proceedings. 4th International Conference on, Oct. 2001, pp.405-408.

[7] T-C Wang, Y-W Huang, H-C Fang, and L-G Chen, "Performance analysis of hardware oriented algorithm modification in H.264," ICME'03, Vol. 3, 6-9 July, 2003, pp.III-601-4.

[8] M. Horowitz, A. Joch, F. Kossentini, A. Hallapuro, "H.264/AVC baseline profile decoder complexity analysis," CSVT, IEEE Trans. on, Vol. 13, Issue 7, July 2003, pp. 704-716.

[9] Detlev Marpe, et al., "Video Coding with H.264/AVC: Tools, Performance, and Complexity," IEEE Circuits and Systems Magazine, First Quarter, 2004, pp. 7-28.

[10] Jianning Zhang, Yuwen He, Shiqiang Yang, and Yuzhuo Zhong, "Performance and complexity joint optimization for H.264 video coding," ISCAS'03, Vol. 2, 25-28 May 2003, pp. II-888-II-891.

[11] S-Y Choi and S-I Chae, "Hierarchical motion estimation in Hadamard Transform domain," Electronics Letters, Vol. 35, Issue: 25, 9 Dec, 1999, pp. 2187-2188.

[12] Y-Y Wang, Y-T Peng, and C-J Tsai, "VLSI architecture design of motion estimator and in-loop filter for MPE-4 AVC/H.264 encoders," ISCAS '04, Vol. 2, 23-26 May 2004, pp. 149-152.

[13] Albert G. Greenberg and Paul E. Wright. Design and Analysis of Master/Slave Multiprocessors. *IEEE Transactions on Computers*, VOL.40, NO.8, August 1991.

[14] Alberto Avritzer, Mario Gerla, Berthier A. N. Ribeiro, Jack W. Carlyle and Walter J. Karplus. The Advantage of Dynamic Tuning in Distributed Asymmetric Systems. In *Proceedings of INFOCOM*, 1990.

[15] Donald Gross and Carl M. Harris. *Fundamentals of Queueing Theory 3rd Edition*. February 6, 1998.

[16] James W. Wendorf, Roli G. Wendorf and Hideyuki Tokuda. Scheduling Operating System Processing on Small-Scale Multiprocessors. In *Proceedings of the Twenty-Second Annual Hawaii International Conference*, 1989.

[17] K. K. P. Research. Increasing functionality in set-top boxes. In *Proceedings of IIC-Korea, Seoul*, 2001.

[18] Maurice J. Bach and S. J. Buroff. Multiprocessors UNIX Operating Systems. *AT&T Bell Laboratories Technical Journal*, 63(8):1733-1749, October 1984.

[19] Momtchil Momtchev and Philippe Marquet. An Asymmetric Real-Time Scheduling for Linux. In *Proceedings of the International Parallel and Distributed Processing Symposium*, 2002.

[20] *OMAP5912 Applications Processor Data Manual*. Texas Instruments. Dallas, Texas.

[21] Paolo Gai, Luca Abeni and Giorgio Buttazzo. Multiprocessor DSP Scheduling in System-on-a-chip Architectures. In *Proceedings of the 14th Euromicro Conference on Real-Time Systems*, 2002.

[22] Saowanee Saewong and Ragunathan Rajkumar. Cooperative Scheduling of Multiple Resources. In *Proceedings of 20th IEEE Real-Time Systems Symposium*, 1999.

[23] The OMAP Linux Kernel Team. *Linux 2.6.11 omap1 patch file*. http://www.muru.com/linux/omap/.

[24] Toshihiro Kobayashi. *DSP Gateway Dynamic Loader Daemon (dsp_dld) Specification*. May 7, 2005.

[25] T. Kobayashi. *DSP Gateway Linux 2.6.11 omap1 patch file*. http://dspgateway.sourceforge.net.

[26] T. Kobayashi. Linux DSP Gateway Specification Rev 3.2. May 4, 2005.