

A Framework of Multi-characteristics Fuzzy Dynamic Scheduling for Parallel Video Processing on MPSoC Architecture

Da Li, Yibin Hou, Zhangqin Huang, Chunhua Xiao

School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an, P.R. China
Embedded Software and Systems Institute, Beijing University of Technology (BJUT), Beijing, P.R. China
lida1204@gmail.com

Abstract—This paper addresses the inherent unreliability and instability of the multiple uncertain characteristics of complex embedded multiprocessor systems, such as MPSoC (multi-processor system on chip) systems. In this work, we propose a fuzzy sets description for the multiple uncertain characteristics of system, and using fuzzy set membership calculation to determine the scheduling priorities of tasks and resources, in order to improve the capability of concurrent executions of tasks. And also we present a method estimation of comprehensive analysis on the multiple performances in order to increase utilization factor and balancing loads on processors. Through simulation, we demonstrate that the fuzzy dynamic scheduling algorithm can handle wide variety requirements of multiple performances, and we also show an improved approach to solve its demerit of partial adjustment. We implemented a prototype MPSoC system, which utilizes the proposed fuzzy dynamic scheduling algorithm to allocating tasks onto the multiprocessors. And in our case studies, we design a parallel architecture h.264 encoder, running on a multicore MPSoC system on FPGA. The speedup ratio of this prototype application system is up to 12.69.

Keywords—MPSoC System; Dynamic Scheduling; Multi-characteristics; Priority Level; Fuzzy Set; Video Processing; Parallel Processing

I. INTRODUCTION

With the rapid improvement of microprocessor technology, MPSoC system, featured as high in performance, extensible, small in size and low in power consumption, is able to meet requirements of complex embedded systems for computing performance, power consumption and cost, thus having become one of the mainstream development trends of embedded systems [1], [2]. Currently, MPSoC is widely used in computer communication, digital media, auto electronics, medical electronics, aviation and aerospace, industrial control and many other fields, and the relevance of research topics on MPSoC has engaged many enterprises and institutes in research on MPSoC, such as programming model, task scheduling, system development, debugging environment and system safety; among these, real-time dynamic scheduling has become an important topic [3], [4].

While scheduling of multicore processor is NP-Hard [5], there is no optimal polynomial-time algorithm. Meanwhile, realization of a scheduler applicable to systems of different

structures and application fields becomes a new challenge. Researchers apply heuristic search to find a partial optimal scheduling solution for this system [16], [17], [18], [19]. Priority scheduling algorithms are most commonly used. Scheduling algorithms based on task priority mainly include RM, EDF, LSF [6], [7], [8], [9], and some complex fix-priority policies such as the policy of integrated priority [10], [11]. And the myopic [12], [13], thrift [14], group-proper [15] algorithms analyzed in this paper are integrated priority.

However, the above algorithms are dependent on prior characteristic values of the system and loads, and can only provide performance guarantee in a predictable environment when all constraints of system and tasks are known accurately. In actual applications however, characteristics of many tasks are always uncertain and fuzzy. Furthermore, execution performance of computing and control algorithms, real-time operating system, scheduling and network delay may cause delays and errors, and system overload or failures may also lead to uncertain changes in the system. And also some characteristics of tasks are time-varying, such as the laxity time of tasks, the executed ratio of tasks and the remaining ratio of tasks. These characteristics can't be calculated completely accurately, so they can be considered as fuzzy and uncertain random variables. Further, the key features of tasks are relative, and it is not suitable to use a certain value describing the criticality of tasks distinguishingly.

Hence some researchers have proposed to utilize the fuzzy theory for dynamic scheduling, fuzzy sets for description of uncertainties and priority levels of different tasks, and the rules of fuzzy sets for determination of priority levels. Some fuzzy dynamic scheduling algorithms [16], [17] and some fuzzy model [18], [19] for representation of fuzzy constraint rules, but most description functions in these papers are certain, it is inflexible to use certain functions for description of time-varying uncertain characteristics; in addition, all the above methods give no consideration to other characteristics such as key features of tasks, resource factor or limited priority levels, and more system performances. They just only focus on a certain feature or a certain performance of system scheduling, that is unable to meet diversified requirements of a system, such as success ratio of scheduling, schedule time length etc.

This work was supported by the China National Natural Science Foundation under Grant 60773186, and the High-level Personnel Training Project of Beijing University of Technology.

The main goal of this paper is to establish fuzzy model of complex MPSoC system, and apply proper scheduling policy to meet composite requirements for multiple performances. In order to do that, we propose a new real-time fuzzy scheduling algorithm for MPSoC, multi-characteristics fuzzy dynamic scheduling (Multi-Fuzzy algorithm). It uses language fuzzy sets for modeling various characteristics of tasks and system, and calculates the target priority functions of tasks and resources based on the membership rules of these fuzzy sets, and this paper further proposes the solution for partial adjustment.

The rest of this paper is organized as follows: section II Introduces our models, definitions and computation methods of algorithm, section III discusses the multi-characteristics fuzzy dynamic scheduling algorithm. In section IV a case study and simulation results are presented and discussed. Finally, section V summarizes the work.

II. CONCEPTS, DEFINITIONS, METHODS

A. System and Task Model

Provided a real-time MPSoC system S has m processors ($m > 1$), r resources ($r > 1$) and n tasks in a task set ($n > 1$), this system can be described as below:

$$S = \{P_m, R_r\} | T_n | H$$

Definitions of the key parameters are summarized in Table I.

TABLE I. SYMBOLS AND DEFINITIONS IN ALGORITHM

symbol	definition
t_{ar}	Task arriving time
$t_{d\tau}$	Task deadline
$t_{e\tau}$	Task executing time
$ret(\tau)$	The remaining executing time of task τ
$ts(\tau)$	The laxity time of task τ , $ts(\tau) = t_{d\tau} - t - ret(\tau)$;
w_{t1}, w_{t2}	The weights in time factor for tasks
EAT	The earliest available time
ft	Factor time, $ft = [t_{d\tau} + w_{t1} \times EAT(\tau) + w_{t2} * ts(\tau)] / t_{e\tau}$
fp	Factor parallel level $fp = \min(tk_{R1}(\tau), \dots, tk_{Rn}(\tau))$
fc	Factor parallel level $fc = fc_0 + t_{e\tau} - ret(\tau)$
wp	The weights in processors priority
$Up(p)$	The utilization ratio of processors
$EAT_p(p)$	The earliest available time of processors

In system resource R_k has r_k access tokens tk (maximum number of shared access), tokens are independent and each enables a shared access. The available resource state can be signed as

$$\{tk_1^{R_k}(EAT_1^{R_k}), \dots, tk_{r_k}^{R_k}(EAT_{r_k}^{R_k})\}$$

Each of the r_k access tokens has a time label $EAT_{..}^{R_k}$, it means this token's earliest available time. And setting EAT_{Rk_s} , EAT_{Rk_e} denote the EAT of resource R_k in shared mode and exclusive mode, then

$$\begin{aligned} EAT_{Rk_s} &= \min(EAT_j^{R_k}), \\ EAT_{Rk_e} &= \max(EAT_j^{R_k}), \\ 0 < j &\leq r_k \end{aligned} \quad (1)$$

So, the EAT of source for task τ is denoted as:

$$EAT_R = \max(\max(EAT_{Rj_e}), \max(EAT_{Rj_s})) \quad (2)$$

The EAT of system processors is

$$EAT_p = \min(EAT_{p_j}), 0 < j \leq m \quad (3)$$

EAT_{p_j} is the EAT of processor P_j .

Then, the earliest available time of task τ should be

$$EAT(\tau) = \max(t_{ar}, EAT_p, EAT_R(\tau)) \quad (4)$$

B. Fuzzy Set Model

Fuzzy sets are described based on uncertain multi-characteristics of tasks, and target priority function $H_T(\tau)$ is worked out after computation and quantitative evaluation based on the rules of fuzzy sets.

- Relative time ft

ft is an evaluation value of the time factor of all tasks. It changes with increase of scheduling time. As described in Table 1, ft is a weighted comprehensive estimation of $t_{d\tau}$, $EAT(\tau)$ and $ts(\tau)$. Therefore, the relative time of tasks is uncertain and can be described with fuzzy sets, such as “most urgent (shortest)”, “urgent (short)”, “moderate (medium)”, “slack (long)”, “slackest (longest)” and etc., and expressed as $A(sa_1), \dots, A(sa_n)$ fuzzy sets. It is denoted as

$$Sa(ft) = \frac{u_{A(sa_1)}(ft)}{A(sa_1)} + \dots + \frac{u_{A(sa_n)}(ft)}{A(sa_n)}, \quad (5)$$

Here $u_{A(sa_i)}(ft)$ is the membership of relative time to fuzzy set $A(sa_i)$, and $\sum_{i=1}^n u_{A(sa_i)}(ft) = 1$.

- Parallel level fp

Similarly, the capability of parallel execution of tasks is decided by the resources available tokens at that time. fp is equal to the minimum of number of available resources tokens ($fp \geq 0$), and task of lower fp value will be given higher priority. The system resources access tokens are time-varying by the system scheduling process. So fp is also uncertain and can be described with fuzzy sets, classified as “higher”, “medium”, “lower”, and denoted as

$$Sb(fp) = \frac{u_{b1}(fp)}{B(s_{b1})} + \dots + \frac{u_{bn}(fp)}{B(s_{bn})}, \quad (6)$$

- Key feature fc

Key feature is the importance mark of a task in the task set, which is describing the criticality of tasks. And as shown in Table 1, it is related to the executed time of task, which is time-varying. So we use fuzzy sets express it as “important”, “ordinary” and “unimportant”, and denoted as

$$Sc(fc) = \frac{u_{c1}(fc)}{C(sc_1)} + \dots + \frac{u_{cn}(fc)}{C(sc_n)}, \quad (7)$$

C. Calculation Methods

- Composite target function of task priority

Let g indicating uncertain characteristics be represented by $\{A, \dots, G\}$, and the j th fuzzy set of uncertain characteristic x_i as $A(aj)(x_i)$; fuzzy membership $u_{A(aj)(x_i)}$ is the possibility of task τ belonging to $A(aj)(x_i)$; then matrix $Q_f(\tau)$ represents the possible membership of a task in various fuzzy sets:

$$Q_f(\tau) = \begin{bmatrix} u_{A(s_{a1})(x_1)} & \dots & u_{A(s_{an})(x_1)} \\ \dots & \dots & \dots \\ u_{G(s_{g1})(x_g)} & \dots & u_{G(s_{gn})(x_g)} \end{bmatrix} \quad (8)$$

To decide the importance of fuzzy sets and multi-characteristics in priority levels, we set up resultant weight vector W_f and evaluation level vector Δ_f of different fuzzy characteristics, shown as:

$$H_\tau(\tau) = W_f \circ Q_f(\tau) \cdot \Delta_f \quad (9)$$

Here,

$$\Delta_f = [\Delta_{f1}, \dots, \Delta_{fn}]^T, W_f = [w_1, \dots, w_g]$$

We get the membership of task priority level, through the computation on max-min rules of fuzzy set, as $Pb = W_f \circ Q_f(\tau) = [p_{b1}, \dots, p_{bn}]$ and then Pb is multiplied by Δ_f to get the corresponding value of $H_\tau(\tau)$. Scheduling of tasks should be decided according to the $H_\tau(\tau)$ value of current tasks. And generally, the smaller the $H_\tau(\tau)$ value is, the higher the priority level is.

- Target function of processor priority

In the policy of processor selection, we should consider the relationships between the feasible time of tasks, available time of resources and available time of processors, and select the appropriate processor in available processor subsets to reduce the schedule time length of task sets and improve processor utilization.

$$\begin{aligned} H_P(p) &= EAT_P(p) + W_P * U_P(p), \\ Pchoose() & \\ &= \{p_{\min} \mid H_P(p_{\min}) = \min(H_P(p_j)), p_j \in P\} \end{aligned} \quad (10)$$

- Auto-adjustment

Algorithm 3.1 : multi-fuzzy algorithm

Input: $\{T\}^n, \{P\}^m, \{R\}^r$;

Output: construct a complete schedule $Schedule(T)$

```

if  $T$  is feasible; otherwise reject.
1   $f \leftarrow 1, b \leftarrow N$ ; //system initialization
2   $Schedule(T) \leftarrow NULL$ 
3  while ( $t_{ar} \leq t$ ) do UpdateTask( $t, T$ ) od; //
   update  $T\_ready$  when new task is ready
4  case // scheduling
5  for  $i=1$  to  $n$  do
6   $f \leftarrow feasible(ws)$  ;// feasibility analysis
7  if ( $f=1$ )
8  then // get a local scheduling for  $ws$ 
    $Schedule(T_i) \leftarrow localScheduling(ws, I)$ ;
9   $stP_i, stR_i, utP_i, utR_i \leftarrow stP_{i+1}, stR_{i+1}, utP_{i+1}, utR_{i+1}$ ; //
   update system state tables and usage tables
10 else // backward to the former schedule
    $Schedule(T_i) \leftarrow Schedule(T_{i-1})$ ;
11  $stP_i, stR_i, utP_i, utR_i \leftarrow stP_{i-1}, stR_{i-1}, utP_{i-1}, utR_{i-1}$ ; //
   backward system state tables and usage tables
12  $b \leftarrow N-1, f \leftarrow f(r)$ ;
13 fi;
14 if ( $b \leq 0$ ) then abort fi;
15 if ( $SR < SR_0$  or  $UR_P < UR_{P_0}$ ) //
   auto-adjustment of the weight  $W$ 
16 then  $w \leftarrow w + \Delta w$ ;
17 fi;
18 od
19 return ;

```

Figure 1. Multi-Fuzzy Algorithm

The method of auto-adjustment is for purposes of better comprehensive performances in scheduling success, schedule time length and resource utilization, and modification of the weights of various fuzzy characteristics of priority levels, if the schedule performance of a partial task subset is lower than the initial threshold value during scheduling of task sets, namely $W_f' = W_f + \Delta W_f$; and then adjusting step by step until the

schedule performance reaches the initial threshold, so as to improve scheduling performance of task sets.

III. MULTI-FUZZY SCHEDULING ALGORITHM

A. Description of Multi-Fuzzy Algorithm

As described in Fig.1, multi-fuzzy algorithm is divided into several parts: Lines 1 to 2 are designed for initial setting of the system, and line 3 for updating tasks ready for the task ready queue T_ready , in the order of increasing deadlines; Lines 4 to 19 for heuristic scheduling; specifically, lines 6 for feasibility analysis of tasks in the search window ws ; lines 8 to 10 for further extension of partial schedule only when feasible, selecting proper tasks and processors for execution, updating the status table after execution, while system scheduling goes to line 5 for the next scheduling; lines 12 to 14 forces search tree to backtrack to the upper level schedule and for resetting the processor choose 1 when unfeasible, while lines 17 to 19 for auto-adjustment of the algorithm's weight parameter W , so as to properly adjust weight ratios between various characteristics.

B. Metrics for Performance Evaluation

For task scheduling of MPSoC, success ratio is an important metric for evaluation; meanwhile, other metrics such as schedule time length, average scheduling cost and processor utilization, affect scheduling performance directly and reflect the scheduling efficiency per unit of time, the power consumption of the system and effective utilization of resources. Hence this paper studies the metrics of these performances.

- Success Ratio $SR(T)$:

$$SR(T) = N_{success} / N_{sum} \times 100\% \quad (11)$$

$N_{success}$, number of task sets that can be successfully scheduled; N_{sum} , total number of task sets.

- Scheduling Time Length $\overline{TL}$:

$$\overline{TL} = \sum_{i=1}^{N_{success}} t_{finish}(i) / N_{success} \quad (12)$$

$t_{finish}(i)$, scheduling time for each task set; $\overline{TL}$, the average value of schedule time of all the task set available for scheduling;

- Average scheduling cost $\overline{N}_H$

$\overline{N}_H$ refers to the number of times that local scheduling function is called during scheduling of task sets, namely the average length of total schedule number of calculation of task priority and evaluation value of processors.

- Processor Utilization UR_P :

$$UR_{P_j} = \sum_{P_j} t_{\epsilon r} / t_{finish} \times 100\% \quad (13)$$

$\sum_{P_j} t_{\epsilon r}$ refers to the total execution time of tasks allocated to processor P_j .

IV. CASE STUDIES

A. Simulation Experiments

The primary MPSoC system is built on the XUPV5-LX110T Development Board from Xilinx, this board has a Virtex-5 XC5VLX110T FPGA, and a video VGA input and DVI output hardware. Our experiments are simulated on the FPGA, setting the MPSoC system on FPGA processing the video data, and output the video multi-display onto a PC screen.

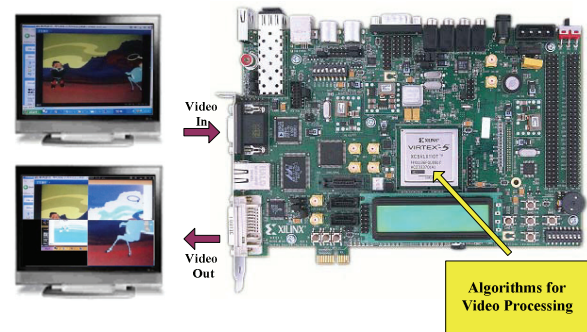


Figure 2. Experiment Platform Overview.

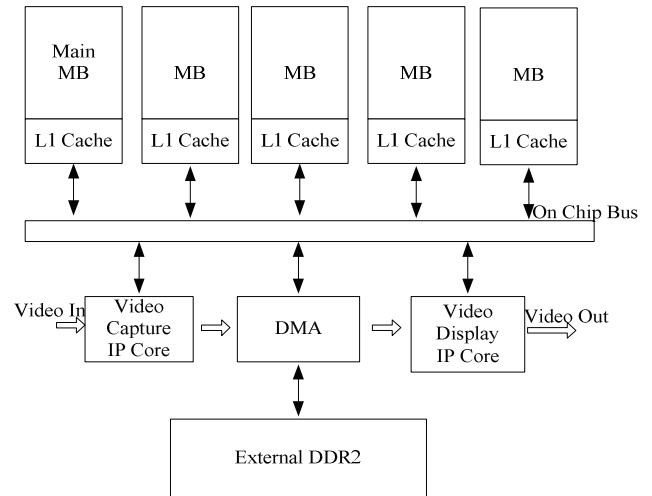


Figure 3. Architecture of MPSoC System for Video Processing.

The video processing is divided into some tasks to be executed on multiprocessors, and each processor has a local BRAM resource and share an external DDR2 SDRAM. The overview of the experiment platform is given in Fig. 2. And the

architecture of MPSoC System for Video Processing is shown in Fig. 3.

B. Results Analysis

Fig. 4 shows effects of laxity r on success ratio SR of algorithms. SR of the four algorithms becomes higher when r

increases; specifically, when r falls in the interval $(0, 0.1)$, SR of multi-fuzzy algorithms are relatively low; but it increases rapidly when r is over 0.3, exceeds other algorithms when r is around 0.5 and remains the highest when r is above 0.5. For multi-fuzzy algorithm with different weights, the success ratio varies significantly when r value is small, and the rate of change is different as r increases, but after r exceeds 0.5, the

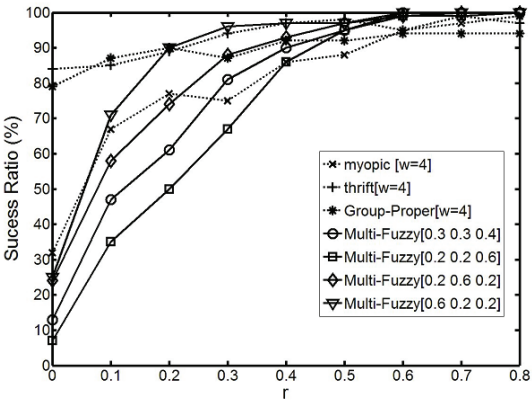


Figure 4. Success Ratio of Scheduling.

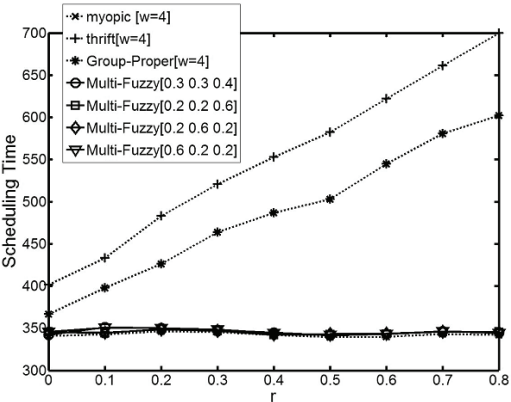


Figure 7. Average Scheduling Time Length.

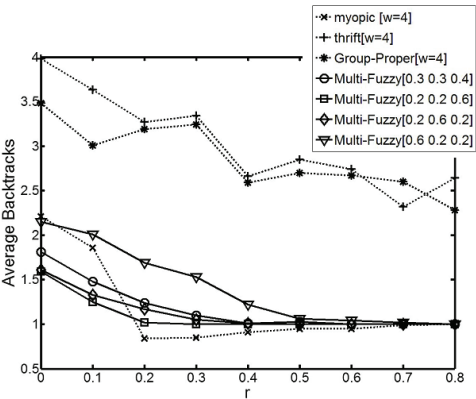


Figure 5. Average Number of Backtracking.

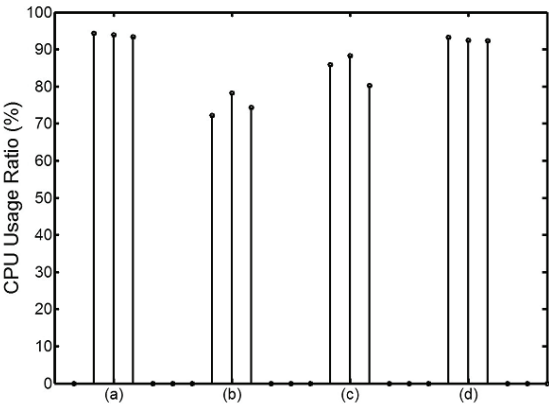


Figure 8. Processor Utilization: (a) Myopic. (b) Thrift. (c) Group-Propor. (d) Multi-Fuzzy

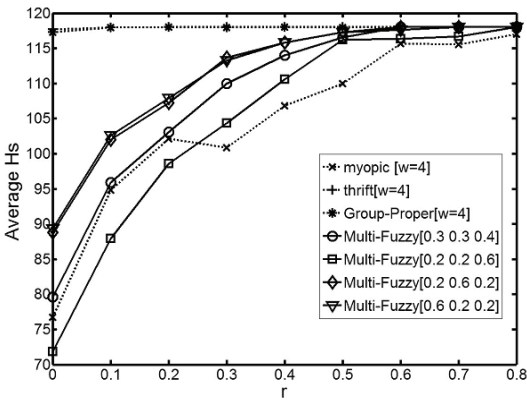


Figure 6. Average Scheduling Cost

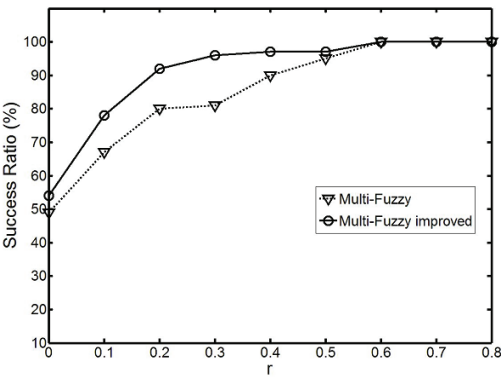


Figure 9. Improvement of Multi-Fuzzy Algorithm

difference becomes insignificant, falling within 10%. The performance test of Average number of backtracking sets maximum backtracking number N as 10. As show in Fig. 5, all algorithms decreases slowly with r ; thrift algorithm is the highest, 3-4 times per task set; that of group-proper algorithm is the second highest with a level of 2.5-3.5 times per task set; and larger number of backtracking means a lower probability of successful search for a feasible solution, and the corresponding scheduling cost and time of the scheduler are added. In Fig. 6, scheduling costs of thrift and group-proper algorithms are obviously higher when r value is small; as r the scheduling costs of algorithms increases, and when laxity r is over 0.6, SR of all algorithms is close to each other, multi-fuzzy algorithm is just only higher than myopic algorithm. $\bar{N}_H$ (shown in Fig. 7) of thrift and group-proper is longer and increases linearly with laxity r , while that of myopic and multi-fuzzy algorithms tend to be stable around 340-350 time units, close to the basic time length of task set generator SC(330 on average); this resulted from effects of the policy of processor allocation of algorithms. And for UR_P in Fig. 8, single processor utilization of the thrift algorithm is the highest, up to 94%; that of myopic and group-proper algorithms are 74% and 85% respectively, and that of the multi-fuzzy algorithm is 92%. In terms of equalizing in processor utilization, performance of the multi-fuzzy algorithm is the best, for loads on various processors are the most even.

C. Improvement Policy of Multi-Fuzzy Algorithm

In the test in section IV-A, SR of multi-fuzzy algorithm is not so high when laxity r is low, thus we adjust part of the scheduling policy. Standards for processor selection can be properly loosened so as to select feasible processors with a suboptimal evaluation function value or a value closer to the average, providing more opportunities for those infeasible tasks. For example, the processor with a $H_P(P)$ value close to $\bar{H}_P$ can be selected, as

$$f_P(r) = \begin{cases} 1, & r > 0.3 \\ Lt(\bar{H}_P), & r \leq 0.3 \end{cases} \quad (14)$$

with set $f_P(r)$ as a selection parameter of the algorithm.

When r is no higher than 0.3, $Lt(\bar{H}_P)$ is the sequence number of processor whose $H_P(P)$ is closest to and less than $\bar{H}_P$; A simulation test is performed in reference to the adjustment policy. It can be seen from simulation results in Fig. 9 that the algorithm is significantly improved when time constraints are tight ($r \leq 0.3$), in that SR is increased from 20%-30% to 50%.

D. Implement h.264 Encoder on FPGA

A high performance FPGA was used to implement an MPSoC of h.264 encoder, as shown in Fig. 10, and the heterogeneous multi-core processor architecture has been designed for the special use of mass data computation. The MPSoC has five MBs (MicroBlaze), which is a software IP core of 32bit general CPU. These MBs are connected by the on

chip high speed interconnection network directly. The Main Processor is in charge of the system initialization and system controlling, and other Synergistic Processors execute parallel video data computation separately. Each processor has a local L1 cache to load and store data used by executing unit. So there is potential competition to the occupation of data transmission. The Multi-Fuzzy Scheduling is running on the Main MB to be evaluated for scheduling and communication of system.

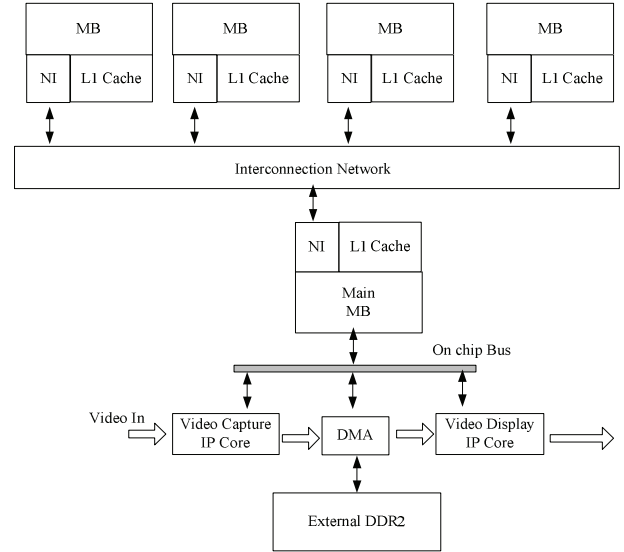


Figure 10. Architecture of Heterogeneous Multiprocessor System for Video Processing

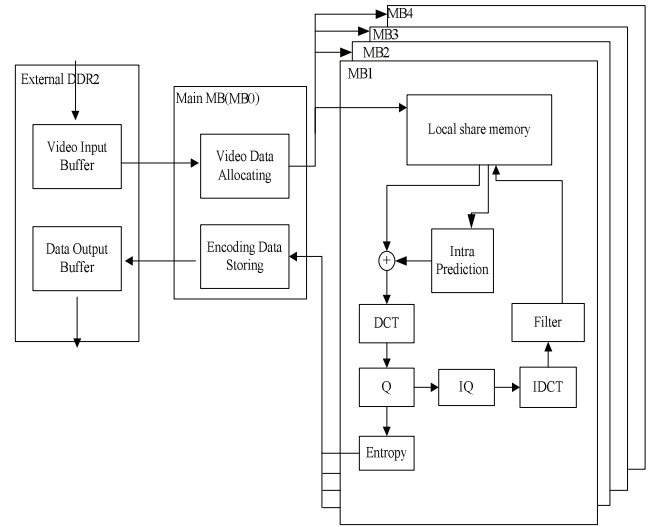


Figure 11. The Data Flow and Task Allocation of h.264 Encoder

In the h.264 video encoding application, the prototype system will capture the video stream as input, encode these video data and stored the encoded data to the output buffer. Fig. 11 outlines the data flow and task mapping of this application. In this application, the main MB is charge for allocating video stream data to the other Synergistic Processors, and gathering the encoded data from the video processing

processors to output into the external memory. Each MB will execute a complete h.264 encoding process. This process is divided up into several tasks: Intra Prediction, Forward and Inverse Transforms, Forward and Inverse Quantization, Reconstruction and Entropy Coder. We map these tasks onto different cores in each MB subsystem.

In the h.264 encoding processing, a frame is divided into many macroblocks. Hence it becomes the basic processing object for each core. The shared local memory (L1 cache) and some buffers between cores (such as FIFOs) are using for cache data, by which the one-way data stream can be divided into Multi-stage pipeline to greatly improve the system parallel performances.

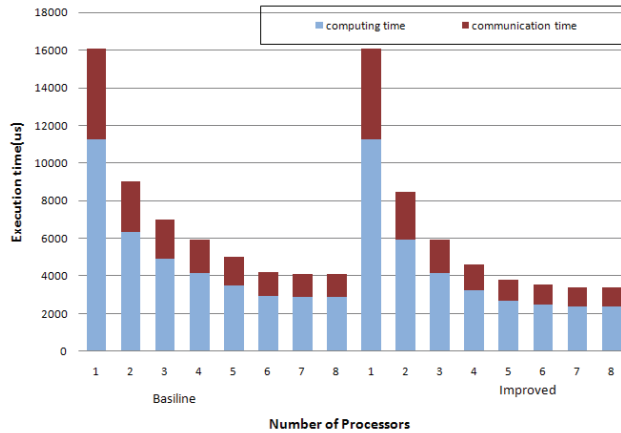


Figure 12. Execution Time of h.264 Encoder

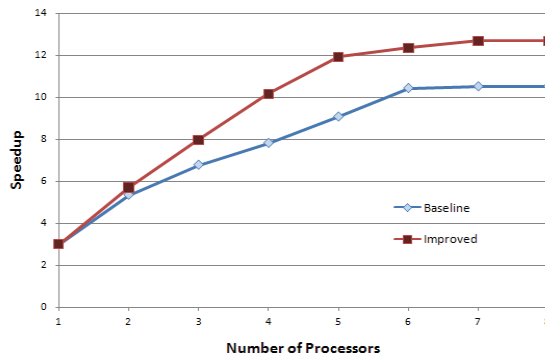


Figure 13. Speedup Ratio of h.264 Encoder

The Fig. 12 shows the execution time of the h.264 encoding for different number of processors. We have run the encoder under 100MHz system clock. In this example the baseline case allocates the tasks with the invariable macroblocks onto these processors, and the improved case use the fuzzy dynamic scheduling to allocate the macroblocks onto the appropriate processor. And the Fig. 13 shows the parallel speedup ratio performance, with the increase of number of processors, the speedup ratio of this application is improved. The highest speedup ratio is up to 10.53 and 12.69 in baseline case and improved case. The improved case can achieve better speedup ratio performance, because it minimize the difference

of each processor's executing time so that there are less communications and high utilization.

V. CONCLUSION

The multiple performances are keys for scheduling algorithms and scheduler. By referring to these scheduling performance parameters, we can study the corresponding performances in detail, such as cost efficiency, scheduling efficiency. The multi-fuzzy algorithm is better suited to task scheduling in a complex system with multiple characteristics, and it works well in various scheduling performances, thus able to meet the requirements for dynamic task scheduling more effectively.

In different target systems actually designed, performance parameter standards and weights may vary significantly due to difference in design key points. Therefore, algorithms should be applied according to actual conditions during design of scheduler.

And we implement a prototype design with the fuzzy dynamic scheduling method for the hierarchical MPSoC architecture, which adopts a MPSoC h.264 encoder. The speedup ratio of this prototype application system is up to 12.69.

REFERENCES

- [1] Tilak Agerwala, Siddhartha Chatterjee, "Computer architecture: Challenges and opportunities for the next decade," IEEE Micro, vol. 25, no. 3, pp. 58-69, 2005
- [2] Wayne Wolf, "The future of multiprocessor systems-on-chips," Proc. 41st Annual Conference on Design Automation, pp. 681-684, 2004
- [3] Baruah S, Nathan F, "The feasibility analysis of multiprocessor real-time systems," Proc. EuroMicro Conference on Real-Time Systems, pp. 85-94, 2006
- [4] Liu Sha, Tarek Abdelzaher, Karlerik Arzen, "Real time scheduling theory: A historical perspective," J. Real-Time Systems, vol. 28, no. 2, pp. 101-155, 2004
- [5] Mok, A.K. "Fundamental Design Problems of Distributed Systems for the Hard Real-Time Environment," PhD dissertation, Dept. of Electrical Eng, MIT, Cambridge, 1983.
- [6] Liu C L, Layland J W, "Scheduling algorithms for multiprogramming in a hard real-time environment," J. ACM, vol. 20, no. 1, pp. 46-61, 1973
- [7] Tarek A. AlEnawy, Hakan Aydin, "Energy-Aware Task Allocation for Rate Monotonic Scheduling," Proc. 11th IEEE Real-Time and Embedded Technology and Applications Symp, pp. 213-223, 2005
- [8] Theodore P. Baker, "An Analysis of EDF Schedulability on a Multiprocessor," IEEE Trans. on Parallel and Distributed Systems, vol. 16, no. 8, pp. 760-768, August 2005
- [9] Myungwon Hwang, Pankoo Kim, Dongjin Choi, "Least Slack Time Rate First: New Scheduling Algorithm for Multi-Processor Environment," Proc. 4th IEEE Int'l Conf. Complex, Intelligent and Software Intensive Systems, pp. 806 - 811, 2010
- [10] Jin Hong; Wang Hong An; Wang Qiang; Dai Guo Zhong, "An Integrated Design Method of Task Priority," J. Software, vol. 14, no. 3, pp. 376-382, 2003
- [11] Xianbo He, Li Yang, Feng Yin, Chunlin He, Gangyuan Zhang, "A fuzzy scheduling model based on multi-characteristics of tasks in embedded soft real-time Systems oriented to communication field," Proc. 2nd IEEE Int'l Conf. on Computer Science and Information Technology, pp. 270-274, 2009
- [12] Ramamritham, K.J., Stankovic, A., Shih, P.-F., "Efficient scheduling algorithms for real-time multiprocessor systems," IEEE Trans. on Parallel and Distributed Systems, vol. 1, no. 2, pp. 184-194, 1990

- [13] Manimaran,G.,Murthy,C.S.R.,“An efficient dynamic scheduling algorithm for multiprocessor real-time systems,” IEEE Trans. on Parallel and Distributed Systems, vol. 9, no. 3, pp. 312-319, 1998
- [14] WangKun, QiaoYing, WangHong An, FangTing, Zou Bing, DaiGuo-Zhong, “Study of a dynamic scheduling algorithm for realtime heterogeneous systems,” J. Computer Research and Development, vol. 39, no. 6, pp. 725-731, 2002
- [15] Bin Xuelian, Yang Yuhai, Jin Shiyao., “A new dynamic scheduling algorithm for real-time multiprocessor systems based on grouping and properly choosing policies,” J. Computers, vol. 29, no. 1, pp. 81-91, 2006
- [16] Terrier F.,Chen Z., “Fuzzy calculus applied to real time scheduling,” Proc. 3rd IEEE Int’l Conf. on Fuzzy Systems, vol. 3, pp. 1905-1910, 1994
- [17] Salmani, V.,Ensafi, R.,Khatib-Astaneh, N., Naghibzadeh, M, “A Fuzzy-Based Multi-criteria Scheduler for Uniform Multiprocessor Real-Time Systems,” Proc. IEEE 10th Int’l Conf. on Integration Technology, pp. 179 – 184,2007
- [18] Litoiu M., Tadei R., “Real-time task scheduling with fuzzy deadlines and processing times,” J. Fuzzy Set and Systems, vol. 117, no. 1, pp. 35-45, 2001
- [19] Dai Hongjun, Xu XinShun, Jia Zhiping, “A Fuzzy algorithm for parallelizability evaluation and load balance on the multi-core processor,” Proc. 4th IEEE Int’l Conf.on Granular Computing, pp. 168 – 171, 2008