Journal of Engineering Science and Technology, Volume 5, Number 3, 2010

DESIGN METHODOLOGY AND HEURISTIC FOR RECONFIGURABLE HYBRID MACRO PIPELINE MULTIPROCESSOR

O. O Olakanmi, O.A Fakolujo
Electrical & Electronic Engineering Department
University of Ibadan, Ibadan, Nigeria

ABSTRACT

*The research developed a reconfigurable hybridized multiprocessor system, **Hybridized Macro Pipeline Multiprocessor (HMPM)**, which combines two multiprocessing techniques -macro pipeline and parallelism. This solves the problems associated with using multiprocessor system to execute sequential applications. In addition, a heuristic is developed for the system, this reconfigures the processors in each cluster according to input load in order to prevent load imbalance. Multiprocessor system has led to increased throughput of system in many applications such as image processing, computer graphic and real time systems. It has also contributed to the increasing use of parallel hardware along with the associated software, which solves the bottleneck associates with high computation tasks. There are different ways of increasing the throughput in a multiprocessing environment. Parallelism technique; where different processors are allocated different parallel-subtasks at the same time. In macro pipeline, all the available processors must sequentially work on all the subtasks. Several researchers have proposed different form of multiprocessors using any of these multiprocessing techniques. However, these techniques are application dependent. That is some applications are sequential, for these applications, parallelizing their execution will reduce the throughput. Some are parallel in nature, executing these sequentially will also reduce the throughput. This made some of the proposed multiprocessor system application-dependent. This paper proposes a reconfigurable **Hybridized Macro Pipeline Multiprocessor (HMPM)**, a hybrid multiprocessor system that combines sequential and parallel execution techniques in order to increase the multiprocessor throughputs and make it application-independent. HMPM uses homogenous processors as the processing elements. A streaming application is taken and broken into a series of subtasks, which are parallel assigned to different pipelined clusters. The problem of mapping each of the streaming unit into each of pipeline stage is solved with a simple architecture and heuristic that efficiently determine the design space.*

***Index Terms**: Reconfigurable, Heuristic, Macro Pipeline, Parallelism, Multiprocessor*

INTRODUCTION

One way to increase computing power beyond the range of today's processors speed is to switch to parallel architectures in which many processors function concurrently. How can the present computing speed be doubled or tripled? The answer lies on creating subtasks from task, and let each computer or processor work on each subtask concurrently. Meanwhile problem with interdependent sub problems, which requires sequentially execution may not benefit from such concurrent parallel processing. This makes Wall(Wall 1993) to understudy applications. He concluded that applications or computer problems can be classified into two categories. The first category consists of applications with low to moderate amounts of parallelism (Asanovic Ras Bodik et al 2006),(Wall 1993) with at most ten instructions per cycle with aggressive branch prediction and large window sizes. The second category comprises applications with large amounts of parallelism; with at least forty instructions per cycle with aggressive branch prediction and large window sizes. With this classification, the first group can be generally referred to sequential applications, which give little or no performance gain if it is parallel executed on multiprocessor system. Meanwhile the second group, generally referred to parallel applications performs 50-100% better if parallel executed on multiprocessor than superscalar processor (Wall 1993).Several researches have been carried out on multiprocessor systems based on one technique or another; however, a few of these are application-specific. This paper proposed a methodology that makes multiprocessor system not to be application- specific and makes the performance-gain intact.The rest of this paper organized as follows: Section 2 gives a broad overview of multiprocessing techniques. Section 3 specifies the reconfigurable Hybrid Macro Pipeline Multiprocessor (HMPM) architecture. Section 4 gives the detail view of the heuristic for reconfiguring the HMPM.

Multiprocessor Techniques

There are a number of ways to develop a multiprocessor. Nowadays, the popular one is to execute multiple tasks in parallel in order to increase the throughput in the multiprocessing environment under the influence of a supporting operating system (Olukotun ; Olukotun and Hammond ; Olukotun, Nayfeh et al. 1996). Another way to use multiprocessor for multiprocessing is to accelerate the execution of sequential applications by using course level pipeline concept called Macro Pipeline (MP) (Howarth, Nazif 1981). MP is a course level pipeline done at the processors' level. It is performances enhance multiprocessing technique, which enables several sub processes of a process to be in various stages of execution at once. It decomposes a process into sequence of sub processes, which can be executed by autonomous processing unit or processor simultaneously instead of concurrently like the parallel concept. By sharing these processing units among several sub processes, MP offers an overlapped processing of activities of one process with those of others, thus increasing the rate at which processes are completed (Bo, Ling et al. 2009).The MP regards an application as a process and executes the derived sub processes consecutively on a chain of processing elements (Olakanmi 2007). It makes use of vector processing paradigm, by indicating the operation to be performed and specifies the list of operands (called vector) on which to operate. However, sequential applications perform 30% better on single processor than multiprocessor architecture. Therefore, it amounts to computational-power wastage if such program is executed on a multiprocessor system, which does not have the ability to execute sequential applications.Many works have been done on the use of macro pipeline technique and development of a reconfigurable multiprocessor system. One of these is DART (David, . et al. 2002), a reconfigurable multiprocessor system. It consists of four pipeline clusters working independently of one another. The communication between these clusters is performed through a shared memory and some reconfigurable point-to-point connection. A programmable processor synchronizes DART operations. A cluster consists of six coarse grain reconfigurable data path and a fine grain FPGA core.Another example is MaRS (Tabrizi, Bagherzadeh et al 2004.); a multiprocessing system which uses macro pipeline concept for multimedia processing and wireless communication. It contains 2D mesh of 32-bit processing elements with a router for routing the data or instructions to neighboring processing elements. MaRS is an improvement on MorphoSys. Some of the weaknesses of MorphoSys are worked. For example, it improves on MorphoSys inter PE communication by the use of router instead of global buses which may introduce delay in data signal transmission. In addition, memory hierarchy in MorphoSys is improved through the inclusion of in-chip memory. This solves the bottlenecked-constraint introduced by some non-streaming data intensive application such as binary spatial partitioning- based ray tracing (Tabrizi, Bagherzadeh et al 2004.). MaRS is reconfigurable based on the number of processing elements participating in the macro pipeline stages. Although MaRS removed overhead caused by buses delay, however, it brings cost overhead through the inclusion of router.PMMLA (Ghare and Soo-Young) is another linear array multiprocessor, which uses macro pipeline method to increase the throughput of system. It engages dynamic reconfiguration scheme to remove load imbalance in the pipeline stages using the execution time at each stage computed during the sequential execution. The ratio of the stages' execution time is used to generate the reconfiguration ratio. It is not flexible. The workload is not constant. If the previous application specification used for reconfiguration differs from next workload specification, the reconfiguration may bring a worse performance. Contrary to PMMLA reconfiguration heuristic, HMPM uses application's cycles in formulating the reconfiguration heuristic and does not use the previous workload cycles in determining the PEs configuration for the next workload. HMPM is an advanced hybrid multiprocessor, which is not application-specific. It operates in two levels; first level inter connection is where parallel subtasks are schedule to each cluster and the second level interconnection is the inner layer of HMPM, which uses macro pipeline technique to execute assigned subtask. It exploits the advantages of the two multiprocessing techniques to solve the bottlenecked constraints in some of the earlier proposed multiprocessors. Pointing to this is the Wall's (Tabrizi, Bagherzadeh et al. ; 2004) conclusion about multiprocessing sequential and parallel applications. In relation to this, is MaRS that uses macro pipeline technique, and an application-specific multiprocessor. In furtherance, if MaRS is used to execute application with large grained thread-level parallelism workloads, the superscalar micro architecture performs 30% better than MaRS. The reason lies with overheads introduced by the macro pipeline technique used by MaRS. However, the multiprocessor MaRS performs 50-100% better than the wide superscalar micro architecture if used to execute sequential application.

Reconfigurable Hybrid Macro Pipeline Multiprocessor (HMPM) Architecture

Recent researches have targeted scheduling, mapping, multiprocessing techniques and reconfiguration heuristic for both homogeneous and heterogeneous multiprocessors. Meanwhile HMPM proposes how the

sequential and parallel execution techniques can co-exist in multiprocessor system. It also proposes a heuristic for determining the efficient architecture based on the tasks' and allocated subtasks' cycles on the multiprocessor system. The HMPM consists of two layers operation; the first level of interconnection contains four clusters each representing parallel unit in massive parallelism multiprocessor system. The second level of interconnection cluster contains number of processing elements PEs, which uses macro pipeline to execute subtask. The intercluster and intracluster communication are performed through the four buses.

Buses

The HMPM uses bidirectional buses for intercluster and intracluster communication; the lower bus provides highway for the allocated subtasks to the first PE of each cluster. Upper four buses physically connected all the PEs in each cluster into ring topology. In order to remove bus contention, switch is introduced at the connecting point of each PEs. Once the switch is set there will be single sender on a bus at a time, hence remove the use of hardware and software for bus control. In HMPM there will never be buses overhead caused by either hardware or software bus controller.In the HMPM, task partitioning is performed with respect to critical path and level of parallelism in the task. At the cluster level, HMPM uses the level of parallelism in the application to partition the application into subtasks for each cluster. Therefore, it makes the multiprocessor at the first interconnection or cluster level to efficiently execute all the subtasks in parallel. Meanwhile at the PEs or second interconnection level, critical path is used to repartition the allocated subtask of the cluster. Critical path is referred to as a call link with the most time consumption in a function call graph of an application. It should be noted for some applications the critical path may be known, that is critical path information might have been stored during programming, or provided by external tool It is assumed that the critical path information and number of cycles of the application are stored during programming. The HMPM contains a controller, which is also a processing element. The controller performed the partitioning for the first and second level interconnection using parallelism and critical path respectively for partitioning the task into subtasks that will allocated to different clusters.

Cluster

To ensure parallel and sequential executions in HMPM, the architecture is split into modularized units called cluster. In each cluster, there is a specific number of processing elements arranged in pipeline pattern. The number of the processing elements for each cluster is not static. It depends on the workload assigned to the cluster. This ensures load balance.Once a cluster is over loaded, the HMPM senses this and reconfigure the architecture by reassigning more processing element(s) to that cluster. As HMPM monitors over-loading in clusters so it senses under-loading in clusters, by releasing some of its processing elements to its neighboring clusters that are overloaded using the reconfiguration ratio.

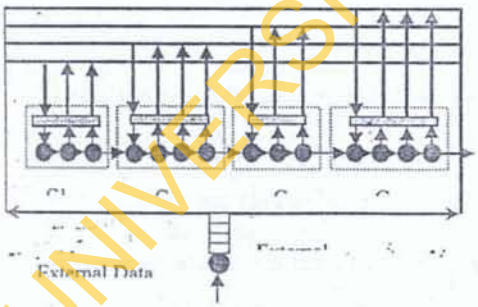

Fig 1: Reconfigurable Hybrid Macro Pipeline Multiprocessor Architecture .

Heuristic for Reconfiguration of HMPM

To eschew load imbalance during reconfiguration of HMPM, the new configuration must satisfied both the new input workload and dependence constraints introduced. Although input dependence constraint would be taken care of by the partitioning controller because it is assumed that for HMPM critical path and dependence information would be stored with the application (during programming). However, no matter how perfect the scheduler or controller may be, changing the workload will fundamentally introduced imbalance if reconfiguration of the architecture is not done. Possibilities of allocating big chunk of task to low capacity cluster can occur. It eventually leads to workload imbalance, which defeats the performance

objective of HMPM as a whole. Thus, there is need for a perfect heuristic that forces load balance on the HMPM.To get a new configuration NC, the subtask cycles x_i (i.e. $x_1,\ldots,x_n$), the corresponding total elements in each cluster C_i (i.e. $PE_1,\ldots,PE_n$) and, task cycles Tc will be used to formulate the reconfiguration ratio. Where $Tc=\sum_{i=0}^{n} x_i+v$ and assuming v represents queue and buses overhead.

The reconfiguration ratio (PE_i') will be the ratio of subtask's cycles (x) allocated to cluster to task's cycles (Tc) multiply by the $\sum_{i=0}^{m} C_i \sum_{i=0}^{n} PE_i$ (i.e. total number of processing element of the entire clusters) to give the new reconfiguration ratio for each cluster $(PE_1',\ldots..,PE_m')$. The under listed conventions are used for the reconfiguration heuristics of HMPM:

Tc: Task cycles.
c: Total number of clusters in HMPM.
B: Number of bidirectional communication buses.
C_i: The ith cluster in the HMPM.
x_i: Total cycles of subtask to ith cluster.
PE_i: Total number of processing elements (PE) in ith cluster of HMPM before reconfiguration.
PE_i': Total number of processing elements (PE) in ith cluster of HMPM after reconfiguration.
E_i: Processing element ith in the cluster of HMPM
OC: Original HMPM before reconfiguration
NC: HMPM after reconfiguration.
N: Number of subtasks

Then,

$$PE_i=\sum_{i=0}^{c} E_i$$

$$PE_i'=\sum_{i=0}^{c} E_i'$$

$$B=\sum_{i=0}^{c} C_{i+1}$$

$Tc=\sum_{i=0}^{n} x_i+v$, where v is the queue and buses overhead

Assuming, $v \square 0$

$$\Rightarrow Tc=\sum_{i=0}^{n} x_i$$

Then, the reconfiguration ratio for each cluster is

$$PE_i=x_i/Tc_i*\sum_{i=0}^{c} PE_i \{\forall[(x_1,\ldots,x_n),(C_1,\ldots,C_n),(PE_1,\ldots,PE_c)]$$

Consider for the entire clusters;
If $PE_i' > PE_i$ then
i.e. change cluster status of $C_{i+1}[E_i]$ to $C_i[E_{n+1}]$ cluster status

$$C_i[E_{n+1}]=C_{i+1}[E_i]$$

else

$$C_{i+1}[E_i]=C_i[E_n]$$

i.e. change cluster status of $C_i[E_n]$ to $C_{i+1}[E_i]$ cluster status

The above heuristic transformed into program segment shown below:

```
while (HMPM loaded)
getTaskcycle() // this function get the workload cycle before partitioning.
getsubtaskcycle() //get the allocated subtasks cycles for all the available
                  //clusters.
```

while ($PE_i' \neq PE_i$)
If $PE_i' > PE_i$ //reconfigure all the clusters to maintain
// load balance for the new task
reset status(C_{i+1}, E_i) //attach the reconfigure processing
//elements to new cluster
$C_i[E_{n+1}] = C_{i+1}[E_i]$
$PE_i = PE_i + 1$
Else if $PE_i' < PE_i$
{
$C_{i+1}[E_i] = C_i[E_n]$
reset status(C_i, En) //attach the reconfigure processing
//elements to new cluster
$PE_i' = PE_i + 1$

CONCLUSION AND ONGOING WORK

This paper has proposed a design for a hybrid multiprocessor system, which hybridizes the macro pipeline and parallelism techniques to increase the throughput of high computational tasks. It also provides a heuristic, which the design uses to adjust itself to variant inputs. The heuristic prevents load imbalance in the architecture. Work is still on going on the simulation of this design; however, it is concluded that:

- HMPM will potentially provide higher throughput than superscalar micro architecture.
- HMPM is not application-dependent.
- The reconfiguration heuristic is simple and more efficient than using the execution time of previous load to determine the next configuration.

REFERENCES

Asanovic, Krste., Ras Bodik ., et al. 2006. The Landscape of Parallel Computing Research: A view from Berkeley, Berkeley.

Bo, F., S. Ling, et al. 2009. Pipeline Processing Method and Apparatus in a Multiprocessor Environment. U. S. patent. **20090077561.**

David, R., ., et al. 2002. A Dynamically Reconfigure Architecture dealing with Mobile Telecommunication Constraints. Parallel and Distributed Processing Symposium IPDPS.

Ghare, G. and L. Soo-Young "Dynamic Reconfiguration of a PMMLA for High Throughput Applications."

Howarth, D. J. and Nazif A. A. 1981. The Dynamic Macro Pipeline. The International Computing Symposium, Proceedings of sixth ACM European Regional Conference 348-360.

Olakanmi, O. O. 2007. Dynamic Macro Pipeline Efficiency Practicability in SAS. Ibadan, University of Ibadan.

Olukotun, K. "A new approach to speculation in the hydra single chip multiprocessor." from http://www-hydra.stanford.edu.

Olukotun, K. and L. Hammond. "The Future of Microprocessor." from http://www-hydra.stanford.edu.

Olukotun, K., B. A. Nayfeh, et al. (1996). "The case for a single-chip multiprocessor." Proceedings 7th International Symp. Architectural Support for Programming Lnguages and Operating Systems (ASPLO) Cambridge, MA **VII.**

Tabrizi, N., N. Bagherzadeh, et al.2004. "MaRS: A Macro Pipeline Reconfigurable System."

Wall, D. W. (Nov. 1993). Limits of instruction-level Parallelism, Digital Western Research Laboratory, WRL.