Exploring Energy-Performance Trade-offs for Heterogeneous Interconnect Clustered VLIW Processors

Rahul Nagpal Y. N. Srikant

IISc-CSA-TR-2005-14

http://archive.csa.iisc.ernet.in/TR/2005/14/

Computer Science and Automation

Indian Institute of Science, India

December 2005

Exploring Energy-Performance Trade-offs for Heterogeneous Interconnect Clustered VLIW Processors

Rahul Nagpal and Y. N. Srikant

Department of Computer Science and Automation

Indian Institute of Science

Bangalore, India

{rahul,srikant}@csa.iisc.ernet.in

Abstract

Clustered architecture processors are preferred for embedded systems because centralized register file architectures scale poorly in terms of clock rate, chip area, and power consumption. Although clustering helps by improving clock speed, reducing energy consumption of the logic, and making design simpler, it introduces extra overheads by way of inter-cluster communication. This communication happens over long global wires having high load capacitance which leads to delay in execution and significantly high energy consumption.

Technological advancements permit design of a variety of clustered architectures by varying the degree of clustering and the type of interconnects. In this paper, we focus on exploring energy performance trade-offs in going from a unified VLIW architecture to different types of clustered VLIW architectures. We propose a new instruction scheduling algorithm that exploits scheduling slacks of instructions and communication slacks of data values together to achieve better energy-performance trade-offs for clustered architectures. Our instruction scheduling algorithm for clustered architectures with heterogeneous interconnect achieves 35% and 40% reduction in communication energy, whereas the overall energy-delay product improves by 4.5% and 6.5% respectively for 2 cluster and 4 cluster machines with marginal 1.6% and 1.1% increase in execution time. Our test bed uses the Trimaran compiler infrastructure.

1. INTRODUCTION

Proliferation of embedded systems has opened up many new research issues. Design challenges posed by embedded processors are ostensibly different from those offered by general purpose systems. Apart from very high performance they also demand low power consumption, low cost, and less chip area to be practical. ILP architectures have been developed to meet the need for high performance in embedded applications [40]. Two major ILP design philosophies are superscalar architecture and VLIW architecture. Superscalar processors have dedicated hardware responsible for scheduling instructions at runtime to improve the performance. The high power consumption, chip area, and cost of these architectures make them less suitable for embedded systems. Another design philosophy is the VLIW architecture, where the compiler is responsible for scheduling. This simplifies the hardware but in order to exploit the ILP in emerging embedded applications, more functional units that can operate in parallel are required. This in turn requires more read and write ports and hence increased chip area, cycle time, and power consumption [42].

Clustering has been proposed to overcome these difficulties with centralized VLIW architectures and to make them suitable for use in embedded systems [15]. A clustered VLIW architecture has more than one register file and connects only a subset of functional units to a register file. Groups of small computation clusters can be fully or partially connected using either a *point-to-point* network or a *bus-based* network. Clustering avoids area and power consumption problems of centralized register file architectures while retaining high clock speed, and can be leveraged to get better performance. Texas Instrument's VelociTI [44], HP/ST's Lx [16], Analog's TigerSHARC [17], and BOPS' ManArray [39] are examples of the recent commercial clustered micro-architectures. IBM's eLite [12] is a research proposal for a novel clustered architecture. A compiler for these architectures is responsible for distributing the operations to resources in different clusters.

Though clustering helps to combat the scalability problem by making components simpler and thereby improving performance and energy consumption, an interconnection network is required for the communication of data values among different clusters. This communication happens over long wires having high load capacitance which in effect takes more time and consumes more energy consumption [33] [21]. Earlier Studies report that a very high percentage (30% to 50%) of total processor energy consumption is attributed to interconnects [32] [45]. Clearly, clustered architectures are attractive only if their benefits outweigh the performance and energy penalties due to interconnections. Thus efficient means of using interconnects are important for clustered VLIW architectures. The primary goal so far has been reduction in the latency of communication to minimize communication delays [43] [23].

Wire delay is a function of its RC time constant where R and C are the resistance and the load capacitance of the wire respectively. R and C are both linear functions of wire length and thus the wire delay has quadratic dependency on the wire length. It is made linear by dividing the wire into segments and by using repeaters [21]. Closely spaced repeaters can help to improve latency but have more area and energy overheads. By tuning the repeater size and spacing between successive repeaters, different latency and power profiles can be obtained for

wires. It has been shown that using 50nm technology, it is possible to design repeaters consuming 1/5 the energy but having twice the delay [9] [34]. Though VLSI technology enables design of interconnects with wires having different energy characteristics, to the best of our knowledge, there have been no efforts in terms of using energy efficient interconnects for clustered VLIW architectures.

In this paper, we propose and evaluate a new energy-aware instruction scheduling algorithm which exploits interconnects of different characteristics in the context of clustered VLIW architectures. The proposed algorithm takes into consideration the interconnect characteristics, and communication slacks of data values together with the scheduling slacks of instructions while steering the communication to an appropriate interconnect, thereby reducing energy consumption without much performance degradation. We consider different flavors of homogeneous interconnects such as latency-optimized and energy-optimized as well as heterogeneous interconnects together with the variation in degree of clustering (no clustering, 2-clustered, and 4-clustered) to perform a detailed performance evaluation. This helps in understanding the energy-performance trade-off in using different varieties of clustered architecture and in making design decisions for clustered architectures targeting embedded domains. Our evaluation uses the Trimaran compiler infrastructure [5].

The rest of the paper is organized as follows. In section 2, we present the motivation for this work with some quantitative results. Section 3 gives a detailed description of our energy-aware instruction scheduling algorithm with a brief mention of implementation details. We also give an example in section 3 to illustrate the notion of communication slack and how it is exploited by our algorithm to yield energy benefits without performance degradation. Section 4 presents a detailed performance evaluation of the proposed algorithm and analyzes the energy-performance trade-offs. Section 5 presents related work in the area of cluster scheduling, energy efficient-scheduling and design of efficient interconnects. We conclude in section 6 with pointers to future directions.

2. MOTIVATION

Previous studies have reported that performance degrades by 12% when the latency of communication is doubled for a four clustered architecture, and that increasing the interconnection bandwidth from one to two improves the performance by as much as 10% [24]. A high speed path for communication of data values among clusters indeed enables better performance, but we argue that not all data values need to be communicated on a high speed path. Though some communications are critical and delaying them can have severe impact on performance, we observe that many communications are non-critical and can still happen on a slow path without affecting performance. *We define the communication slack of a data value on clustered architectures as the number of cycles between the time when the data value to be communicated becomes available (due to completion of execution of the producing instruction) and when the instruction that requires the data value is actually scheduled.* Various causes that can affect the available communication slack of a data value on clustered architecture are as follows :

1) Data dependency among instructions adds to the available communication slack of data values because different parents of an instruction may produce results at different points in time.

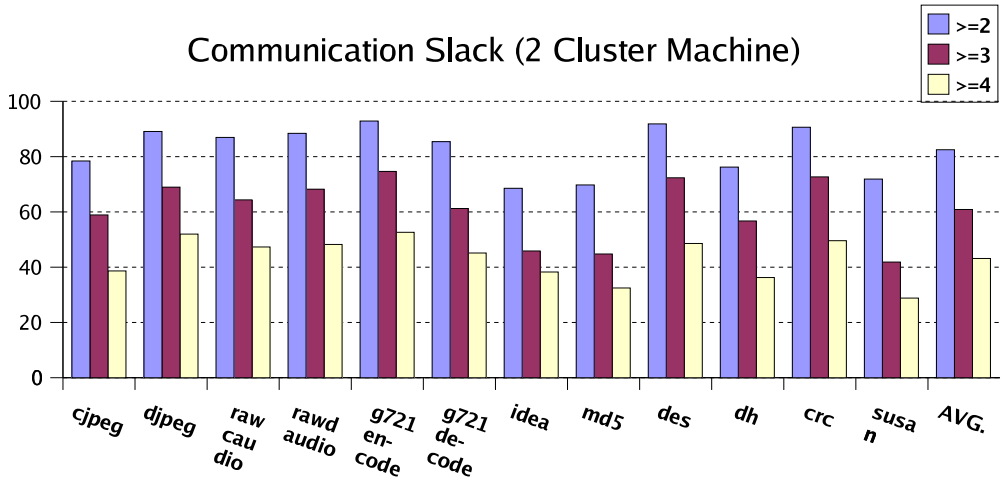

Fig. 1. Communication Slack for 2 Cluster Machine Model

2) Limitation on the available number of functional units makes an instruction requiring communication getting scheduled many cycles after it actually becomes ready to be scheduled.
3) Limitations on the number of available cross-paths, their bandwidth, and latency of cross-path communication are another factors that add to the communication slack of data values.
4) Finally, the peculiarities of cluster scheduling algorithms also add to the communication slack of data values.

Figure 1 presents quantitative results to substantiate our arguments. This fi gure presents the percentage of required communication that has a slack of three cycles (two cycles and four cycles) or more. These results are for a two-cluster machine which has two high speed bidirectional cross-paths between clusters. We observe that all the benchmarks have many communications with high slack values. In particular *djpeg, g721encode, des, and crc* have 70% to 75% of communications with slack value of three cycles or higher. On an average, we observe that 60.88% (82.51% and 43.16%) of communications can sustain a latency of three cycles (two cycles and four cycles respectively) or higher. Thus, even though having a cross-path with inter-cluster communication bandwidth of two is desirable from a performance point of view, having both the wires optimized for low latency is an over kill. This is because improving the latency of communication channel requires closely spaced repeaters which increase the area and energy overheads of repeaters [9]. A more suitable option is to design interconnect with some paths optimized for latency and others for energy [34]. Thus critical communication can take place over the fast but more energy-consuming wires, and the other not-so-critical communication can happen on the slower but energy-effi cient wires. Such a design is going to be benefi cial only if the target workload has a suffi cient number of communications that are non-critical or as we call it have enough communication slack. Further mechanisms (software or hardware) that can steer the communications to the appropriate cross-path depending upon the communication slack of the data value should be available. Our instruction scheduler is one such mechanism.

3. THE SCHEDULING ALGORITHM

The Elcor backend of Trimaran infrastructure has a list scheduling algorithm designed and implemented for flat VLIW architectures [5] [6]. We have extended this algorithm by adding another loop inside the main scheduling loop of the list scheduler to perform cluster scheduling in an integrated fashion. The integrated approach [38] [30] [24] to cluster scheduling makes the cluster assignment decision during temporal scheduling. This is in contrast to phase-decoupled approaches [7] [13] [27] which perform cluster assignment prior to temporal scheduling. Essentially, our integrated clustered scheduling algorithm consists of the following main steps.

1) Prioritizing the ready instructions
2) Assignment of a cluster to the selected instruction
3) Assignment of cross-paths for transferring data values (from other clusters) to the target cluster.

In what follows, we describe how each of these step is performed in our algorithm. An outline of our algorithm is shown in Figure 2.

3.1. Prioritizing the Ready Instructions

Instructions in the ReadyList are prioritized using a priority function that uses instruction slack and number of consumers of the instruction respectively. Slack is defi ned as the difference between earliest start time and latest fi nish time of the instruction and is determined during dependence graph analysis. Instructions with less slack should be scheduled early and are given higher priority over instruction with more slack to avoid unnecessary stretching of the schedule. Instructions with the same slack values are further ordered in decreasing order of the number of consumers. An instruction with more successors is more constrained in the sense that its spatial and temporal placement affects scheduling of more instructions and hence should be given higher priority. Giving preference to an instruction with more dependent instructions also enables better scheduling decisions by uncovering a larger portion of the graph.

3.2. Cluster Assignment

Once an instruction has been selected for scheduling, we make a cluster assignment decision. The primary constraints are :

- The chosen cluster should have at least one free resource of the type needed to perform this operation
- Given the bandwidth of the channels among clusters and their usage, it should be possible to satisfy the communication needs of the operands of this instruction on the cluster by scheduling these communications in the earlier cycles.

Selection of a cluster from the set of the feasible clusters is done as follows. We determine the earliest time when we can schedule the operation under consideration on each of the clusters in the feasible cluster list while adhering to all the resource and communication constraints. The operation is primarily assigned to that cluster where it can be scheduled at the earliest after accommodating all the communications. In case of a tie in this metric, the operation

```
Initialize Early Cycle, Late Cycle And Priorities of the operations
ReadyList=Start Operation
while (CurrentOperations =UnSchedList.pop()) do
  Compute EarlyCycle of the CurrentOperation
  Initialize MinCycle, MinCommCost, and MinCommOption
  for (CurrentCluster ranging from FirstCluster through LastCluster) do
    for (CurrentClusterCycle ranging from EarlyCycle through MaxScheduledCycle) do
      Compute the Cross-path Requirements in CurrentCommOption
      Compute the Communication Cost in CurrentCommCost
      if (FU and Cross-paths required by CurrentOperation are available in CurrentCycle for CurrentCluster) then
        break
      end if
    end for
    if ((CurrentClusterCycle < MinCycle) || (CurrentClusterCycle == MinCycle && CurrentCommCost <=
    MinCommCost)) then
      MinCycle=CurrentClusterCycle
      MinCommCost=CurrentCommCost
      MinCommOption=CurrentCommOption
      TargetCluster=CurrentCluster
    end if
  end for
  while (CurrentComm=CurrentCommOption.pop()) do
    Determine the EarlyCommCycle, LateCommCycle and the CommSlack for CurrentComm
    Schedule the CurrentComm using minimum energy consuming cross-path between EarlyCommCycle and LateCommCycle
  end while
  Schedule CurrentOperation on TargetCluster in MinCycle
end while
```

Fig. 2. The Integrated Cluster Scheduling Algorithm

is assigned to the cluster that minimizes communication requirements. The communication cost is computed by determining the number and type of communications needed by a binding in the earlier cycles as well as the communication that will happen in the future. Future communications are determined by considering the successors of this instruction which have one of their parents bound on a cluster different from the cluster under consideration. This is because if the instruction is bound to the cluster under consideration, it will surely lead to communication(s) in the future while scheduling the successor of the instructions in the future. Although, we have experimented with many other heuristics for cluster assignment, the above mentioned heuristic seems to generate the best schedule in

almost all cases.

3.3. Cross-path binding

The cross-path assignment scheme is designed to minimize the energy consumption due to inter-cluster communication without affecting runtime performance. In order to meet this objective, the low power cross-paths are used in preference to the high power cross-paths wherever possible. More precisely, the assignment of cross-paths to communications is done as follow. To schedule a communication, its earliest scheduling cycle, latest scheduling cycle, and slack values are determined first. The earliest scheduling cycle for a communication is the cycle in which the data value to be communicated is produced in the source cluster, plus one. The latest scheduling time for communication is the scheduling cycle of first consuming instruction, minus one. The difference between the earliest scheduling cycle and the latest scheduling cycle is the communication slack. In order to avoid delaying the consuming instruction and the consequent possible stretch of the schedule, a communication is assigned to a least energy consuming cross-path that can transfer the data value within the available slack for communication. Thus the cross-path assignment scheme maximizes the usage of low power cross-paths subject to the availability of slack in the communication, and thus, as far as possible, performance degradation is minimized and energy saving is maximized.

3.4. Scheduler Implementation

The List scheduler as implemented in Trimaran [5] targets a flat VLIW class of architectures [6]. It maintains a ReadyList of operations whose predecessors have already been scheduled. In each iteration of the main scheduling loop, the highest priority operation is selected from the ReadyList and scheduled in the earliest cycle that satisfies all the resource constraints, starting from the EarlyCycle of the operation. EarlyCycle is the earliest cycle that an operation can be scheduled without violating any dependence constraints with its predecessors. Once an operation is scheduled, the ReadyList is updated. Each operation has an associated NumUnsched count, indicating the number of incoming dependence edges whose source operation has not been scheduled. After an operation is scheduled, NumUnsched count and EarlyCycle of the successor operations are updated. The operation whose NumUnsched count reaches zero is moved to ReadyList. The operations are not necessarily scheduled on cycle-by-cycle basis but dependencies are of course satisfied.

The pseudo code of our integrated clustered list scheduling algorithm is given in Figure 2. For cluster scheduling, instruction in the ReadyList are considered in the priority order as described above. After taking the highest priority instruction, we consider all the clusters one after another. We determine the earliest time when the instruction under consideration can be scheduled on the particular cluster. This is called CurrentClusterCycle. CommunicationCost and CommunicationOption track the cost and interconnect usage respectively for requisite communication and for the binding under consideration. If the earliest time so found is less than the minimum scheduling time on clusters considered so far then CurrentCluster becomes TargetCluster of the instruction. MinCycle, MinCommCost,

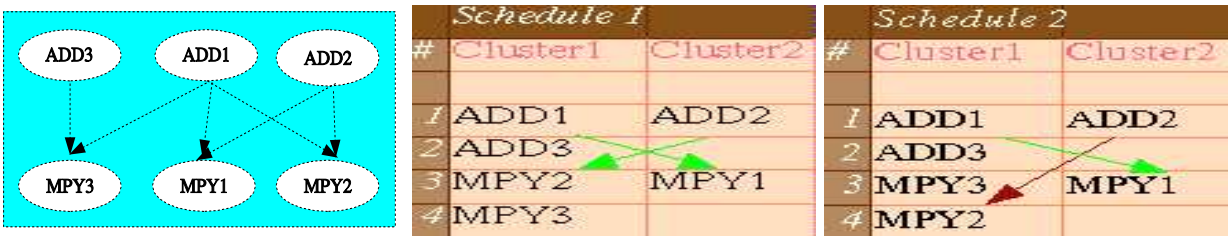

Fig. 3. An Example (a) Data Dependency Graph (b) Schedule 1 (c) Schedule 2

and MinCommOption are also updated accordingly for later use. Once all the clusters have been considered the instruction is assigned to that cluster where it can be scheduled earlier or incurs less communication, in the case of tie. CommunicationOption is used to schedule a cross-path according to the cross-path binding scheme described earlier.

3.5. An Example

In this subsection we present an example to illustrate the notion of communication slack and how it is exploited by the proposed scheduling algorithm to get energy benefi ts without hurting performance. Figure 3 shows a portion of a data dependency graph and two possible schedules for this dependency graph. We Assume a two-clustered machine with each cluster having an adder, a multiplier and a fast communication bus. Schedule 1 has ADD1 and ADD2 scheduled on adders of cluster 1 and cluster 2 respectively in cycle 1. To perform multiplication, the results of these operations are transferred to the other cluster in cycle 2. The remaining addition operation ADD3 is also initiated in cycle 2 on cluster 1. The results of ADD1 and ADD2 can be used in cycle 3 on cluster 1 and cluster 2 respectively to perform MPY2 and MPY1 multipliers. Though MPY3 does not require any inter-cluster communication, it is still executed in cluster 1 at cycle 4 because of non-availability of a multiplier in cycle 3. The scheduler decides to schedule MPY2 ahead of MPY3 in schedule 1 assuming that MPY2 is on the critical path. However, MPY3 gets preference if it is on the critical path as shown in schedule 2. Note that in this case, MPY2 needs to be scheduled in cycle 4 on cluster 1 again because cluster 1 has only one multiplier. *The important point to note here is that the scheduler when scheduling MPY2 in cycle 4 in cluster 2 has the knowledge that it can take two cycles to transfer the result of ADD2 over the communication channel without stretching the schedule*. In such a situation if a slow but more energy-effi cient bus is available, our schedulers decide to steer communication to such a bus (as shown with darker arrow in scheudle 2). Notably, even though three additions are ready to be scheduled in the fi rst cycle only two of them can be scheduled (only two adders are available in this case). Similarly though the addition operations fi nish in opposite clusters in cycle one the results can not be utilized for multiplications in cycle 2 because it takes at least one cycle to transfer the results to the other clusters. This shows how contention among computation and communication resources in clustered architectures manifests itself in the form of greater computation and communication slack. Notably, the contention for resources is more in clustered architectures as compared to flat architectures because of distribution of resources. Our scheduling algorithm leverages this increased slack and takes into consideration the criticality of an instruction and the available cycles to communicate requisite

data values while scheduling an instruction in a given cycle. Accordingly, communication is assigned to the most energy-efficient cross-path that can transfer the value in the available communication cycles.

4. Experimental Evaluation

4.1. Setup

We have used the Trimaran suite [5] for our experimentation. Trimaran was developed to conduct state-of-art research in compilation techniques for ILP architectures with a specific focus on VLIW class of architectures. We have to modify the Trimaran suit [5] to generate and simulate code for a variety of clustered VLIW configurations. The machine description module has been upgraded to describe various clustering related parameters such as the number of clusters, number and types of functional units in each cluster, interconnection network parameters such as number and types of buses among different clusters, their latency and energy consumption. These parameters are fed to the parameterized machine dependent optimization modules in the backend. Major modifications have been performed in the Trimaran scheduler and register allocator module (which was originally written for a class of flat VLIW architectures) to faithfully account for the conflicts due to limitations on the available functional units and registers in a cluster as well as the limitations on the the available cross-paths among clusters. The scheduler has been modified to implement the scheduling algorithm described in the last section. We have used twelve benchmarks out of which nine are from mediabench [28] [2] *(viz. cjpeg, djpeg, rawcaudio, rawdaudio, g721encode, g721decode, md5, des, and idea)*, two from netbench [19] [4] *(viz. crc, and dh)*, and one *(susan)* is from MiBench [20] [3]. We have tried other benchmarks from these suits as well but these are the only ones which compiled successfully and executed correctly using Trimaran framework and hence we report results for them.

We present results for a two-cluster machine and a four-cluster machine with each machine having one integer unit, one floating unit, one branch unit, and one load store unit in each cluster. To ensure a fair comparison and to determine the exact trade-offs results are presented in comparison with an equivalent BASE machine with no clustering but with the same number of functional units and registers. Thus, the BASE flat VLIW machine has two functional units and four functional units of each type for 2-clustered and 4-clustered VLIW. Each configuration has two buses between each pair of clusters. Thus, it is possible to transfer two data values between any pair of clusters simultaneously in both the directions. The first configuration called LL, uses a high speed homogeneous interconnect. This configuration has both the buses implemented with delay-optimized wires and it is possible to transfer a data value between any pair of clusters in one cycle. The second configuration uses a low speed homogeneous interconnect called PP. This configuration has both the buses implemented with the energy-optimized wires and it takes three cycles to finish transfer of a data value between any pair of clusters. However, we assume that the communication network is fully pipelined and that it is possible to initiate a new transfer every cycle. The third configuration represents a heterogeneous interconnect called LP. This configuration has one L wire (capable of transferring one data value in one cycle) and a P wire (which takes three cycles to finish a data transfer). To limit the number of simulations, we do not present the results for cluster configurations with number of buses other than

two. However, earlier studies have demonstrated that increasing the inter-cluster communication bandwidth beyond two gives diminishing returns while increasing area overheads. However, a cross-path bandwidth of two certainly provides signifi cantly better performance compared to a cross-path bandwidth of one [24].

4.2. Energy Model

We have adapted the Epic-Explorer backend [18] [1] to determine the energy consumption in the different components of the data-path. Epic explorer is a collection of activity based power models. An activity based energy model takes into account the statistics of program execution that comprises usage of different components to determine the total energy consumption during program execution. We briefly describe the model that we have used to determine the energy consumption in different components and provide references that describe the associated energy models in detail.

1) Energy Consumption in Functional Units: We have used the model proposed by Cai and Lim [10] to model the energy consumption in various functional blocks. We consider four different kinds of functional units namely integer, floating point, branch, and memory. The active power density and dynamic power density for each kind of functional unit is used along with the usage statistics of different functional units (which is tracked during the execution of the program) to determine the overall power consumption for different functional units. The number and types of functional units in each cluster are taken into consideration to arrive at fi nal energy consumption for functional units.

2) Energy Consumption of Register files : To model the energy consumption in register fi les, the model proposed by Lio et al. [31] is used. This model determines the static and dynamic power consumption of register fi le by taking into account the word size of the register in register fi le and number of registers in a particular register fi le. We have taken into account each different type of register fi le and different register fi les in all clusters with their respective parameters in order to model the overall energy consumption of register fi les.

3) Energy Consumption of Memory Hierarchy: To model the energy consumption of different level of memories (L1 and L2 Data cache, L1 and L2 instruction cache etc.), we have used an energy model proposed by Kamble and Ghosh [25]. This model is also based on the number for transitions for the various circuit elements involved in the activity of a cache. We determine these dynamic statistics during program execution.

4) Energy Consumption of Interconnect: Energy consumption of interconnect is determined by counting number of transitions of the bus and applying the following formula :

$E_{bus} = 1/2V_{dd}^2 \alpha f C_l T$

where V_{dd} is power supply voltage, α is the activity factor, f is the clock frequency C_l is the load capacitance of the bus, and T is time. The energy parameters used for heterogeneity in interconnects are same as the one used in [8] which is based on analysis done in [9] and [34]. It takes into account wires with different latency energy profi les to determine the overall energy consumption of the interconnect. In our simulations, we consider a latency of one cycle and three cycles for L bus and P bus respectively. The dynamic and the leakage energies of L bus are 2.64 times and 2.80 times the dynamic and the leakage energies of P bus respectively [8] [34].

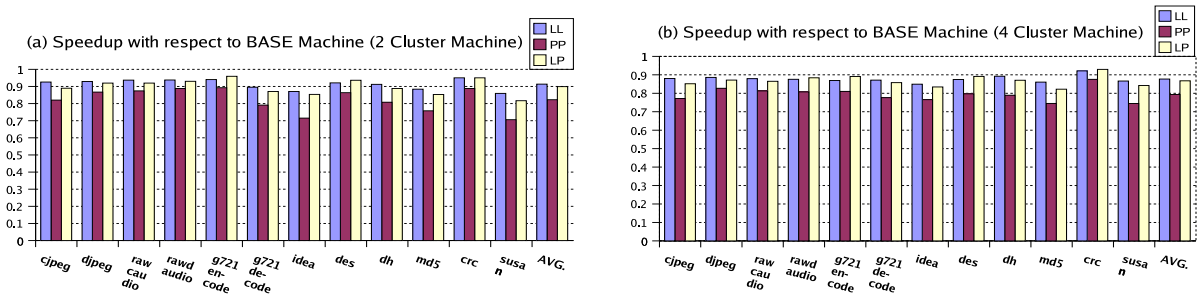

Fig. 4. Speedup w.r.t. the BASE Machine (a) 2 Cluster (b) 4 Cluster

4.3. Results

We have performed a detailed experimental evaluation of the proposed scheme in terms of run-time performance, processing, communication and total energy, and energy-delay product. These results are discussed in detail in the following subsections.

1) Performance: We compare the number of cycles taken to execute the program on different configurations. Figure 4(a) shows speedup for a two-cluster machine with different interconnect configurations with respect to the corresponding BASE machine. We observe that the LL configuration achieves the best performance among all the clustered configurations as expected. The average performance degradation while going from a BASE machine to a 2-clustered machine with the LL configuration is 8.65% whereas the average performance degradation for the PP and the LP configurations is 17.74% and 10.11% with respect to the BASE configuration. The results for a 4-cluster configuration (Figure 4(b)) show similar trends. Table 1 summarizes all the average performance results.

It is clear that heterogeneous interconnects (with a fast and a slow bus) offer nearly the same performance as that of a homogeneous interconnect (with two fast buses) as the performance degradation is only marginal (1.64% and 1.11% for two-cluster and four-cluster machines respectively). However, an energy optimized homogeneous interconnect (having two slow buses) suffers an intolerably high performance degradation of 10.08% and 9.56% for two and four cluster machines respectively. Reading Figure 4(a) (and Figure 4(b)) together with Figure 1 shows the general trend that higher the number of communications that can tolerate high latency, lesser the performance degradation and vice versa. Programs having more communications with high slack values *viz. djpeg, g721encode, des, and crc* suffer only a marginal performance degradation and programs with fewer communications with high slack values *viz. idea, md5, and susan* suffer a moderate performance degradation with the LP configuration. A very small overall performance degradation occurs with the LP configuration. This shows the effectiveness of our communication scheduling mechanism that selectively maps communications with high latency tolerance onto a high latency bus.

2) Energy Consumption: Energy results presented here are based on a conservative assumption that communication energy is 20% of the overall processor energy consumption (though future technological projections indicate higher contribution of communication energy to the overall processor energy consumption). We have used the same technology parameters for determining the energy consumption in the flat BASE and clustered configurations. Clustering reduces the complexity of components and may give additional benefits in terms of energy consumption.

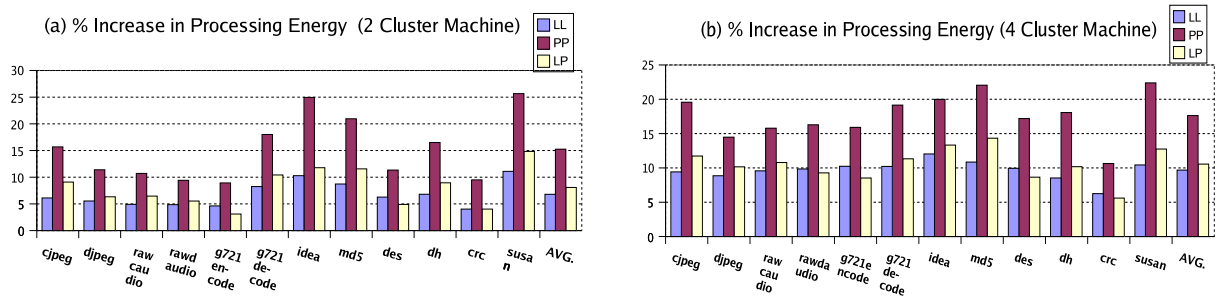

Fig. 5. % Increase in Processing Energy with respect to the BASE Machine (a) 2 Cluster (b) 4 Cluster

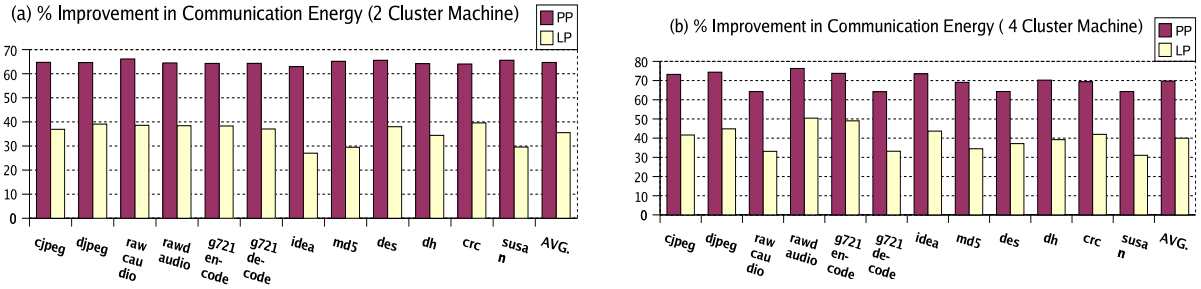

Fig. 6. % Reduction in Communication Energy with respect to the LL Confi guration (a) 2 Cluster (b) 4 Cluster

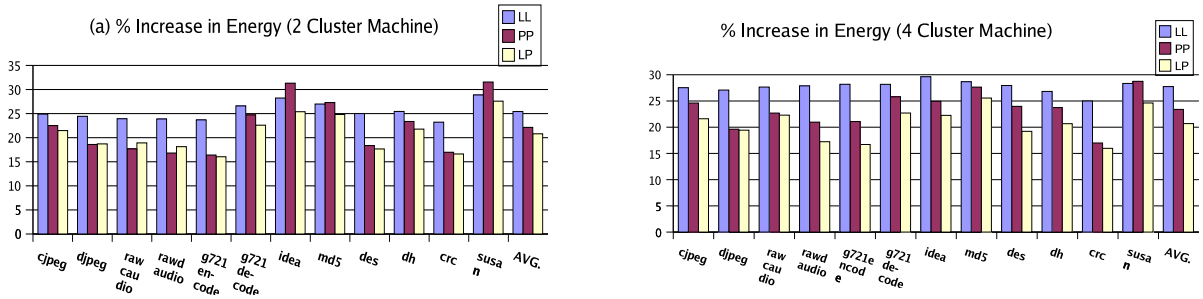

Fig. 7. % Increase in Energy w.r.t. the BASE Machine (a) 2 Cluster (b) 4 Cluster

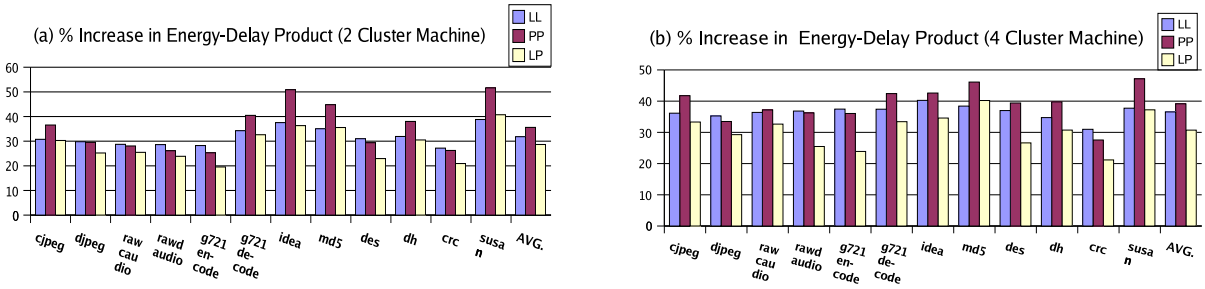

Fig. 8. % Increase in Energy-Delay Product w.r.t the BASE Machine (a) 2 Cluster (b) 4 Cluster

TABLE 1

RESULT SUMMARY FOR PERCENTAGE INCREASE IN EXECUTION TIME, PROCESSING ENERGY, COMMUNICATION ENERGY, ENERGY, AND ENERGY-DELAY PRODUCT. X/Y SHOWS % INCREASE (NEGATIVE NUMBERS MEAN DECREASE) IN X CONFIGURATION WITH RESPECT TO Y CONFIGURATION.

	ET (%)		PE (%)		CE (%)		Energy (%)		EDP (%)	
	2C	4C	2C	4C	2C	4C	2C	4C	2C	4C
LL/BASE	8.65	12.28	6.80	9.68	-	-	25.44	27.75	31.83	36.56
PP/BASE	17.74	20.64	15.26	17.63	-	-	22.14	23.4	35.59	39.15
LP/BASE	10.11	13.25	8.09	10.55	-	-	20.18	20.69	28.67	30.72
PP/LL	10.08	9.56	9.17	8.81	-64.66	-69.74	-4.32	-5.99	5.96	4.19
LP/LL	1.64	1.11	1.41	0.89	-35.54	-39.98	-6.15	-7.67	-4.45	-6.43

However, there is no direct way to exactly quantify these benefits and hence the results presented here are rather conservative estimates. In reality, we expect the energy benefits to be more than these. See Figures 7(a) and 7(b) We observe 25.44%, 22.14%, and 20.81% increase in energy for the LL, PP, and the LP configuration compared to the BASE configuration. Energy consumptions of the PP and the LP configurations are 4.32% and 6.15% less than the LL configuration respectively. There are two reasons of high energy consumption in clustered architectures apart from energy consumption due to interconnects. Firstly, the communication delays prolong the execution on clustered architectures which in turn increases the leakage energy consumption in components. Secondly, clustering causes execution of extra move instructions due to inter-cluster communication which cause extra dynamic energy consumption. The corresponding figures for a four cluster machine are slightly higher but show similar trends. Table 1 summarizes all the energy results.

For a better interpretation of energy results, we present a breakup of energy as processing energy and communication energy in Figure 5 and Figure 6. Improvement in communication energy is presented with respect to the LL configuration (note that the BASE configuration does not have any communication energy consumption). We found that there is only a marginal increase in processing energy while going from the LL to the LP configuration. However, there is a significant increase in processing energy for the PP configuration when compared to the LL configuration. This is due to higher performance degradation in the case of the PP configuration which leads to longer execution and hence more leakage energy consumption in components. Increase in processing energy is higher for *idea, md5, and susan* as compared to *djpeg, g721encode, des, and crc* and this can be attributed to large performance degradation of the former set of programs. Figure 6 presents the reduction in communication energy. We observe that the PP configuration shows about 64.66% reduction in communication energy as compared to the LL for 2-clustered architecture. The reduction in communication energy is almost constant irrespective of the benchmark. This is as expected because both the buses have been replaced by low power buses which offer no diversity to be exploited by the scheduler in terms of improving performance or reducing energy consumption. The communication energy reduction for the LP configuration is 35.54% with respect to the LL configuration. There is

much more variation here in communication energy saved for different benchmarks depending upon the available communication slack for the benchmark and the effectiveness of the proposed scheme in terms of mapping the communication to appropriate wire (we observed 69.74% (PP over LL) and 39.98% (LP over LL) reduction in communication energy while going from the LL to the PP and the LP configuration respectively for a 4-clustered configuration.). Huge saving in communication energy in the PP configuration offsets the increase in processing energy in all the benchmarks except *idea, md5, and susan*. These benchmarks have exceptionally high increase in processing energy because they have less number of communications with high slack values compared to other benchmarks. The LP configuration performs the best in terms of energy consumption. This is because of significant savings in communication energy in the LP configuration with only marginal increase in the processing energy compared to the LL configuration.

We see that going from the BASE machine to a clustered machine adds extra communication overheads in terms of performance as well as energy. The extra execution time also leads to an increase in the processing energy mostly because of more leakage energy consumption. A heterogeneous interconnect (LP) offers less increase in processing energy and less performance degradation while offering significant reduction in communication energy. A homogeneous interconnect optimized for delay incurs high communication energy but offers less performance degradation and least increase in processing energy. In contrast, a homogeneous interconnect optimized for nergy offers high saving in communication energy but intolerably high performance penalty which also translates to increase in processing energy. A heterogeneous interconnect (LP) is the middle ground that offers less increase in processing energy and less performance degradation while still offering significant reduction in communication energy. In the next subsection we present energy delay product to gain a better understanding of energy-performance trade-offs among unified architectures and different clustered VLIW architectures.

3) Energy-Delay Product: Figure 8(a) presents the total energy-delay product for different configurations of a 2-cluster machine as compared to the BASE machine. We observe that the total energy-delay product increases by 31.83%, 35.59%, 28.67% for the LL, PP, and the LP configurations respectively. The average increase in total energy-delay product while going form the LL to the PP and the LP configurations is 5.96% and -4.45% (actually decrease for LP) respectively. Notably the LP configuration provides an improvement over the PP configuration by 10.12%. The results for 4-cluster machine is depicted in Figure 8(b). Table 1 summarizes all the average results for energy-delay product.

The PP configuration, having both interconnects optimized for energy, achieves huge reduction in communication energy. However, the performance degradation due to slow interconnects leads to large performance penalties and the resulting increase in processing energy annuls the benefits obtained due to reduction in communication energy. On the other hand, the LL configuration offers the best performance but at the cost of high energy penalty of delay-optimized interconnects. The LP configuration performs extremely well in terms of energy-delay product. The proposed selective scheduling steers only critical communications to the high speed interconnect. Thus, it maximizes the usage of the low energy interconnect. As a result, it incurs only slight performance and energy penalties as compared to the delay-optimized LL configuration, but is still able to obtain a significant reduction

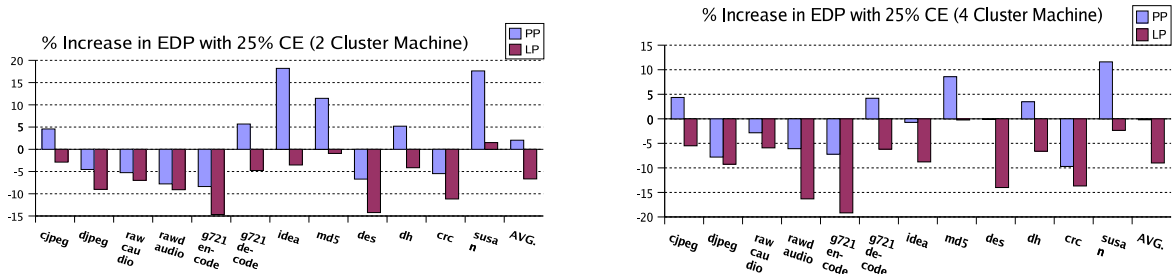

Fig. 9. % Increase in Energy-Delay Product with 25% CE w.r.t the LL Configuration (a) 2 Cluster (b) 4 Cluster

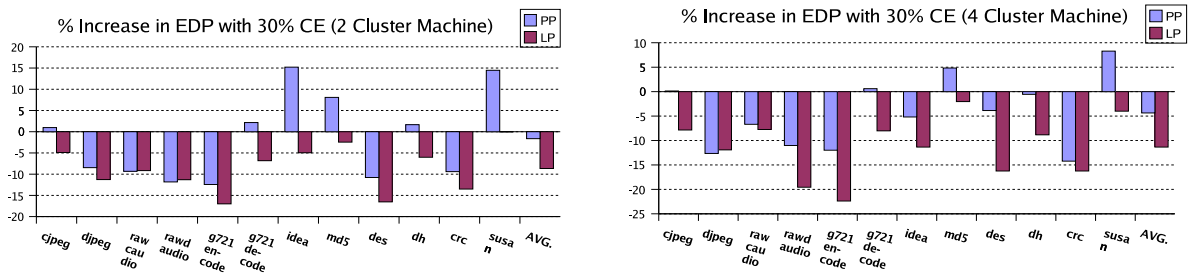

Fig. 10. % Increase in Energy-Delay Product with 30% CE w.r.t the LL Configuration (a) 2 Cluster (b) 4 Cluster

in communication energy. Programs in which more communications have high communication slacks, *viz., djpeg, g721encode, des, and crc* suffer less performance degradation and consequently less increase in processing energy in the LP configuration as compared to the LL configuration. These programs also achieve significant reduction in energy in the LP configuration because of usage of P bus whenever possible. As a result, the energy-delay product for these programs is significantly better in the LP configuration. Even programs in which moderate number of communications have high slack values yield significant benefit in energy-delay product in the LP configuration as compared to the LL configuration because of selective choice of bus.

4) Sensitivity Analysis: Energy and energy-delay product results presented in previous subsections assume conservatively that communication energy contributes only 20% to overall processor energy consumption. See Figure 9 and Figure 10. Considering communication energy as contributing 25% and 30% to overall processor energy, proposed scheme yields significant savings of 6.7% (9.0%) and 8.6% (11.4%) for the 2-clustered (4-clustered) LP configuration compared to the LL configuration. The corresponding savings for the PP configuration are -2.05% (1.64%) and 0.19% (4.36%) with respect to the LL configuration for 2-clustered (4 clustered) machine. Even the PP configuration has better energy-delay product as compared to the LL configuration particularly in case of a 4-clustered machine. However, severe performance penalty makes this configuration less attractive than the LP configuration which provides the best better energy-delay with only marginal performance degradation. The improvement in energy-delay product is increasingly high with more clusters. These results point towards the importance of the proposed scheme with the increasing contribution of communication energy in future technologies and the increasing degree of clustering in future architectures.

5. Related Work

In this section we briefly describe the earlier work done in the area of instruction scheduling for clustered architectures, power aware scheduling for VLIW architectures and efficient interconnects for distributed architectures.

5.1. Instruction Scheduling for Clustered Architectures

Earlier proposals for scheduling on clustered VLIW architectures can be classified into two main categories, viz., phase-decoupled approaches and phase-coupled approaches. A phase-decoupled approach to scheduling works on a data flow graph (DFG) and performs partitioning of instructions into clusters to reduce inter-cluster communication while approximately balancing the load among clusters. The annotated DFG is then scheduled using a traditional list scheduler while adhering to earlier spatial decisions. A major argument in favor of this approach is that a partitioner having a global view of a DFG can perform a better job of reducing inter-cluster communication and load-balancing. The proposals in this direction are due to Ellis [14], Desoli [13], Gonzalez [7], Lapinskii [27], Mahlke [11], Lee [29], and Nystrom [37]. However, the phase-decoupled approach is known to suffer from the phase ordering problem. Since the spatial scheduler has only an approximate knowledge of load on clusters, usage of functional units, and cross-paths, approximate load-balancing often leads to cluster assignments which unnecessarily constrain the temporal scheduler in the later phase. Moreover, some of these schemes are geared towards reducing inter-cluster communication and end up reducing the ILP in the program in this pursuit [38] [24].

An integrated approach to scheduling combats the phase-ordering problem by combining spatial and temporal scheduling decisions in a single phase. The integrated approach considers instructions ready to be scheduled in a cycle and the available clusters in some priority order. The priority order for considering instructions is decided based on mobility, scheduling alternatives, the number of successors of an instruction etc. Similarly, priority order for considering clusters is decided based on communication cost of assignment, earliest possible schedule time. etc. An instruction is assigned a cluster to reduce communication or to schedule it at the earliest. The proposals in this direction are due to Ozer [38], Leupers [30], Kailas [24], Zalamea [47], and Nagpal [36] [35].

5.2. Energy-Efficient Scheduling

Most of the work in the area of energy-efficient scheduling has been done in the context of VLIW architectures. Zhang et al. [48] have proposed a scheme to reduce dynamic and leakage energy in the functional units of VLIW processor. The proposed dynamic energy management scheme assumes availability of multiple operationally duplicate functional units whereas the proposed leakage energy management scheme considers the availability of low leakage mode for the functional units. They have proposed a rescheduling algorithm that exploits the slacks in already scheduled code to remap the functional unit for improving the energy consumption. Leakage energy management scheme is applied to longer slack durations by switching the unused functional units into low power mode. Detailed results have been provided to justify the benefits of these schemes. Kim et al. [26] have proposed a leakage energy management scheme for VLIW processors that determines the ILP available in the program at a the

loop level granularity and determine the canonical set of functional units sufficient to exploit this ILP. The proposed scheme switches the remaining functional units off in order to save energy. Experimental results show that their scheme provide significant leakage energy savings without much performance loss. In a proposal by Gupta et al. [41] leakage energy management is performed for superscalar architectures. They introduced a novel data structure called power-aware flow graph. The proposed scheme works over this graph to determine larger program regions called power blocks which offer opportunities to save leakage energy. Functional units are switched on and off at the boundaries of this region to save leakage energy. Kim et al. [46] have proposed a modulo scheduling algorithm that produces more balanced schedule for software pipelined loops with an objective to reduce the peak power and step power. Kraemer et al. [22] have proposed a compiler directed dynamic voltage scaling technique that uses a compiler algorithm to identify the regions of the code where the clock frequency and voltage can be scaled down with a given performance degradation threshold, thereby reducing energy consumption.

5.3. Efficient Cross-path Design

As compared to reducing energy consumption in function blocks, study of energy efficiency in interconnects is still in its infancy. Previous work has concentrated on improving latency for interconnects in the context of distributed architectures. Gonzalez et al. [23] have evaluated different kinds of interconnects with different topologies and concluded that a point-to-point interconnect with an effective steering scheme is more efficient than a bus-based interconnect. Their experimental results also demonstrate that an asynchronous interconnect offers a performance comparable to an idealized interconnect at a low hardware implementation cost. Terechko et al. [43] has proposed various inter-cluster communication models for clustered architecture and perform a quantitative analysis to compare their benefits.

Closest to our proposal is the work by Balasubramonian et al. [8]. They have also used the same interconnect energy model as proposed in [34] to evaluate techniques such as cache pipelining, exploiting narrow bit-width operands, and interconnect load balancing in the context of superscalar architectures with heterogeneous interconnect. In contrast, our work is more focused on how communication slack in the context of clustered VLIW architecture can be exploited to gain the energy benefits and to explore the energy-performance trade-off while going from a BASE architecture to different configurations of clustered VLIW architectures. *Our results demonstrate that compile-time instruction scheduling utilizing a larger view of program can combine the instruction scheduling and communication scheduling in a profitable manner. On the other hand, a architecture with dynamic scheduling suffers from the problem of limited program view and incurs overheads and complexities of extra hardware for exploiting heterogeneous interconnects at run-time. Thus, the choice of a heterogeneous interconnect is more suitable and beneficial for statically scheduled VLIW architectures as compared to dynamically scheduled architectures.*

6. Conclusions and Future directions

In this work, we have proposed a new energy-aware instruction scheduling algorithm for clustered VLIW architectures that is capable of exploiting interconnect characteristics to get energy benefits without showing high

performance degradation. A number of experiments were conducted to demonstrate the capability of the proposed algorithm and to understand the energy-performance trade-offs in going from the BASE architectures to varieties of clustered architectures on a range of media and networking benchmarks. The major conclusion that we draw form this work is that clustered architecture with heterogeneous interconnect offers better energy-performance trade-offs when used with an effective scheduling algorithm as compared to a cluster VLIW architecture with homogeneous interconnect (which is either optimized or latency or power). Experimental results demonstrate that our instruction scheduling algorithm achieves 35% and 40% reduction in communication energy whereas energy-delay product improves by 4.45% and 6.43% respectively for 2-cluster and 4-cluster machines with a marginal 1.64% and 1.11% degradation in performance. We feel that our technique would become even more important with future architectures, where communication energy would be a larger fraction of total energy consumed. We propose the following extensions to this work.

1) Understanding the interaction between energy-efficient scheduling for functional components (such as functional units, decoder, instruction buses etc.) and the proposed energy efficient scheduling algorithm for interconnects to design an integrated cluster scheduling algorithm that reduces the energy consumption in logic and interconnects.
2) Adapting the proposed algorithm for software pipelining of inner loops and quantifying the associated benefits in the context of modulo-scheduling.

REFERENCES

[1] Epic Explorer. *http://epic-explorer.sourceforge.net/*.

[2] MediaBench . *http://cares.icsl.ucla.edu/MediaBench/*.

[3] MiBench. *http://www.eecs.umich.edu/mibench/*.

[4] NetBench. *http://cares.icsl.ucla.edu/NetBench/*.

[5] Trimaran System. *http://www.trimaran.org/*.

[6] S. G. Abraham, W. M. Meleis, and I. D. Baev. Efficient Backtracking Instruction Schedulers. In *Proc. of Intl. Conf. on Parallel Architectures and Compilation Techniques*, pages 301–308, 2000.

[7] A. Aleta, J. M. Codina, J. Sanchez, and A. Gonzalez. Graph-partitioning based Instruction Scheduling for Clustered Processors. In *Proc. of Intl. Symp. on Microarchitecture*, pages 150–159, 2001.

[8] R. Balasubramonian, N. Muralimanohar, K. Ramani, and V. Venkatachalapathy. Microarchitectural Wire Management for Performance and Power in Partitioned Architectures. In *Proc. of Intl. Symp. on High-Performance Computer Architecture*, pages 28–39, 2005.

[9] K. Banerjee and A. Mehrotra. A Power-Optimal Repeater Insertion Methodology for Global Interconnects in Nanometer Designs. In *Proc. of IEEE Transactions on Electron Devices*, pages 2001–2007, November 2002.

[10] G. Cai and C. H. Lim. Architecture Level Power/Performance Optimization and Dynamic Power Estimation. In *Cool Chips Tutorial at Intl. Symp. on Microarchitecture*, 1999.

[11] M. Chu, K. Fan, and S. Mahlke. Region-based Hierarchical Operation Partitioning for Multicluster Processors. *SIGPLAN Notices*, pages 300–311, 2003.

[12] J. Derby and J. Moreno. A High-performance Embedded DSP Core with Novel SIMD Features. In *Proc. of 2003 Intl. Conf. on Acoustics, Speech, and Signal Processing*, 2003.

[13] G. Desoli. Instruction Assignment for Clustered VLIW DSP Compilers: A New Approach. Technical Report, Hewlett-Packard, 1998.

[14] J. R. Ellis. *Bulldog: A Compiler for VLIW Architectures*. MIT Press, 1986.

[15] P. Faraboschi, G. Brown, J. A. Fisher, and G. Desoli. Clustered Instruction-level Parallel Processors. Technical report, Hewlett-Packard, 1998.

[16] P. Faraboschi, G. Brown, J. A. Fisher, G. Desoli, and F. Homewood. Lx: A Technology Platform for Customizable VLIW Embedded Processing. In *Proc. of 27th annual Intl. Symp. on Computer architecture*, pages 203–213, 2000.

[17] J. Fridman and Z. Greefi eld. The TigerSHARC DSP architecture. *IEEE Micro*, pages 66–76, 2000.

[18] M. P. G. Ascia, V. Catania and D. Patti. EPIC-Explorer: A Parameterized VLIW-based Platform Framework for Design Space Exploration. In *First Workshop on Embedded Systems for Real-Time Multimedia*, 2003.

[19] B. M.-S. Gokhan Memic and W. Hu. NetBench: A Benchmarking Suit for Network Processor. *CARES Technical Report*, 2002.

[20] M. Guthaus, J. Ringenberg, and D. Ernst. MiBench: A Free, Commercially Representative Embedded Benchmark Suite. *IEEE 4th Annual Workshop on Workload Characterization*, 2001.

[21] R. Ho, K. Mai, and M. Horowitz. The Future of Wires. *Proc. of IEEE*, 89(4):490–504, 2001.

[22] C. H. Hsu and U. Kremer. The Design, Implementation, and Evaluation of a Compiler Algorithm for CPU Energy Reduction. In *Proc. of Conf. on Programming language design and implementation*, pages 38–48, 2003.

[23] A. G. Joan-Manuel Parcerisa, Julio Sahuquillo and J. Duato. Effi cient Interconnects for Clustered Microarchitectures. In *Proc. of Int. Conf. on Parallel Architectures and Compilation Techniques*, pages 291–300, 2002.

[24] K. Kailas, A. Agrawala, and K. Ebcioglu. CARS: A New Code Generation Framework for Clustered ILP Processors. In *Proc. of 7th Intl. Symp. on High-Performance Computer Architecture*, page 133, 2001.

[25] M. B. Kamble and K. Ghose. Analytical Energy Dissipation Models for Low-Power Caches. In *Proc. of Intl. Symp. on Low power electronics and design*, pages 143–148, 1997.

[26] H. S. Kim, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin. Adapting Instruction Level Parallelism for Optimizing Leakage in VLIW Architectures. In *Proc. of Conf. on Language, Compiler, and Tool for Embedded Systems*, pages 275–283, 2003.

[27] V. S. Lapinskii, M. F. Jacome, and G. A. De Veciana. Cluster Assignment for High-Performance Embedded VLIW processors. *ACM Trans. on Design and Automation of Electronic Systems*, pages 430–454, 2002.

[28] C. Lee, M. Potkonjak, and W. H. Mangione-Smith. MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems. *Intl. Symp. on Microarchitecture*, 1997.

[29] W. Lee, D. Puppin, S. Swenson, and S. Amarasinghe. Convergent Scheduling. In *Proc. of Intl. Symp. on Microarchitecture*, pages 111–122, 2002.

[30] R. Leupers. Instruction scheduling for clustered VLIW DSPs. In *PACT '00: Proc. of 2000 Intl. Conf. on Parallel Architectures and Compilation Techniques*, page 291, Washington, DC, USA, 2000. IEEE Computer Society.

[31] W. Liao, J. M. Basile, and L. He. Leakage Power Modeling and Reduction with Data Retention. In *Proc. of Intl. Conf. on Computer-aided design*, pages 714–719, 2002.

[32] N. Magen, A. Kolodny, U. Weiser, and N. Shamir. Interconnect-power Dissipation in a Microprocessor. In *Proc. of Intl. workshop on System Level Interconnect Prediction*, pages 7–13, 2004.

[33] D. Matzke. Will Physical Scalability Sabotage Performance Gains. *IEEE Computer*, September 1997.

[34] M. L. Mui, K. Banerjee, and A. Mehrotra. A Global Interconnect Optimization Scheme for Nanometer Scale VLSI with Implications for Latency, Bandwidth and Power Dissipation. In *IEEE Transactions on Electron Devices*, pages 195–203, 2004.

[35] R. Nagpal and Y. N. Srikant. A Graph Matching Based Integrated Scheduling Framework for Clustered VLIW Processors. In *Proc. of ICPP Workshop on Compile and Runtime Techniques Parallel Computing*, pages 530–537, 2004.

[36] R. Nagpal and Y. N. Srikant. Integrated Temporal and Spatial Scheduling for Extended Operand Clustered VLIW Processors. In *Proc. of Conf. on computing frontiers*, pages 457–470, 2004.

[37] E. Nystrom and A. E. Eichenberger. Effective Cluster Assignment for Modulo Scheduling. In *Proc. of 31st annual ACM/IEEE Intl. Symp. on Microarchitecture*, pages 103–114. IEEE Computer Society Press, 1998.

[38] E. Ozer, S. Banerjia, and T. M. Conte. Unified Assign and Schedule: A New Approach to Scheduling for Clustered Register File Microarchitectures. In *Proc. of Intl. Symp. on Microarchitecture*, pages 308–315, 1998.

[39] G. G. Pechanek and S. Vassiliadis. The ManArray Embedded Processor Architecture. In *Proc. of Euromicro Conf.*, pages 348–355, 2000.

[40] B. R. Rau and J. A. Fisher. Instruction-level Parallel Processing: History, Overview, and Perspective. *Journal of Supercomputing*, pages 9–50, July 1993.

[41] S. Rele, S. Pande, S. Onder, and R. Gupta. Optimizing Static Power Dissipation by Functional Units in Superscalar Processors. In *Proc. of 11th Intl. Conf. on Compiler Construction*, pages 261–275, 2002.

[42] S. Rixner, W. J. Dally, B. Khailany, P. Mattson, U. J. Kapasi, and J. D. Owens. Register organization for media processing. *Proc. of Intl. Symp. on High Performance Computer Architecture*, pages 375–386, 2000.

[43] A. Terechko, E. L. Thenaff, M. Garg, J. V. Eijndhoven, and H. Corporaal. Inter-Cluster Communication Models for Clustered VLIW Processors. In *Proc. of Intl. Symp. on High-Performance Computer Architecture*, page 354, 2003.

[44] Texas Instruments Inc. TMS320C6000 CPU and Instruction Set reference Guide. http://www.ti.com/sc/docs/products/dsp/c6000/index.htm, 1998.

[45] H. Wang, L.-S. Peh, and S. Malik. Power-driven Design of Router Microarchitectures in On-chip Networks. In *Proc. of Symp. on Microarchitecture*, page 105, 2003.

[46] H. Yun and J. Kim. Power-aware Modulo Scheduling for High-Performance VLIW Processors. In *Proc. of 2001 Intl. Symp. on Low Power Electronics and Design*, pages 40–45. ACM Press, 2001.

[47] J. Zalamea, J. Llosa, E. Ayguade, and M. Valero. Modulo Scheduling with Integrated Register Spilling for Clustered VLIW Architectures. In *Proc. of annual Intl. Symp. on Microarchitecture*, pages 160–169, 2001.

[48] W. Zhang, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, D. Duarte, and Y.-F. Tsai. Exploiting VLIW Schedule Slacks for Dynamic and Leakage Energy Reduction. In *Proc. of Intl. Symp. on Microarchitecture*, pages 102–113, 2001.