C-016

Design and Architecture of Queue Processor Computing Among queue Words and Random Access Registers (QRP)

Dahanayakage Chandana Dinesh, Masahiro Sowa
Graduate School Of Information Systems
The University Of Electro-Communications
1-5-1 Chofugaoka, Chofu-shi, 182-8585,Tokyo,Japan
E-mail: dahana@sowa.is.uec.ac.jp

ABSTRACT

In this paper we propose Queue with Register Processor (QRP) architecture that can compute among Queue words and random access registers. QRP consists queue registers as well as random access registers for its storage. First we introduce QRP computing model then we introduce hardware structure for proposed model. Each unit of QRP hardware behavior model was tested for its functionality using VerilogHDL. Evaluating process is still continuing for proposed architecture.

1.INTRODUCTION

Microprocessor architectures can be divided in to Register machine, Stack machine and Queue machine concerning its storages. Generally Register machines are popular because of its vast usage [3]. Queue is new architecture still researching on various aspects for its applicability to real world applications [1]. Queue machine significantly changes in two ways with register machines. One is Queue processors use queue registers as its register storage component and second is using Queue Computational Model (QCM) as its computing model [2].

QCM contains instruction level nature parallelism and also it is easy to extract parallelism for higher performances. On the other hand QCM use queue for storing intermediate results that equal to RAM in general microprocessors. Here in QRP design we introduce both random access register computational model and queue computational model to make a flexible and simple computing model to achieve higher performance which suitable for parallel processing [2].

2. QRP GENERAL FEATURES

(i) Arithmetic instruction can compute on both queue words and random access registers.
(ii) Ability to execute in out of order.
(iii) 2 byte fixed instructions.
(iv) 4 stage of pipeline.
(v) 32 bit memory address space.
(vi) 16 random access registers.
(vii) 256 queue registers.
(viii) Four type of functional units. 1.ALU 2.Shift 3.LdSt 4.Branch

2. GENERAL QUEUE MODEL AND QRP MODEL

In any microprocessor design operands are taken from intermediate storages. Generally in Queue processors it is queue storage. But in QRP design it is either queue or random access registers. We believe this effect gives vast accessible area for QRP model.

Following sample instructions in Table2 show the difference between general QCM and QRP model for instruction "add".

General QCM	*QRP*
add	add t,ri,rj
add (offset)	

Table2. instruction format between General QCM ,QRP

(1) source operands taken from queue head and (queue head +1). Result is written to (queue tail).

(2) source operands taken from queue head and (queue head + offset). Result is written to (queue tail).

(3) operands taken from queue or random access registers or both queue and random access registers depend on 3 bit value of operand selection in " t ". Result is written in to (queue tail) or registers depend on " t " value.

3.QRP INSTRUCTION SET FORMAT

QRP processor has 2byte fixed instructions mainly categorized in to 3 parts as shown below.

3.1 Basic Format

Opcode	operand selection	3bit(jjj) operand	4bit(iiii) operand
6 bit	3bit	3bit	4 bit

All arithmetic and logical instructions, shift ,compare instructions belong to this pattern. Each instruction interpreted in to 8 ways using operand selection part enable to access queue as well as random access registers for source data and destination address as shown Table3.1. (qt- queue tail pointer, r-random access register, i,j-signed offset.)

Operand selection	Interpretation (dest<-source1,source2)
000	qt ← q(0+-i) op q(1+-j)
001	r(i) ← q(0) op q(1+-j)
010	r(i) ← q(1+-j) op q(0)
011	qt ← r(i) op q(1+-j)
100	qt ← q(0+-I) op r(j)
101	r(i) ← q(0) op r(j)
110	r(i) ← r(i) op r(j)
111	q(t) ← r(i) op r(j)

Table3.1 interpretation of operands

3.2 Memory access forma

opcode	d /a	Memory offset
6 bit	1 bit	9 bit

load/store instructions ,branch and jump instructions belong to this pattern. "d/a" is 1 bit for accessing base register for memory address calculation.

3.3 ETC format

opcode	offset
6 bit	10 bit

Processor state control instructions are belong to here. example: nop, halt, serial execution on/off .

4. SYSTEM ARCHITECTURE OVERVIEW

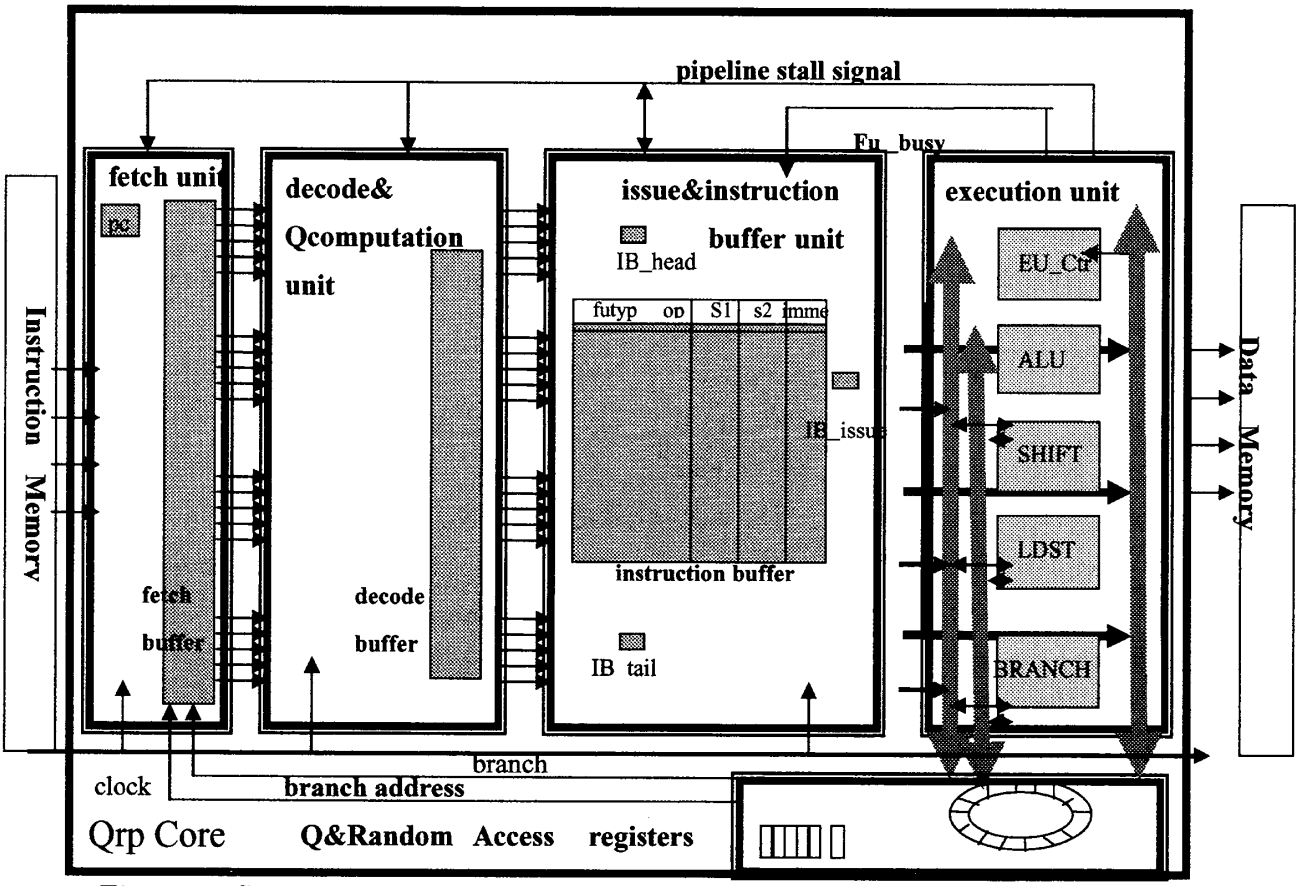

Figure 4 System architecture overview for QRP

QRP system architecture illustrated in figure 4 consist fetch unit, decode and Queue pointer calculation unit, instruction buffer and issue unit, execution unit. There are 4 pipeline stages included in QRP processor architecture.

4.1 Instruction Fetch stage

QRP fetches 8 bytes from instruction memory per cycle and store them in Fetch

buffer. Each fetch steps contains 4 instructions. Fetch buffer output 4 instructions to decode stage in following cycle.

4.2 Instruction Decode and Qpointer Calculation Stage

Decode unit takes 4 instructions in each cycle decode them and write decoded information in to decode buffer. Qpointers for queue operations also calculated here. Both decode and Qpointer calculation done by 4 Dqueue circuits in decode unit. Following cycle decode information for each instruction goes to instruction buffer and issue unit.

4.3 Instruction Buffer and Issue Stage

All executable instructions are written in to instruction buffer that formed cyclic and logically contains infinity entries. Instruction buffer mainly contain 3 pointers. Starting pointer (IB_head) ,last occupied entry s pointer (IB_tail), instruction issue pointer (IB_issue). Issue unit finds dependency and add dependency resolve flag in to instruction. Issue algorithm considering those resolve flags and functional unit availability for issue policy.

4.4 Instruction Execution and Write back Stage

Instructions execution in functional units and write back in to register and memory is controlled by EU_Ctr unit. Eu_Ctr generate signal for data bus read/write operations. QRP contains ALU ,SHIFT unit ,LDST unit, BRANCH unit for execution.

5. Results

We designed all units for QRP, using verilogHDL and tested each module s functionality as shown in Table5. decode and queue computation unit will take the maximum area for QRP.

Unit name	Code size(Num of line)
Fetch unit	67
Dqueue unit	1562
Ibissue unit	1100
Execution unit	949
Register unit	120
Memory unit	50

Table 5. Unit name and code size for QRP

Now we are integrating whole units in to single unit CPU.

6.REFERENCE

[1]Masahiro Sowa, Ben A.Abderazek, Soichi Shigeta, Kirilka Nikolova, Tsutomu Yoshinaga , Proposal And Design Of A Parallel Queue Processor Architecture(PQP),IASTED Int. Conf. On Parallel and Distributed Computing and System Nov 2002

[2]OverQueue QRP(PQPrr),V5.11, Technical Report 2004/06/17, UEC, IS ,SOWA Laboratory

[3]John L. Hennessy, David A. Patterson ,Computer Architecture: A Quantitative Approach Third Edition, Morgan Kaufmann,