ESE 545 Project White Paper

Yufei Ren
<yufren@ic.sunysb.edu>

Group Members

Yufei Ren 108006070 yufren@ic.sunysb.edu

Brief Project Description

Model implementation Language

This project will be implemented by **Verilog**.

SPU Instruction Subset

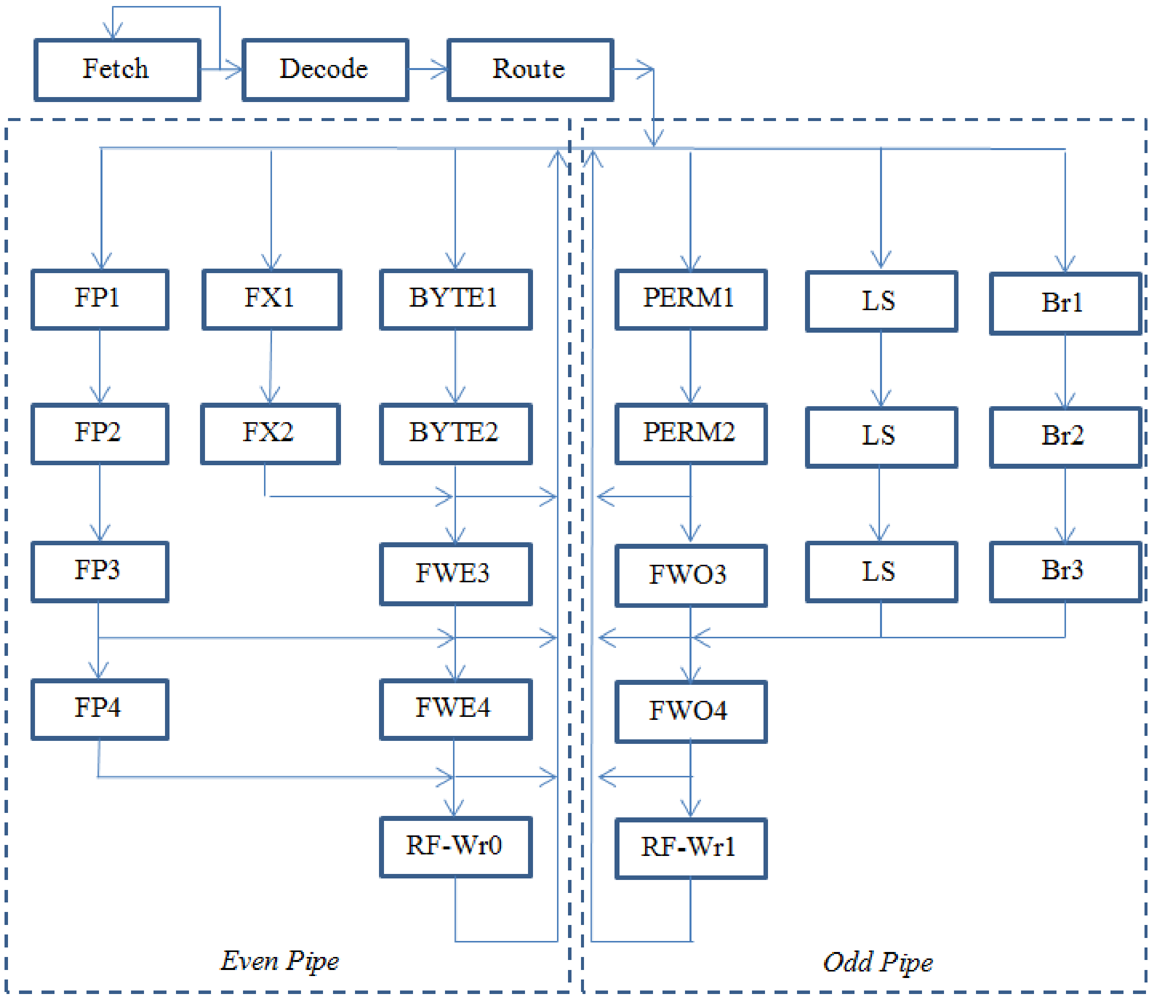
The project implements 31 instructions which is a subset of Cell Processor Instruction Set. The following table specifies the execution unit and the latency of such unit.

Mnomonic	Instruction	Execution Unit	Unit Latency
`lqx`	Load Quadword(x-form)	Load Store	4
`stqx`	Store Quadword(x-form)	Load Store	4
`il`	Immediate Load Word	Load Store	4
`ah`	Add Halfword	Simple Fixed	2
`a`	Add Word	Simple Fixed	2
`ai`	Add Word Immediate	Simple Fixed	2
`sf`	Subtract from Word	Simple Fixed	2
`sfi`	Subtract from Word Immediate	Simple Fixed	2
`mpy`	Multiply	Simple Precision	4
`mpyi`	Multiply Immediate	Simple Precision	4
`avgb`	Average Bytes	Byte	2
`absdb`	Absolute Differences of Bytes	Byte	2
`gbb`	Gather Bits from Bytes	Permute	2
`and`	And	Simple Fixed	2
`or`	Or	Simple Fixed	2
`xor`	Exclusive Or	Simple Fixed	2
`nand`	Nand	Simple Fixed	2
`nor`	Nor	Simple Fixed	2
`shl`	Shift Left Word	Permute	2
`rot`	Rotate Word	Permute	2

br	Branch Relative	Branches	4
bra	Branch Absolute	Branches	4
brnz	Branch If Not Zero Word	Branches	4
brhnz	Branch If Not Zero Halfword	Branches	4
hbr	Hint for Branch (r-form)	Branches	4
fa	Floating Add	Single Precision	4
fs	Floating Subtract	Single Precision	4
fm	Floating Multiply	Single Precision	4
fceq	Floating Compare Equal	Single Precision	4
fcgt	Floating Compare Greater Than	Single Precision	4

Dual-issue SPU Instruction Pipeline Structure

The Dual-issue SPU instruction pipeline structure includes general steps such as fetch instruction, decode instruction, and route instruction. Two parallel pipes, even pipe and odd pipe, are used for Fix point/Float point/Byte computation and Permute/Local Store/Branch computation respectively. The pipeline architecture is shown as follows,



There are 8 stages in the processor for pipelined architecture.

Fetch

SPU instruction `Fetch` stage gets data from local storage read.

Decode

`Decode` stage parses instruction. If it is the `Branch` instruction category, this step will get the hint for branch result. `Decode` stage also determines whether there is any related or dependent register being used or waited in the previous instructions to avoid read after write, etc.

Route

`Route` stage checks whether the instruction will be executed in even pipe or odd pipe.

FP

`FP` stages are used to do float point computation.

FX

`FX` stages are used to do fixed point computation.

BYTE

`BYTE` stages are used to do byte computation, such as average bytes, Absolute Differences of Bytes.

PERMUTE

`PERMUTE` stages are used to compute shifts, rotates, gathers etc.

LS

`LS` (Local Store) stages are used to load or store data between registers and local store module.

BR

`BR` (Branch) stages are used to do branch prediction.

FWE, FWO

`FWE` (Forward in Even pipe) and `FWO` (Forward in Odd pipe) stages are used to forward the result of the previous instructions into the following executing instructions which needs the result value.

RF-Wr

`RF-Wr` stages are used to write the result back into the register file.

Convert SPU Assemble Code to Binary Code

For testing and evaluation of the processor implementation, the project uses a separate C program to translate the assemble text file into the true binary code file. The binary code follows the principle of the cell processor.

Here is the logic for converting assemble code to binary code. Each instruction contains OpCode, register number, immediate data and/or reserved fields. The OpCode field is translated according to the Cell Manual. There are 128 registers in total, and each register is assigned a unique number. The program translates the ascii string, *r1* for example, into the binary bits - *0000001*, which stands for the first register. The same thing is with the immediate data. The immediate might be various length, so the program should cut or extends the proper bits.

Another important issue is that the instruction fields is not 8 bit alignment. The program needs offer bit operation binary combination or splitting. Because all the instruction in this project is 32 bits, it is a not hard for program to translate a 32 bytes ascii into a 32 bits - 4 bytes instruction.