Instruction Set Architecture

Op1	Op2	Instruction	Type	Opcode (3)	rs2(4)	rs1(4)	rd (4)	func (4)	Operation
				18-16	15-12	11-8	7-4	3-0	
rs1	rs2	ADD rd, rs1, rs2	R	0_0_0	r2(4)	r1(4)	rd (4)	0_0_0_0	rd = rs1 + rs2
rs1	rs2	SUB rd, rs1, rs2	R	0_0_0	r2(4)	r1(4)	rd (4)	0_0_0_1	rd = rs1 – rs2
rs1	rs2	MUL rd, rs1, rs2	R	0_0_0	r2(4)	r1(4)	rd (4)	0_0_1_0	rd = rs1 * rs2
rs1	rs2	DIV rd, rs1, rs2	R	0_0_0	r2(4)	r1(4)	rd (4)	0_0_1_1	rd = rs1 / rs2
rs1	1	INC rd, rs1	R	0_0_0	x	r1(4)	rd (4)	0_1_0_0	rd = rs1 + 1
rs1	1	DEC rd, rs1	R	0_0_0	x	r1(4)	rd (4)	0_1_0_1	rd = rs1 - 1
rs1	rs2	AND rd, rs1, rs2	R	0_0_0	r2(4)	r1(4)	rd (4)	0_1_1_0	rd = rs1 & rs2
rs1	rs2	OR rd, rs1, rs2	R	0_0_0	r2(4)	r1(4)	rd (4)	0_1_1_1	rd = rs1 | rs2
rs1	rs2	XOR rd, rs1, rs2	R	0_0_0	r2(4)	r1(4)	rd (4)	1_0_0_0	rd = rs1 ^ rs2
rs1	x	NOT rd, rs1	R	0_0_0	x	r1(4)	rd (4)	1_0_0_1	rd = ~rs1
x	rs2	fft rd, r2	R	0_0_0	r2(4)	x	rd (4)	1_0_1_0	
x	rs2	enc rd, r2	R	0_0_0	r2(4)	x	rd (4)	1_0_1_1	
x	rs2	dec rd, r2	R	0_0_0	r2(4)	x	rd (4)	1_1_0_0	

Op1	Op2	Instruction	Type	opcode (3)	imm_4	rs1/imm_4	rd/imm_4	func (4)	Operation
				18-16	15-12	11-8	7-4	3-0	
x	x	LDbu rd, addr	I	0_0_1	imm_4	imm_4	rd(4)	0_0_0_0	rs1 = memory[addr] (byte unsigned)
x	x	LDb rd, addr	I	0_0_1	imm_4	imm_4	rd(4)	0_0_1_0	rs1 = memory[addr] (byte signed)
x	x	LD rd, addr	I	0_0_1	imm_4	imm_4	rd(4)	0_1_0_0	rs1 = memory[addr] (word)
x	x	STb addr, rs1	I	0_0_1	imm_4	rs1(4)	imm_4	0_0_0_1	memory[addr] = rs1 (byte)
x	x	ST addr, rs1	I	0_0_1	imm_4	rs1(4)	imm_4	0_1_0_1	memory[addr] = rs1 (word)

Op1	Op2	Instruction	Type	opcode (3)	r2(4)	r1(4)	imm_8	Operation
				18-16	15-12	11-8	7-0	
rs1	rs2	BEQ rs1, rs2, addr	B	0_1_0	r2(4)	r1(4)	imm_8	if (rs1 == rs2) PC = addr
rs1	rs2	BNE rs1, rs2, addr	B	0_1_1	r2(4)	r1(4)	imm_8	if (rs1! = rs2) PC = addr

Op1	Op2	Instruction	Type	opcode(3)	imm_14	func (2)	Operation
				18-16	15-2	1-0	
x	x	JMP addr	J	1_0_0	imm_14	0_0	PC = addr
PC	3	CALL addr	J	1_0_0	imm_14	0_1	stack [SP] = PC + 3; SP = SP - 1; PC = addr
x	x	RET	J	1_0_0	imm_14	1_0	SP = SP + 1; PC = stack [SP]

FFT r1, r2	PS	1_0_1	r2(4)	r1(4)	x	0000	FFT (memory[r2], result); memory[r1] = result
ENC r1, r2	PS	1_0_1	r2(4)	r1(4)	x	0001	encrypted data = Encrypt(memory[r2]); memory[r1] = encrypted data
DEC r1, r2	PS	1_0_1	r2(4)	r1(4)	x	0010	decrypted data = Decrypt(memory[r2]); memory[r1] = decrypted data

FFT r1, r2	LD r2, r2	fft (r2, r2)	ST r1, result
ENC r1, r2	LD r2, r2	enc (r2, r2)	ST r1, result
DEC r1, r2	LD r2, r2	dec (r2, r2)	ST r1, result

All **Address/Immediate** are unsigned, so extended with {0}

All ALU operation is signed operations

Operand Selection Mux

Operand1 Mux		**Operand2 Mux**	
Op1_Sel	*Op1*	*Op2_Sel*	*Op2*
0_0	0	0_0	0
0_1	Rout1	0_1	Rout2
1_0	PC	1_0	3
1_1	0	1_1	0

Write-Back Selection Mux

Write Back Mux	
Sel	
0	rd_WB
1	load