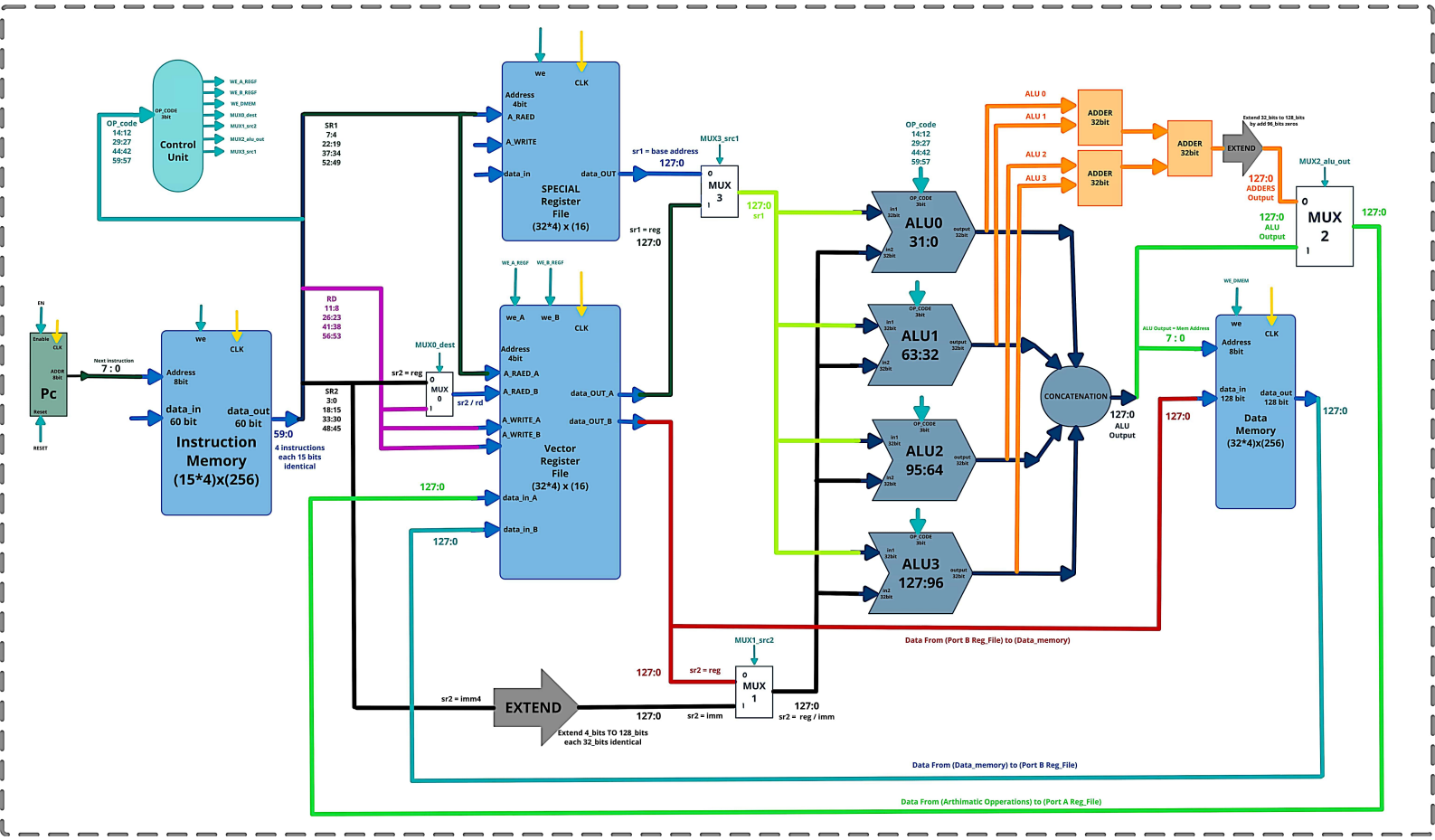
Vector Processor


Control Unit
OP_code 14:12 29:27 44:42 59:57
WE_A_REGF
WE_B_REGF
WE_DMEM
MUX0_dest
MUX1_src2
MUX2_alu_out
MUX3_src1
SR1 7:4 22:19 37:34 52:49
SR2 3:0 18:15 33:30 48:45
RD 11:8 26:23 41:38 56:53
SPECIAL Register File (32*4) x (16)
Vector Register File (32*4) x (16)
Instruction Memory (15*4)x(256)
Pc
Next instruction 7 : 0
59:0
4 instructions each 15 bits identical
sr1 = base address
sr1 = reg
sr2 = reg
sr2 / rd
sr2 = imm4
sr2 = imm
sr2 = reg / imm
EXTEND
Extend 4_bits TO 128_bits each 32_bits identical
ALU0 31:0
ALU1 63:32
ALU2 95:64
ALU3 127:96
ALU 0
ALU 1
ALU 2
ALU 3
ADDER 32bit
EXTEND
Extend 32_bits to 128_bits by add 96_bits zeros
ADDERS Output
ALU Output
CONCATENATION
ALU Output = Mem Address 7 : 0
Data Memory (32*4)x(256)
Data From (Port B Reg_File) to (Data_memory)
Data From (Data_memory) to (Port B Reg_File)
Data From (Arthimatic Opperations) to (Port A Reg_File)

Vector processor

Control signals

FORMATE TYPE	OP TYPE	Input op_code			Output controle signal						
********	******	A	B	C	WE_A _REG	WE_B _REG	WE_ MEM	MUX0	MUX1	MUX2	Mux3
Data Inst	ADD	0	0	0	1	0	0	0	0	1	1
Data Inst	MUL	0	0	1	1	0	0	0	0	1	1
		0	1	0							
		0	1	1							
Mem inst	LDR	1	0	0	0	1	0	X=0	1	X=0	0
Mem inst	STR	1	0	1	0	0	1	1	1	X=0	0
Data Inst	MOV	1	1	0	1	0	0	X=0	X=0	1	1
Data Inst	DPRO	1	1	1	1	0	0	0	0	0	1

Vector processor

Instructions format

1- Data Instruction

14 : 12	11 : 8	7 : 4	3 : 0
Op_code (3_bits)	RD (4_bits)	SR1 (4_bits)	SR2 (4_bits)

59:57	56:53	52:49	48:45	44:42	41:38	37:34	33:30	29:27	26:23	22:19	18:15	14:12	11:8	7:4	3:0
op	rd	Sr1	Sr2	op	rd	Sr1	Sr2	op	rd	Sr1	Sr2	op	rd	Sr1	Sr2

Note : mov instruction sr2 not used > put 4_zeros

2-Memory Instruction

14 : 12	11 : 8	7 : 4	3 : 0
Op_code (3_bits)	RD (4_bits)	SR1=base address (4_bits)	Imm4 (4_bits)

59:57	56:53	52:49	48:45	44:42	41:38	37:34	33:30	29:27	26:23	22:19	18:15	14:12	11:8	7:4	3:0
op	rd	Bas_a	Imm4	op	rd	Bas_a	Imm4	op	rd	Bas_a	Imm4	op	rd	Bas_a	Imm4

Vector processor

Instruction Set

OP TYPE	FORMATE TYPE	Assembly code	Machine code	description
ADD	DATA INST	ADD RD,SR1,SR2	Op_code (000) | RD (4_bits) | SR1 (4_bits) | SR2 (4_bits)	RD= SR1 + SR2
MUL	DATA INST	MUL RD,SR1,SR2	Op_code (001) | RD (4_bits) | SR1 (4_bits) | SR2 (4_bits)	RD= SR1 * SR2
DPRO	DATA INST	DPRO RD,SR1,SR2	Op_code (111) | RD (4_bits) | SR1 (4_bits) | SR2 (4_bits)	RD= SR1 . SR2
MOV	DATA INST	MOV RD,SR1	Op_code (110) | RD (4_bits) | SR1 (4_bits) | SR2 (0000)	RD= SR1
LDR	MEM INST	LDR RD,[SR1,#IMM4]	Op_code (100) | RD (4_bits) | SR1 (4_bits) | IMM4 (4_bits)	RD=DATA_MEM[SR1+IMM4]
STR	MEM INST	STR RD,[SR1,#IMM4]	Op_code (101) | RD (4_bits) | SR1 (4_bits) | IMM4 (4_bits)	DATA_MEM[SR1+IMM4]=RD

Vector processor

Summary

➢ Main module (vertex_processor)

- vector processor to processing 4-instructions identical each one 15-bits on 32-bits different data.

➢ sub modules

- **Program counter (pc)**
  - 8-bits counter to determine next instruction.
- **Instruction memory (instruction_mem)**
  - ram memory ,data bus 60-bits and address bus 8-bits Containing instruction data.
- **Special register file (sreg_file)**
  - ram memory ,data bus 128-bits and address bus 4-bits Containing memory addresses.
- **dual port vector register file (vreg_file)**
  - dual port ram memory ,data bus 128-bits and address bus 4-bits Containing 4-processing data each one 32-bits "each register is 128-bits".
- **data memory (data_mem)**
  - ram memory ,data bus 128-bits and address bus 8-bits to (read from \ write to) it data.
- **arithmetic logic unit (alu)**
  - 4-alus operate in parallel ,each one is 32-bits data bus, 3-bits op_code identical for all ,then concatenate outputs (128-bits alu output).
- **control unit (control_unit)**
  - contain only one input (3-bit opcode) ,7 outputs (control signals), to determine what is the type of instruction format and what is the type of operation is executed.