DRAFT: Do not distribute

MIT Concurrent VLSI Architecture Memo ??

Massachusetts Institute of Technology
Artificial Intelligence Laboratory

The Architecture of RR Stage of the MAP Pipeline
Draft Version

Andrew Chang

February 14, 1995

Abstract

This document describes the operation, architecture, and implementation of the Register Read stage (RR) of the multi-ALU processor (MAP) pipeline. This document should be read in conjunction with *M-Machine Architecture v1.1* and *M-Machine Microarchitecture v1.0*.

Contents

1 Functional Overview **3**

1.1 Function . . . 3

2 Interfaces **6**

2.1 IF/RR Interface . . . 6

2.2 RR/SZ Interface . . . 7

2.3 RR/EX Interface . . . 9

2.4 RR/WB Interface . . . 9

2.5 RR/CSW Interface . . . 9

2.6 RR/CFG Interface . . . 9

3 Operation **10**

3.1 Register File . . . 13

3.1.1 Registers . . . 13

3.1.2 General Registers . . . 15

3.1.3 Special Registers . . . 15

3.2 Scoreboard . . . 16

3.3 Operation Predecode . . . 17

3.4 SEND . . . 17

4 RTL **21**

5 Timing **23**

6 Area Estimate **23**

6.1 Register File . . . 23

6.2 Scoreboard . . . 23

6.3 OPSZ Registers . . . 26

6.4 Decode Logic . . . 26

7 Power Consumption **26**

8 Open Issues **26**

8.1 Logical Design . . . 26

8.2 RTL Implementation 28

9 Logic Design Review Action Item List **28**

10 RTL Design Review Action Item List **29**

A MAP Instruction Encodings **31**

B Verilog Source Files **31**

1 Functional Overview

The RR stage of the MAP pipeline performs instruction predecode, scoreboard updates and register reads. The RR stage is responsible for providing the source operands for each operation. For each of the three execution units, the RR stage performs the same functions. However, the implementation of of the RR stage varies to accommodate the differences between execution units.

Figure. 1 shows an overview of the MAP chip pipeline. The pipeline for the MAP chip is composed of five stage:

1. Instruction Fetch (IF) - Responsible for fetching and unpacking instructions, managing the ICache and managing the Instruction Pointers (IPs).
2. Register Read (RR) - Responsible for decoding instructions, performing sign extension on immediates, providing source operands, and managing the Register Files and Scoreboards.
3. Synchronization (SZ) - Responsible for selecting instructions to issue, arbitrating for the Network resource, and generating the thread select (TSEL) control signals.
4. Execution (EX) - Responsible for executing operations, detecting exceptions, arbitrating for CSW and MSW resources, stalling the pipeline, validating and invalidating registers, and creating Errval's.
5. Writeback (WB) - Responsible for writing results to the local register file, for writing results to the CSW for remote destinations and updating the scoreboard.

Figure 2 shows the high level logical block diagram of the RR stage. Logically, the MAP pipeline contains one shared RR function for all the execution units. The actual implementation will be distributed. The INTU RR stage will have some added functionality to handle global issues.

1.1 Function

Specifically, the RR stage is responsible for the following:

- Read the appropriate Register Files Based on Operation
- Multiplex Input Sources (including Bypass paths) to OPSZ Register
- Perform "Predecode" of operation field
- Set Valid Bits in OPSZ Register

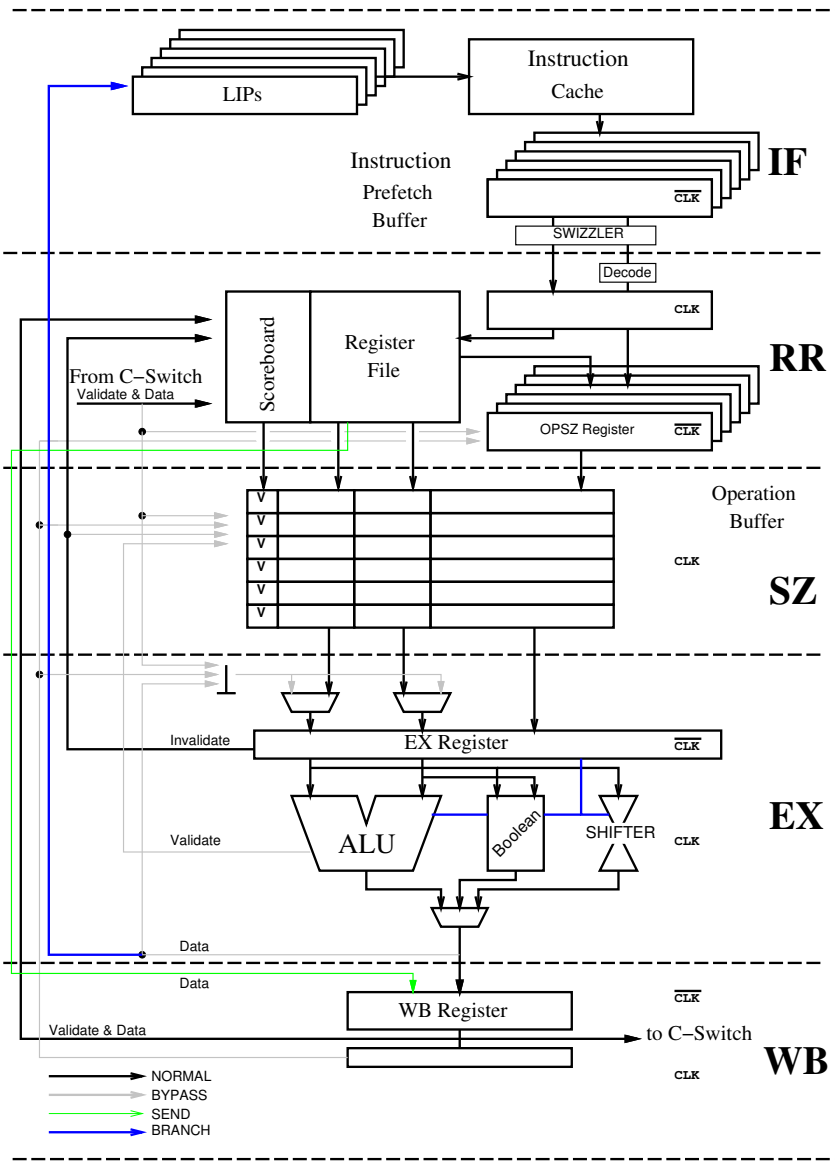


Figure 1: *The MAP Pipeline*

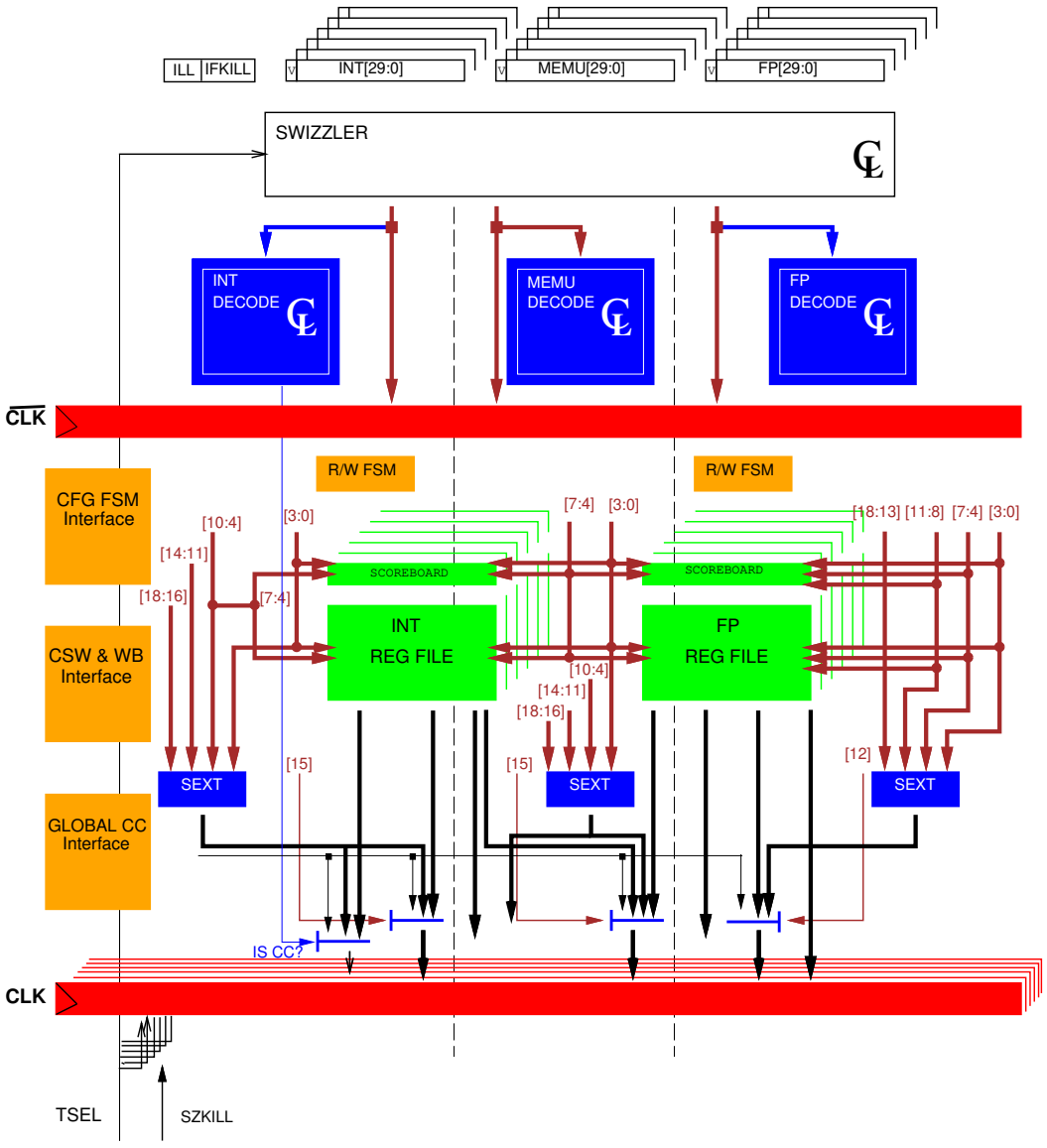


Figure 2: *RR Logical Overview*

- Perform Sign Extension on Immediates
- Destination for validation/invalidation of Scoreboard Entries
- Destination for updates to Register File Data
- Provide message data from the Register Files to the WB stage during SEND operations.

2 Interfaces

The RR stage has six interfaces which are itemized below and detailed in the subsequent figures.

1. Instruction Fetch (IF/RR) - Operation Fields for INTU, MEMU FPU

   160 bits x 6 Registers = 960 bits

2. Synchronization (RR/SZ) - OPSZ Pipeline Register for Operations

   752 bits x 4 Registers + 756 x 2 Registers

3. Execution Stage (RR/EX) - Invalidation and Register addresses for SENDs

   29 bits

4. Write Back Stage (RR/WB) - Source of Validation/Invalidation and Data during normal operations. Destination for register data during SENDs

   294 bits

5. Cluster Switch (RR/CSW) - Validation and Data

   TBD

6. Configuration Space FSM (CFG/RR) - Pseudo-Operations

   TBD

2.1 IF/RR Interface

Figure 3 presents the bit fields of the IF/RR Interface. The IF stage passes the 30-bit operation for each execution unit to the RR stage (32 bit operation less the 2 packing bits). In addition, the IF stage generates a valid bit (VAL) for each operation as well as a late kill line (IFKILL). and an Illegal Bit (ILL). The TSEL lines are bussed to all the pipeline stages. Finally, the IF stage provides the corresponding IP with the instruction.

The following combinations have meaning:

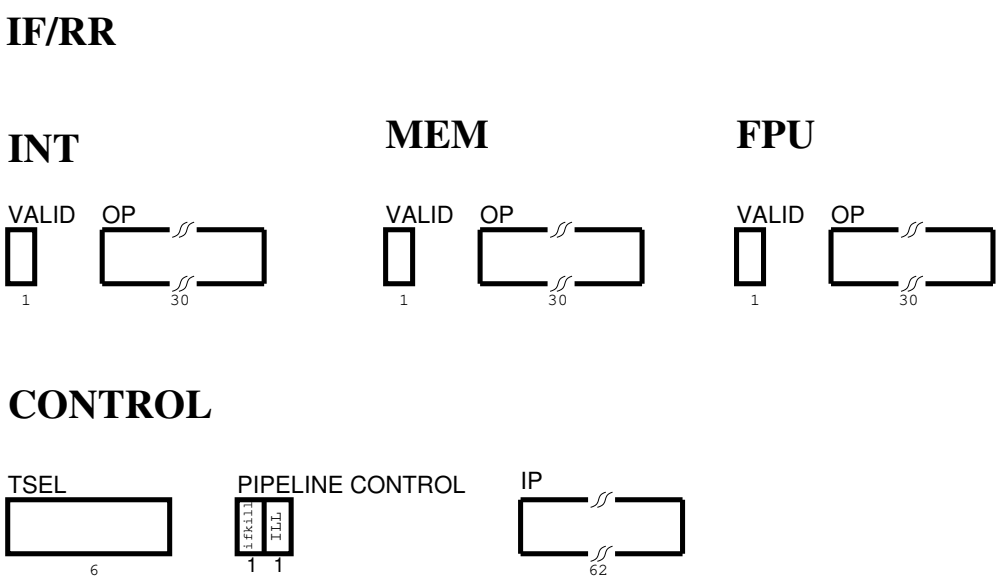


Figure 3: *IF/RR Interface*

- VAL & $\overline{IFKILL}$ = Non-NOP operation for the function unit.
- $\overline{VAL}$ & $\overline{IFKILL}$ = NOP operation for the function unit.
- ILL = Illegal operation caused by branch to ErrVal

2.2 RR/SZ Interface

Figure 4 presents the bit fields of the RR/SZ Interface. The interface drawing details the OPSZ pipeline register format. There are five 752-bit registers, one each for the four user threads and the exception thread. The system thread has a 756-bit since it has 2 additional MC-Present lines from the INT register file and the FP register file. The detailed breakdown and count for each function unit is detailed in the figure.

The OPSZ pipeline register is composed of the following fields:

- Decoded Operation
- Source Operand Addresses, Sync status, and Values
- Destination Register Addresses
- Send Length and accompanying MC register status
- MBAR flag indicating that a CFG pseudo operation should decrement the MBAR counter

RR/SZ

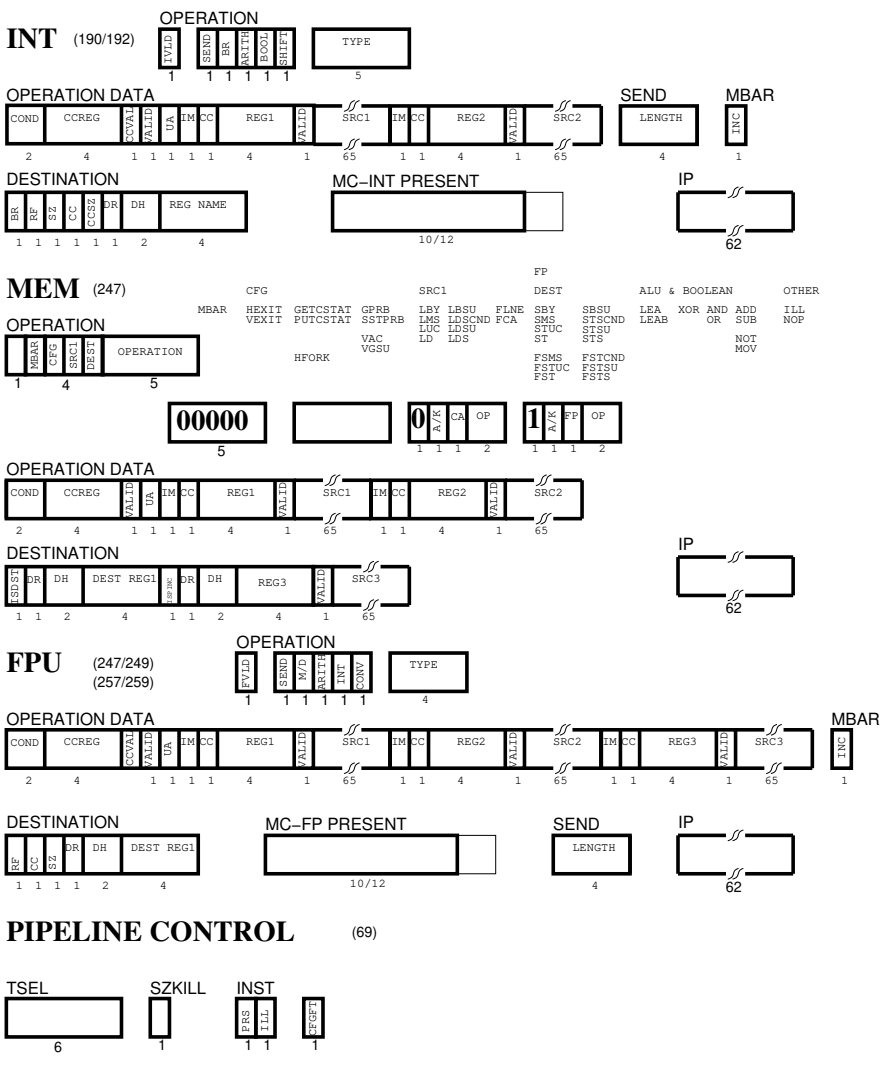


Figure 4: *RR/SZ Interface*

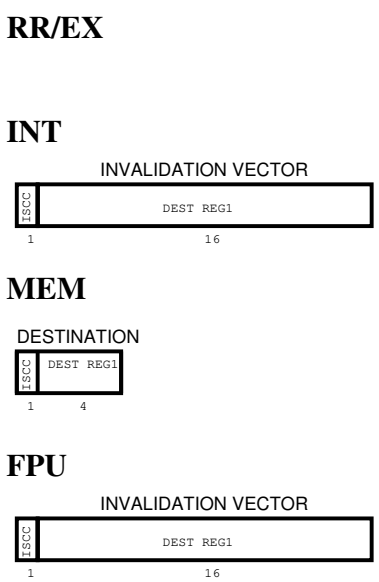


Figure 5: *RR/EX Interface*

2.3 RR/EX Interface

Figure 5 presents the bit fields of the RR/EX Interface. The RR/EX interface is used to invalidate destination registers in the scoreboard and to bypass invalidations to the OPSZ buffer for EMPTY and ISND in the INTU and for cycle operations in the MEMU and FPU.

2.4 RR/WB Interface

Figure 6 presents the bit fields of the RR/WB Interface. The RR/WB interface is used to write data and validate of destination registers in the register file and the scoreboard, to bypass data and validation to the OPSZ buffer.

The READ bit signals the direction of data travel over the interface: to the Register Files during normal operation, from the Register Files during SENDs.

2.5 RR/CSW Interface

The RR/CSW interface remains to be defined.

2.6 RR/CFG Interface

The RR/CFG interface remains to be defined.

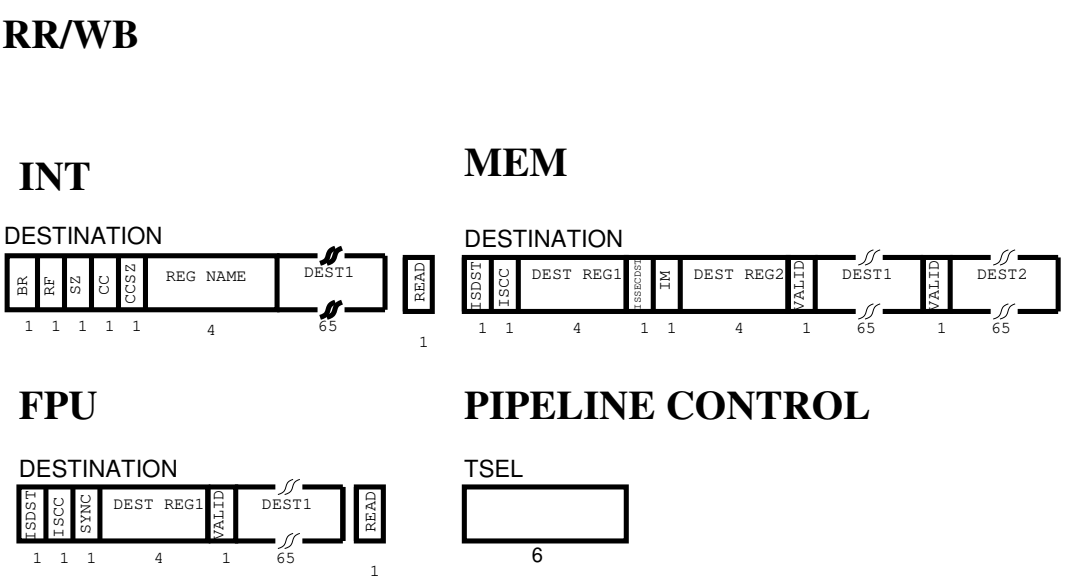


Figure 6: *RR/WB Interface*

3 Operation

The RR stage performs "predecode" for each instruction, multiplexes the required inputs sources, manages the register file and the scoreboard. The integer unit RR stage is detailed in figure 7.

Figure 8 details the possible sources for operands in each integer unit operation.

M-Machine instructions follow the standard three address format. The instruction bit encodings ensure that SRC1, SRC2, and the Immediate indicator always reside in the same bit positions. DEST is in the same location for all operations except SHORU, where it is actually in SRC1.

Therefore, the RR stage *always* attempts to read the register addresses indicated by the SRC1 and SRC2 bit fields. In the case of single source instructions or instructions using immediates, the proper operand is multiplexed in before being latched into the OPSZ Register and the values resulting from the read are discarded. This policy was chosen because the Register File access time is in the critical path and any logic added to determine whether to perform a register read would directly add to the stage delay.

During normal operation, the RR stage receives an instruction from the IF stage based on the value of the thread select lines (TSEL) asserted by the SZ stage. Each instruction is composed of the following:

- Three 30 bit operations (INT, MEM, FPU).
- Three (3) valid bits (one for operation) indicating whether there is an actual operation

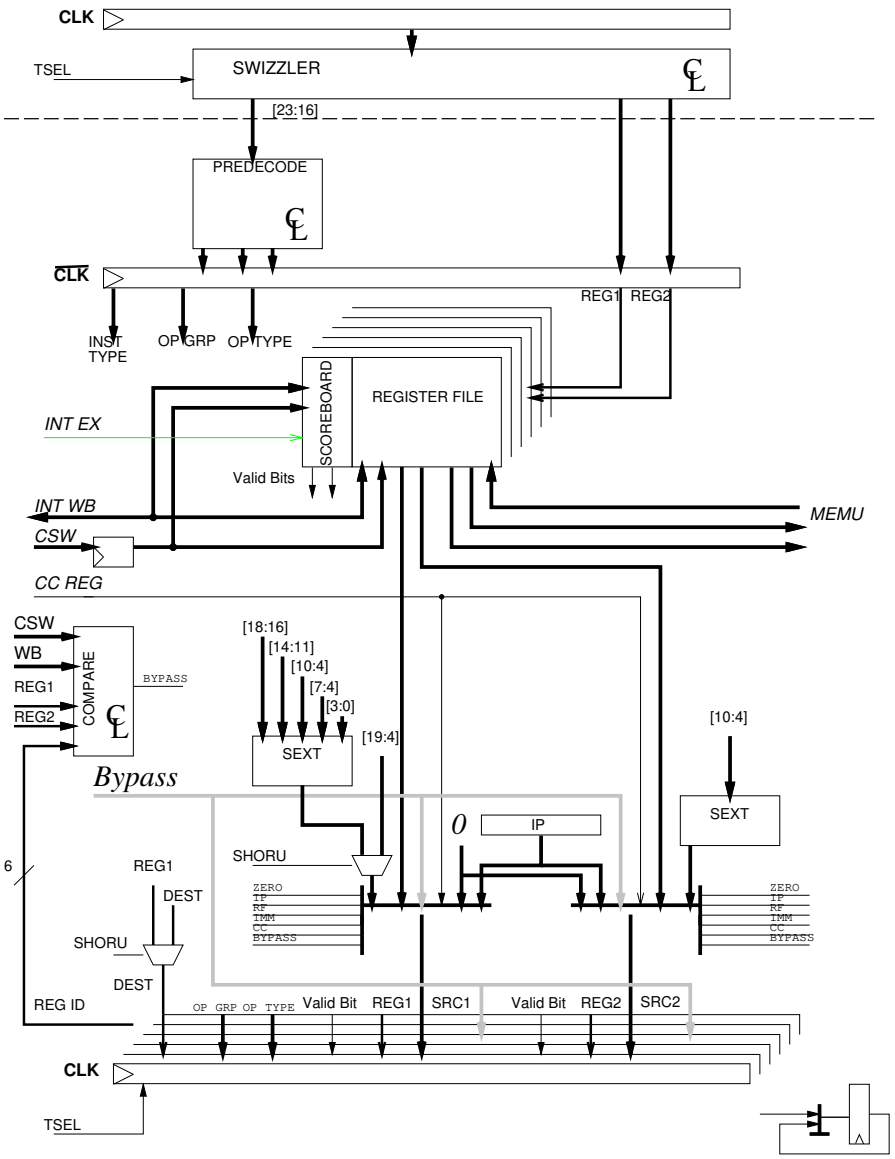
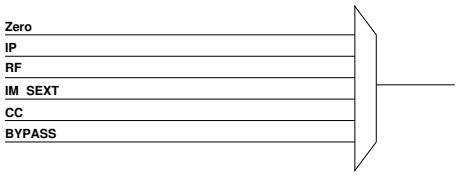


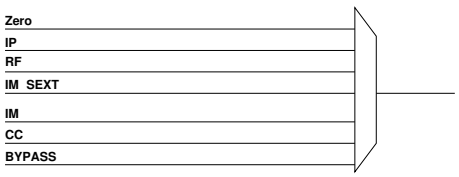
Figure 7: *IU RR Logical Block Diagram*

SRC1



Zero =	i0 + IMM
IP =	i1
RF =	i[15:2]
IM SEXT =	b[15] • (TYPE B + TYPE G)
CC =	TYPE E
BYPASS =	BYPASS

SRC2



Zero =	i0
IP =	i1 + BR
RF =	i[15:2]
IM SEXT =	b[15] • (TYPE A1 + TYPE C + TYPE P)
IM =	SHORU
CC =	TYPE E
BYPASS =	BYPASS

Figure 8: *Input Sources*

for the relevant execution unit.

- Two (2) control bits indicating Illegal instruction and IF late kill
- Associated Instruction Pointer IP

The RR stage *always* performs each of the following tasks to gather the required operands for an instruction:

- Reading the Register File
- Performing Sign Extension on the bits in the immediate field of the instruction.

The RR stage decodes the 5-8 Bit operation field to determine whether to use the Register File Values, sign-extended immediate, unsigned immediate, IP, Condition Code Register Values, or Zero as input operand value. In addition, bypassed data and validation information from the CSW, EX, or WB stages are also multiplexed into the OPSZ register.

The RR stage maintains 6 OPSZ Buffers, one for each thread. The OPSZ buffers are bit interleaved. The SZ stage asserts TSEL to select operations for the desired thread.

Timing: The RR stage employs a *clk* transparent latch between the output of the IF Swizzler and the Register File. Most of the Instruction Decode is performed before this latch. The results from the Register File are latched into the OPSZ buffer by a $\overline{clk}$ transparent latch.

3.1 Register File

The M-Machine architecture provides each VThread with a unique integer register file and a unique floating point register file. Thus, each cluster of the M-Machine contains six (6) integer register files and six (6) floating point register files. The four user threads (A,B,C,D) and the system thread (S) each have 14 integer registers and 15 floating point registers. The exception handling thread (X) has only 5 integer and 4 floating point registers. As shown in figure 9, each integer register file has 4 read ports and 3 write ports and each floating point register file has 4 read ports and 2 write ports.

3.1.1 Registers

Note: This section is reproduced from M-Machine Architecture v1.1 .

The register state accessible within a cluster consists of:

- General registers

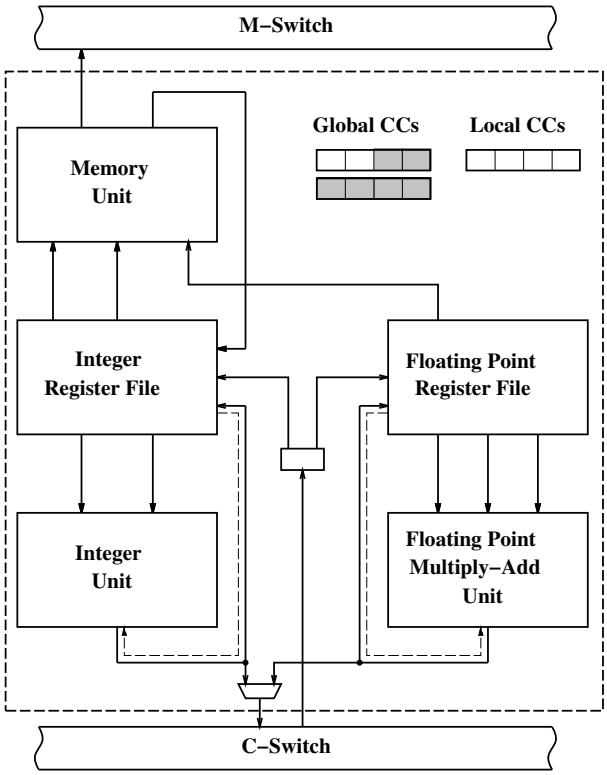


Figure 9: *Register File Overview*

- Special registers
- Condition registers

3.1.2 General Registers

Each H-Thread sees 14 integer registers and 15 floating point registers in its own cluster. The integer register file has 4 read ports and 3 write ports. Within a single cycle, the integer unit and the memory unit can each read two operands. In addition, both units and the C-Switch may write a result. The floating-point register file has 4 read and 2 write ports The floating point unit may read 3 operands and write a result. The memory unit may read 1 operand and write a result.

The general registers are divided into two portions *public* and *private*. The private registers reside in the lower half of the register file [i7:i0], [f7:f0] , while the public registers reside in the upper half. Private registers may only be written by operations executed within the cluster. Any communication between clusters must use the public registers. The hardware enforces this separation; an attempt to writes to a remote private register results in a write to the corresponding remote public register.

3.1.3 Special Registers

The special registers reside in the same namespace as the general registers. Table 1 specifies the special register names and meanings. Explanations are as follows:

`Zero` : Integer register 0 (*i0*) in all clusters is mapped to zero. Reading *i0* gets a zero value, while writing *i0* has no affect on the state of the register file. Register *f0* holds floating point zero, but otherwise has the same behavior as *i0*.

`IP` : The Instruction Pointer register points to the instruction that the cluster is executing. This register may not be written directly, but is modified via *branch*, or *jump*. Writing directly to *i1* has no affect on the state of the register file.

`HEAD` : The dedicated system slot on each cluster has an associated *event* or *message* queue. The first word of the next message is accessed by reading integer register 14. This flushes any remaining words from the current event or message.

`BODY` : This is mapped to integer register 15 and accesses the next word in the message or event queue. If the next word is the first word of the next message or event, *Errval* is returned. User threads may access *i14* and *i15* as a general registers.

`MC` : Each thread has 10 special registers used for message composition. Outgoing messages are composed in these registers and then launched using a *send* instruction. These registers may also be used by the programmer as general registers.

Cluster	Register	Meaning
all	i0	Always zero. Writes to i0 disappear.
all	f0	Always fp zero. Writes to f0 disappear.
all	i1	Instruction Pointer. Writes to i1 disappear
all	i14	Head (System V-thread, Event Cluster only).
all	i15	Body (System V-thread, Event Cluster only).
all	i4–i13	Message composition (MC).

Table 1: Special registers.

3.2 Scoreboard

The integer unit scoreboard is physically located in the RR stage. The scoreboard maintains the presence bits for each register in the integer register file.

The Scoreboard maintains the value of the presence bits for each register and is updated in the following manner:

- Normal Operation: the WB stage of the Pipeline writes data and validates the presence bits during the first half cycle (Clock High) of the master clock *clk*. The data is latched at clock low and the values that were written can be read in the second half of the same clock cycle.
- Normal Operation: the EX stage of the Pipeline invalidates the presence bits during the first half cycle (Clock High) of the master clock *clk*. The data is latched at clock low and the values that were written can be read in the second half of the same clock cycle.
- Normal Operation (MOV): the WB stage of the remote cluster writes the presence bits via the CSW during the first half cycle of the master clock. Data follows in the second half cycle (Clock LOW) of the master clock *clk*. The data is latched at clock high of the subsequent period.
- Bypass (EX invalidation to OPSZ buffer): the EX writes presence bits during the first half cycle of the master clock *clk*. The OPSZ buffer latches these values with all other validation/invalidation requests.
- Bypass (LD returning over CSW): the CSW writes presence presence bits during the first half cycle of the master clock *clk*. These values are latched during the second half cycle. The data over the CSW is provided during the second half cycle and is bypassed directly to the input of the OPSZ and EX pipeline stages if the appropriate register address matches.

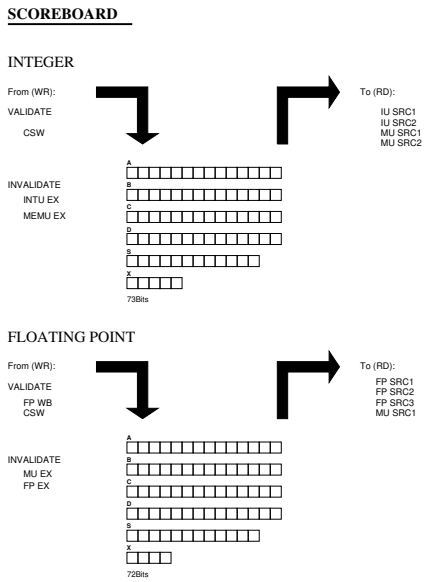


Figure 10: *Scoreboard Logical Overview*

3.3 Operation Predecode

Integer Unit Operations The integer unit performs 41 different operations. The RR stage decodes the 5-8bit operation field for each integer operation to determine which input sources are needed for each instruction. Also, the RR stage generates a five (5) bit group identifier and a five (5) bit type identifier for use by the execution unit.

3.4 SEND

When a SEND operation is issued, the Integer EX stage stalls the pipeline until the SEND completes. This decision was made to reduce design complexity and eliminate the need to add an additional Read Port on each register file. In both the INTU and FPU the SEND FSM in the EX stage generates the SEND control signal. Each WB stage then asserts READ and provides the required register addresses to the corresponding register file to access the Message Composition Registers (MC) using the existing RR/WB wires. The data values are transmitted to the corresponding WB stage by indicating "READ" and "turning around" and reusing the data wires of the RR/WB interface.

Figure 12 shows the SEND FSM.

Figure 13 shows the SEND timing.

INT

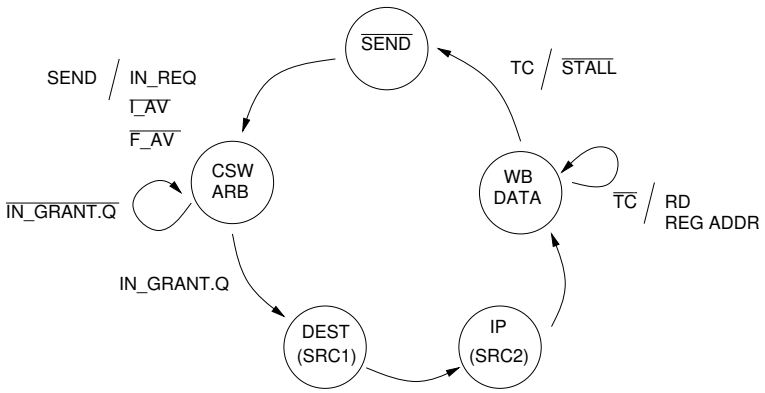
GROUP: SEND | BR | ARITH | BOOL | SHIFT

TYPE: [ ] [ ] [ ] [ ] [ ]

SEND	X	[GTLB]	[Pri]	[Phys]	[~INC]		5
BR	X	X	X	[TRIP]	[JMP]		2
ARITH	PTR	NEQ	SIGN	EQ	$\overline{\text{SRC2}}$		11
	0	0	1	0	0	ADD	
	0	0	0	0	0	ADDU	
	0	0	1	0	1	SUB	
	0	0	0	0	1	SUBU	
	0	0	1	0	1	ILT	**CC DEST**
	0	1	1	1	1	ILE	
	0	0	0	0	1	ULT	
	0	1	0	1	1	ULE	
	0	0	1	1	1	IEQ	
	0	1	1	0	1	INE	
	1	0	1	0	0	LEA	
	1	0	1	0	1	LEAB	
BOOL	MOV	PTR	XOR	NOT	AND/$\overline{\text{OR}}$		12
	0	0	0	0	0	OR	
	0	0	0	0	1	AND	
	0	0	0	1	0	NOT	
	0	0	1	0	0	XOR	
	0	0	0	0	0	CCOR	**CC DEST**
	0	0	0	0	1	CCAND	
	0	0	0	1	1	CCNAND	
	1	0	0	0	0	MOV, EMPTY, CCEMPTY	
	0	1	0	0	0	SETPTR	
	0	1	0	0	1	ISPTR	**CC DEST**
	0	1	1	0	0	ISERR	
SHIFT	OR	H/$\overline{\text{B}}$	W	ROT	LG/AR		9
	0	0	1	0	0	ASH	
	0	0	1	0	1	LSH	
	0	0	1	1	1	ROT	
	0	0	0	0	1	EXTB	
	0	1	0	0	1	EXTH	
	1	0	0	0	1	INSB	
	1	1	0	0	1	INSH	
	1	0	1	0	1	SHORU	
	1	0	1	0	1	IMM	

Figure 11: *Integer Operation Predecode*

LOGICAL



IMPLEMENTATION

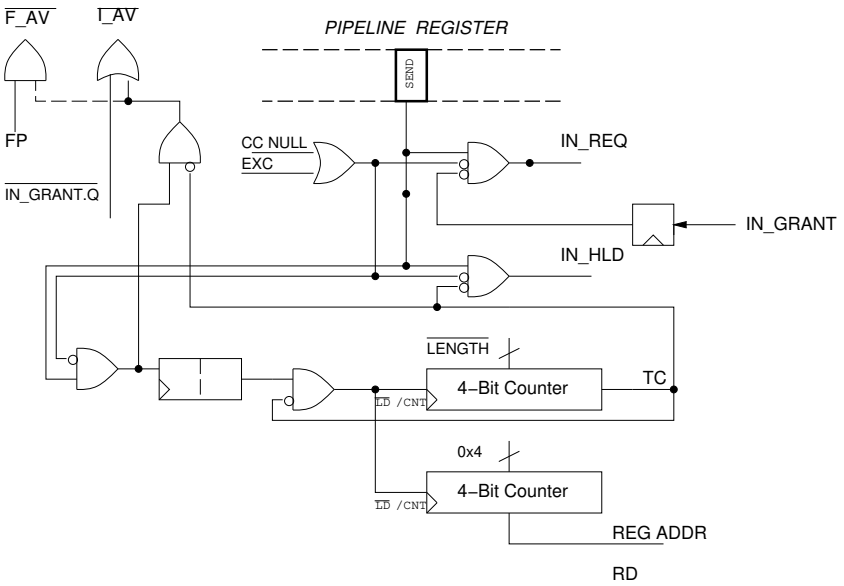


Figure 12: *EX SEND State Diagram and Implementation*

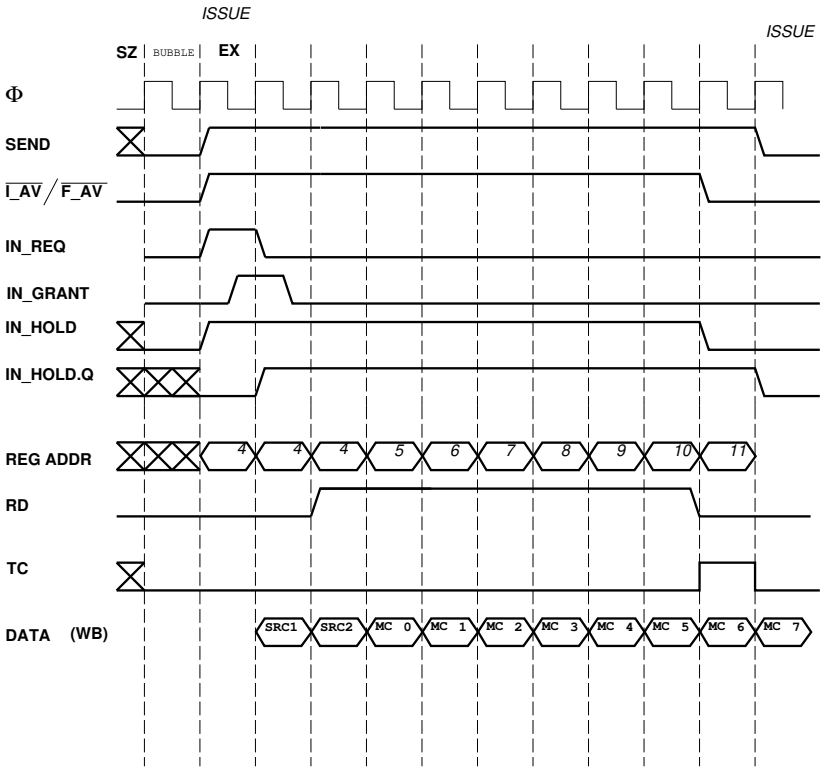


Figure 13: *SEND Timing*

4 RTL

A RTL module view of the INTU RR Stage is provided in figure 14.

The INTU RR stage is composed of the following 15 modules:

- Control Logic
  - IDEC Operation decode
  - ISEL Operand select
  - ISXT Sign extension and immediate generation
  - IDST Destination decode and selection
  - IVAL Validation/invalidation
  - CC Conditon code control
  - IBYP Bypass control and select
  - ICFG Config Space Interface
  - ICSW C-Switch Interface
  - ISND Send control
- Datapath
  - IRF Integer Register File (7-ports, 4 read, 3 write)
  - ISC Integer scoreboard
  - CCRF Condition code register file and valid bits
  - IMUX Operand Multiplexers
  - IOPSZ Integer portion of the OPSZ Pipeline Registers

OPDEC The *opdec* module is a combinational logic block which takes operation bits (op[23:16]) from the output of the IF stage and decodes the instruction type as defined in the instruction encoding appendix. The instruction type information (9 bits) is one hot encoded and is used subsequently by the OPSEL, SEXT, DEST modules. The *opdec* module also translate the operand validity (iv, mv, fv) and late kill (ifkill) signal from the IF stage into valid and present bits for the downstream pipeline stages.

OPSEL The *opsel* module is a combinational logic block which takes the operation bits op[23:0] from the output of the IF stage, the instruction type vector from *opdec* and generates a "one hot" encoded operand select vector for each source operand.

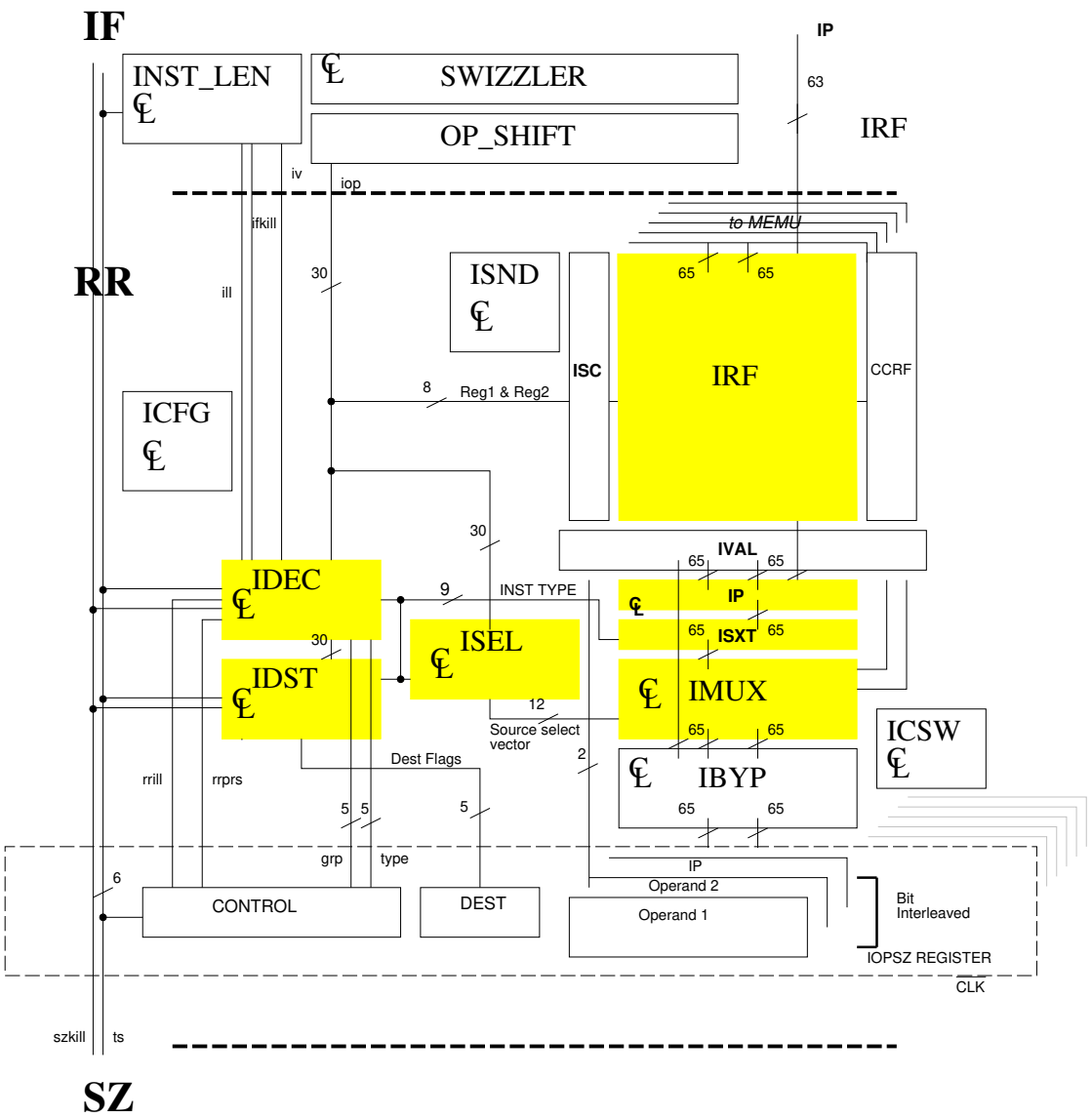


Figure 14: *INTU RR Module Diagram*

SEXT The *sext* module is a combinational logic block which takes the operation bits op[23:0] from the output of the IF stage, the instruction type vector from *opdec*, the operand select vector from *opsel* and generates the requested 65-bit constant: unsigned immediate, sign-extended immediate, IP or zero.

DEST The *dest* module is a combinational logic block which takes the operation bits op[23:11] from the output of the IF stage, and the instruction type vector from *opdec* and generates a "one hot" encoded destination vector indicating: branch, register file, scoreboard, cc register file, cc scoreboard. The *dest* module *deasserts* bits in the vector for *NO* destination. No destination is asserted when the operation has no destination (ILL,NOP) or when the destination is suppressed (writes to i0, i1, f0).

OPMUX The *opmux* module is a datapath module containing the multiplexors to select between register file data, *sext* generated immediate and CC values. The *opmux* module is driven by the source selection vector generated by the *opsel* module.

5 Timing

Figure 15 shows the RR stage critical timing for a normal integer add operation.

Figure 16 shows the RR stage critical bypass timing.

6 Area Estimate

6.1 Register File

The schematic for a single 7-port register cell is shown in figure 17. The preliminary area estimate for this cell is 7.6Kλ^2 (473μm^2). This estimate is based on a 60μm^2 cell size and 8λ wire pitch (2μm).

The anticipated area for each 14 register integer file is 6.92Mλ^2. The anticipated total area required for all integer register files is 37.1Mλ^2.

6.2 Scoreboard

The anticipated area for each scoreboard is 0.22Mλ^2. The anticipated total area required for all integer scoreboards is 1.2Mλ^2.

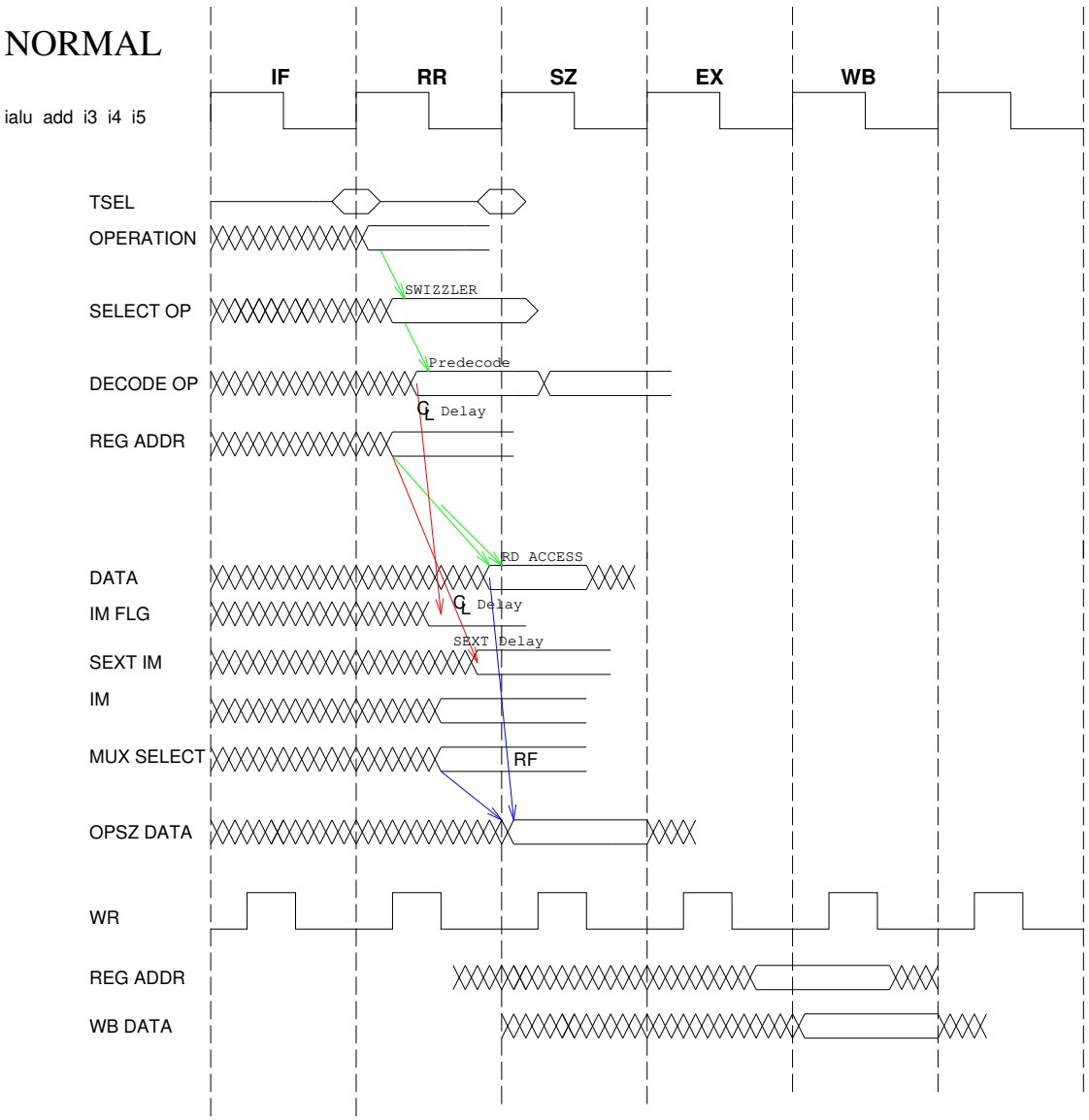


Figure 15: *Normal Operation Timing for RR Stage*

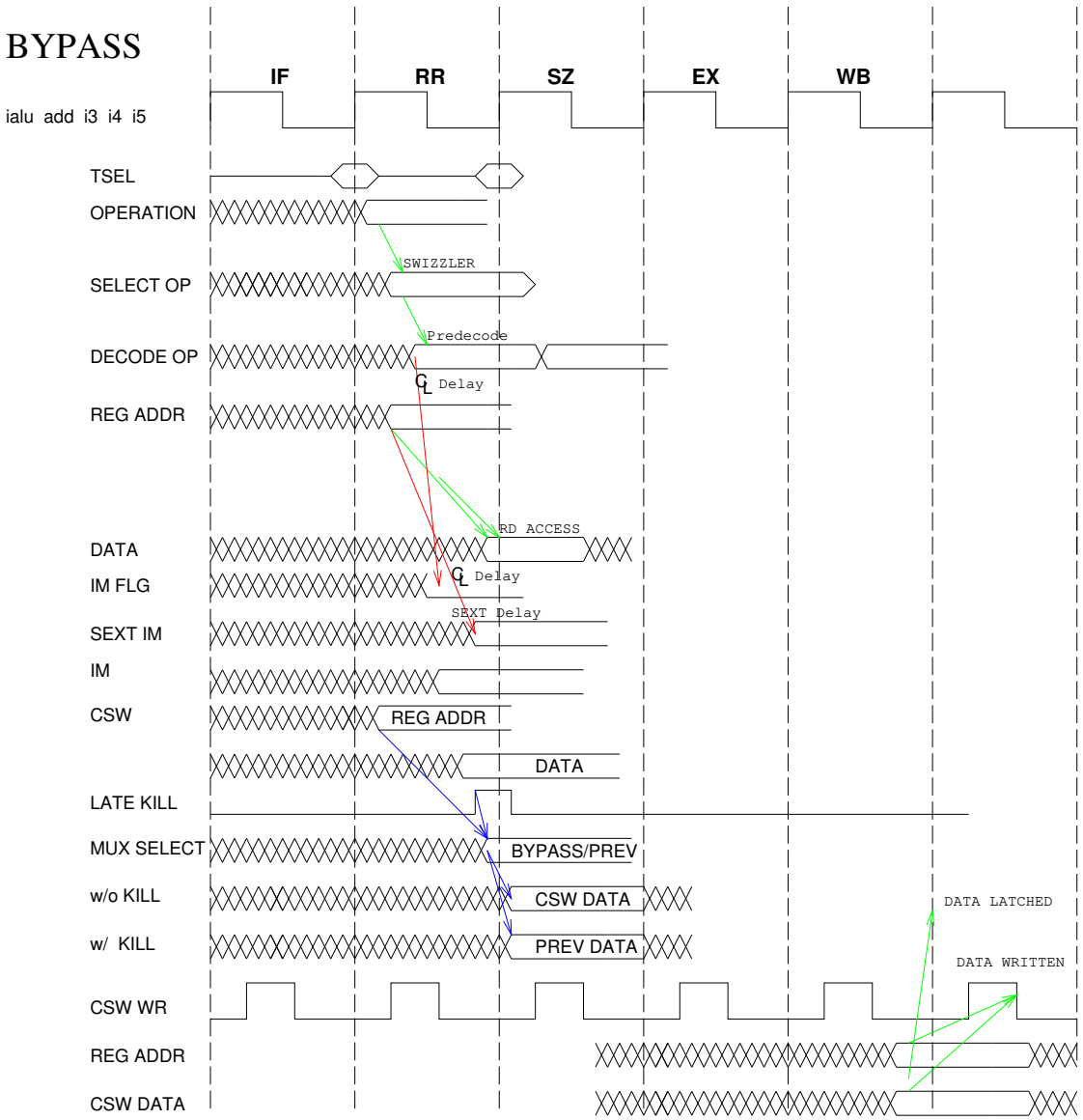


Figure 16: *Bypass Operation Timing for the RR Stage*

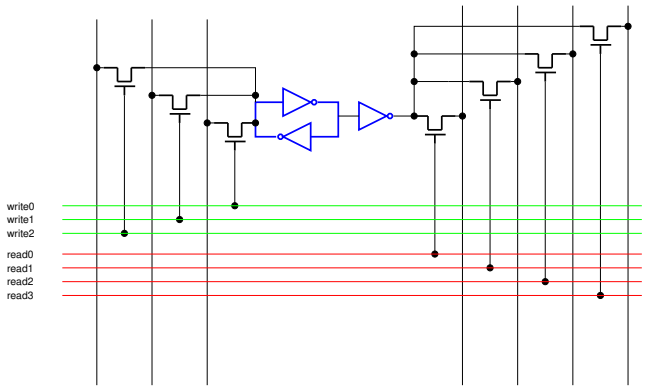


Figure 17: *Register Cell Schematic: 7 Ports*

6.3 OPSZ Registers

The anticipated area for the 6 registers is 7.296Mλ^2. The anticipated area required for bit line input multiplexors is 0.97MMλ^2. The anticipated area required for bit line output multiplexors is 0.73MMλ^2.

The total anticipated area required for the OPSZ pipeline registers is 9Mλ^2. A preliminary floorplan for the RR stage is presented in figure 18

6.4 Decode Logic

7 Power Consumption

8 Open Issues

The following issues remain to be resolved:

8.1 Logical Design

1. Physical location of Local and Global CC Registers
2. Signal distribution for Local and Global CC Registers
3. The definition of the RR/CFG interface

RR

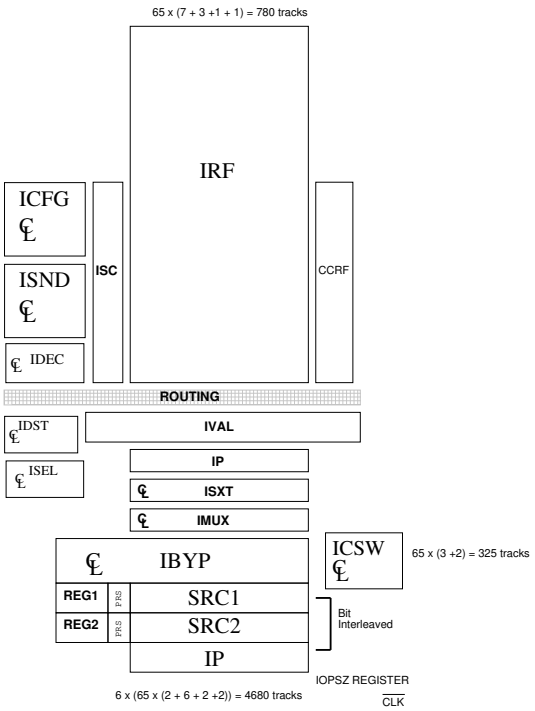


Figure 18: *Preliminary OPSZ Floorplan*

8.2 RTL Implementation

1. Finish MEMU and FPU sections of opdec.v, opsel.v, sext.v, dest.v and opmux.v
2. Finish remaining modules: byp.v, opsz.v, val.v, send.v, csw.v cfg.v.
3. Integrate and test completed RR stage

9 Logic Design Review Action Item List

The following action items resulted from the logic design review:

1. Explicitly work out bypass paths and timing to determine number and sense of required latches. Verify that "it all works".
   STATUS: **Remains to be done**
2. Decide pipeline clocking convention
   STATUS: **Done**, clk = pass on high, $\overline{clk}$ = pass on low.
3. Nuke RD line
   STATUS: **Done**
4. Nuke TSEL for bottom part of pipeline
   STATUS: **TSEL needed to for INTU to generate bs for IF**
5. Nuke MEMU WB validate for INT Scoreboard
   STATUS: **Done**
6. Added FSND instruction
   STATUS: **Done**
7. Added RR logic to coerce ISND, FSND combination into ISND only.
   STATUS: **Done**
8. Add separate IP pointers back into EACH EX stage
   STATUS: **Done**
9. Multiplex access of SRC1 and SRC2 in INTU/FPU SEND controller
   STATUS: **Done**
10. Move bypass mux's to front of EX stage
    STATUS: **Done**

11. Show clocking of REG FILE

    STATUS: **Remains to be done**

12. Show clocking of REG FILE

    STATUS: **Remains to be done**

13. Add path from EX to IF for Branch

    STATUS: **Done**

10 RTL Design Review Action Item List

The following action items remain to be completed in the RTL implementation:

1. Use more realistic timing delays (n*'TGPD, where n = number of levels) instead of just 'TGPD.

   STATUS:

2. Examine RR bypass logic to see if it can be centrally grouped with SZ Bypass logic - specifically CSW interface.

   STATUS

3. Replace all "?" multiplexors with library MUX version.

   STATUS:

4. Decide on Register File prototype.

   STATUS:

5. Zero out SRC1 base for LEAB in opmux.

   STATUS:

6. Change RR-SZ interface drawing to reflect three IP's.

   STATUS:

7. Use "OR" instead of "MUX" for final type field select. in opdec.v

   STATUS:

8. Add tDEC and tAC (decoder and access time) constants to map_consts.vh

   STATUS:

9. Separate out the description and count of interface wires from register bits in Section 2 for RR/SZ.

   STATUS:

10. Fix extra comma in IDST inputs list.
    STATUS:

A MAP Instruction Encodings

Figure 19 presents the current Integer Unit operation encodings.

Figure 20 presents the current Memory Unit operation encodings.

Figure 21 presents the current Floating Point Unit operation encodings.

B Verilog Source Files

See attached files:

opmux.stim

opdec.v

opsel.v

opsext.v

opdest.v

opmux.v

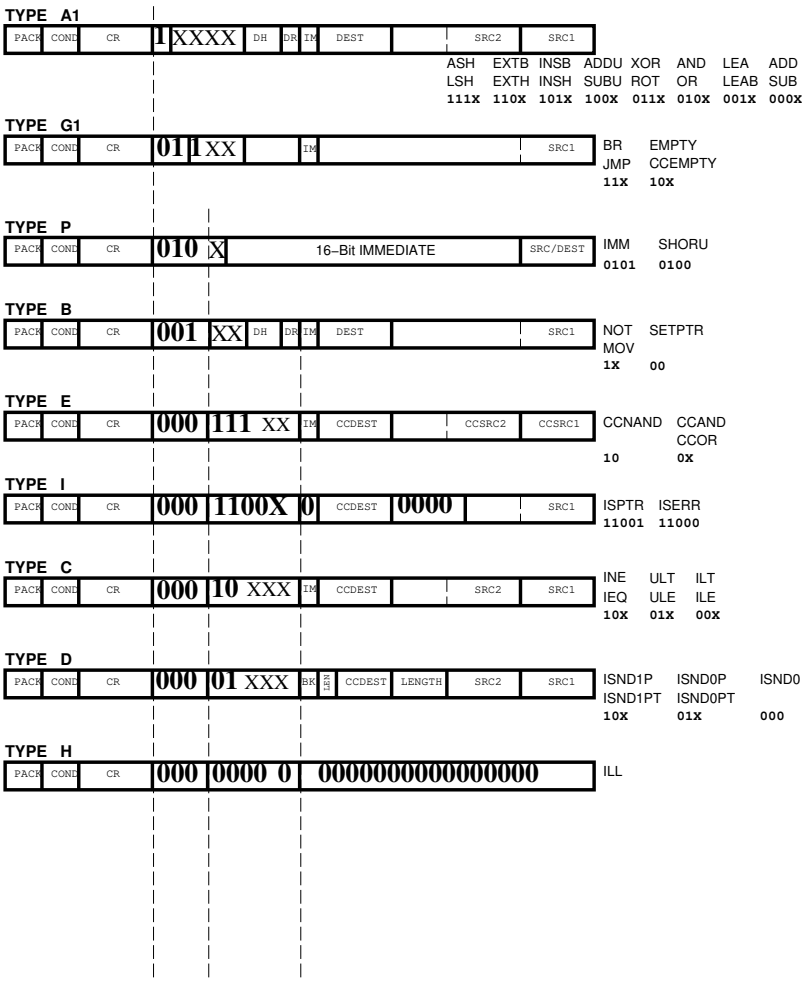


Figure 19: *Integer Unit Operations: 41*

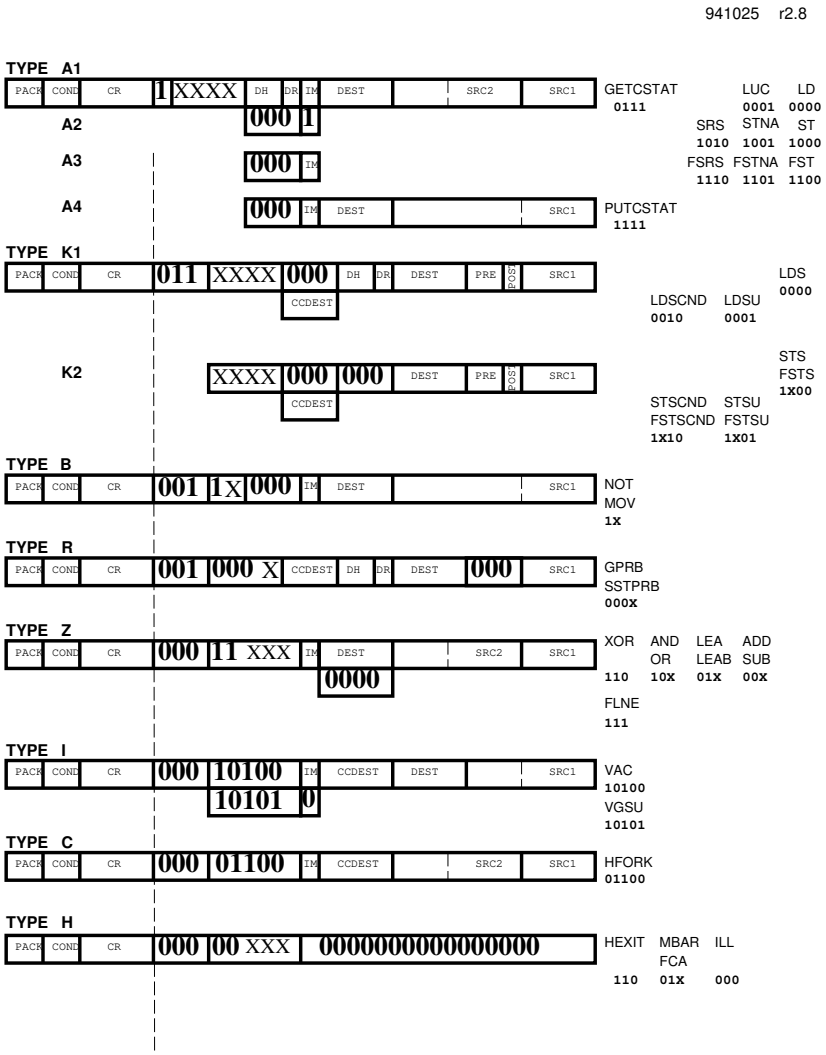


Figure 20: *Memory Unit Operations: 39*

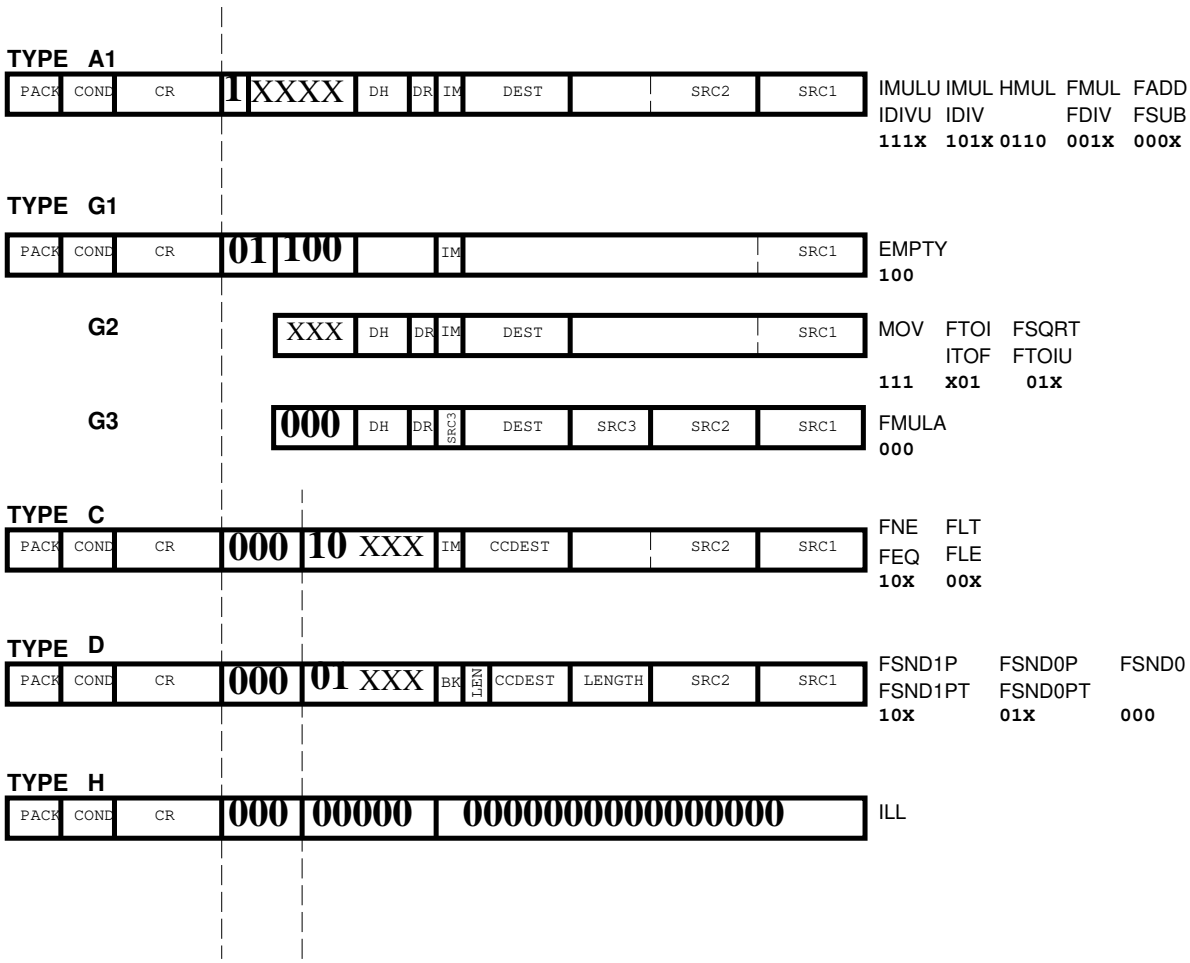


Figure 21: *Floating Point Unit Operations: 26*