Instruction Set

The CGRA consists of multiple functional units (FUs) which each have their own instruction set. These FUs are the Load Store Unit (LSU), the Register File (RF) and the Arithmetic Logic Unit (ALU), although other units might be added in the future. For all FUs that are implemented and functionally tested the instruction set is listed below.

Load Store Unit:

Type 1 instructions:

These instructions have the following format (bit widths for destination and inputs might change in the future as we decide to add or remove in- or outputs from the FUs):

Opcode [11:7], Data**T**ype [6:5], **D**estination [4:4], Input**B** [3:2], Input**A** [1:0]

The 5 bit opcode describes the operation that is to be performed, The DataType field is used for loading bytes, halfwords and words. The remaining 6 bits are divided into 2 sets of 3 bit, the first set specifying ‘load’ operations and the second set specifying ‘store’ operations. These 3-bit sets are formatted as follows:

Bit 2: Load or Store / NOP
Bit 1: Global / Local memory
Bit 0: Implicit / Addressed

The DataType field can be one of the following values:

00: BYTE
01: HWORD
10: WORD
11: DWORD

The destination specifies which output register has to be written.

The input describes which of the inputs of the functional unit will be selected for a specific operand. For all operations in this instruction type: input B is the address input and input A is the data input.

Note: when a simultaneous load and store are performed on the same read- and write address the NEW value will be returned by the load.

Operation	**Description**	**Opcode and parameters**
NOP	No OPeration	0000000_?_??_??
PASS outD, inA	PASS A to output	0010011_D_??_AA
SLA TYPE, inB, inA	Store Local Address	00001_TT_?_BB_AA
SLI TYPE, inA	Store Local Implicit	00010_TT_?_??_AA
SGA TYPE, inB, inA	Store Global Address	00011_TT_?_BB_AA
SGI TYPE, inA	Store Global Implicit	10110_TT_?_??_AA
LLA TYPE, outD, inB	Load Local Address	00101_TT_D_BB_??
LLI TYPE, outD	Load Local Implicit	00110_TT_D_??_??
LGA TYPE, outD, inB	Load Global Address	00111_TT_D_BB_??
LGI TYPE, outD	Load Global Implicit	01000_TT_D_??_??
LLI_SLA TYPE, outD, inB, inA	Load Local Implicit | Store Local Address	01001_TT_D_BB_AA
LGI_SLA TYPE, outD, inB, inA	Load Global Implicit | Store Local Address	01010_TT_D_BB_AA
LLA_SLI TYPE, outD, inB, inA	Load Local Address | Store Local Implicit	01011_TT_D_BB_AA
LLI_SLI TYPE, outD, inA	Load Local Implicit | Store Local Implicit	01100_TT_D_??_AA
LGA_SLI TYPE, outD, inB, inA	Load Global Address | Store Local Implicit	01101_TT_D_BB_AA
LGI_SLI TYPE, outD, inA	Load Global Implicit | Store Local Implicit	01110_TT_D_??_AA
LLI_SGA TYPE, outD, inB, inA	Load Local Implicit | Store Global Address	01111_TT_D_BB_AA
LGI_SGA TYPE, outD, inB, inA	Load Global Implicit | Store Global Address	10001_TT_D_BB_AA
LLA_SGI TYPE, outD, inB, inA	Load Local Address | Store Global Implicit	10010_TT_D_BB_AA
LLI_SGI TYPE, outD, inA	Load Local Implicit | Store Global Implicit	10011_TT_D_??_AA
LGA_SGI TYPE, outD, inB, inA	Load Global Address | Store Global Implicit	10100_TT_D_BB_AA

LGI_SGI TYPE, outD, inA	Load Global Implicit | Store Global Implicit	10101_TT_D_??_AA

Type 2 instructions:

This type of instruction takes a 4-bit register address and input number as parameters. The opcode is 6 bit in size, hence the instruction format is as follows:

Opcode [11:6], Register **Y** [5:2], Input**A** [1:0]

Input A is the data input.

The SRM operation is used to write the LSU's configuration registers, which can be found in the LSU description.
The LRM operation can be used to read configuration registers. To ensure compatibility with the LRM operation used for the RF, this operation always writes to the highest output port number.

Operation	**Description**	**Opcode and parameters**
SRM rY, inA	Store data from input A in register YYYY	100000_YYYY_AA
LRM rY	Load data from register YYYY into the output buffer	101000_YYYY_??

Register File

Type 1 instructions:

These instructions have the following format (bit widths for destination and inputs might change in the future as we decide to add or remove in- or outputs from the FUs):

Opcode [11:5], **D**estination [4:4], Input**B** [3:2], Input**A** [1:0]

The 7 bit opcode describes the operation that is to be performed. For the RF, the only instruction of this type is NOP.

Operation	**Description**	**Opcode and parameters**
NOP	No OPeration	0000000_?_??_??

Type 3 instructions:

This type of instructions has an opcode of only 2-bit in size. The parameters are: register X, register Y, and input A. The format of this instruction is:

Opcode [11:10], Register **X** [9:6], Register **Y** [5:2], Input**A** [1:0]

The RF only has one instruction of this type and it performs a parallel register read and write on two different addresses. The input (A) specifies the input for the data that is to be used in the register write part of the operation.

Operation	**Description**	**Opcode and parameters**
LRM_SRM rX, rY, inA	Load data from register XXXX | Store data from input A in register YYYY	11_XXXX_YYYY_AA

Type 2 instructions:

This type of instruction takes a 4-bit register address and input number as parameters. The opcode is 6 bit in size, hence the instruction format is as follows:

Opcode [11:6], Register **Y** [5:2], Input**A** [1:0]

Input A is the data input and is only used in register write operations.

? = don't care

Operation	**Description**	**Opcode and parameters**
SRM rY, inA	Store data from input A in register YYYY	100000_YYYY_AA
LRM rY	Load data from register YYYY into the output buffer	101000_YYYY_??

Type 4 instructions:

This type of instruction takes a input specified address and data input number as parameters. The opcode is 8 bit in size, hence the instruction format is as follows:

Opcode [11:4], Input**B** [3:2] , Input**A** [1:0]

Input A is the data input and is only used in register write operations. Input B is the input on which the register address is present.

? = don't care

Operation	**Description**	**Opcode and parameters**
SRA inB, inA	Store data from input A in register specified on input B	10010000_BB_AA

LRA inB	Load data from register specified on input B into the output buffer	10001000_BB_??

Arithmetic Logic Unit

Type 1 instructions:

These instructions have the following format (bit widths for destination and inputs might change in the future as we decide to add or remove in- or outputs from the FUs):

Opcode [11:5], **D**estination [4:4], Input**B** [3:2], Input**A** [1:0]

The 7 bit opcode describes the operation that is to be performed.

The DataType field can be one of the following values:

110: BYTE
010: HWORD
101: WORD

The destination specifies which output register has to be written.

The input describes which of the inputs of the functional unit will be selected for a specific operand.

Operation	**Description**	**Opcode and parameters**
NOP	No OPeration	0000000_?_??_??
ADD outD, inB, inA	Add A and B	0011010_D_BB_AA
ADD_SE TYPE, outD, inB, inA	Add A and B and sign extend the inputs from specified width to datapath width	TTT1010_D_BB_AA
SUB outD, inB, inA	Subtract B from A	0011011_D_BB_AA
SUB_SE TYPE, outD, inB, inA	Subtract B from A and sign extend the inputs from specified width to datapath width	TTT1011_D_BB_AA
AND outD, inB, inA	Bitwise AND of A and B	0010000_D_BB_AA
NAND outD, inB, inA	Bitwise NAND of A and B	0110000_D_BB_AA
OR outD, inB, inA	Bitwise OR of A and B	0010001_D_BB_AA
NOR outD, inB, inA	Bitwise NOR of A and B	0110001_D_BB_AA
XOR outD, inB, inA	Bitwise XOR of A and B	0010010_D_BB_AA

XNOR outD, inB, inA	Bitwise XNOR of A and B	0110010_D_BB_AA
NEG outD, inA	Bitwise negate of A	0110011_D_??_AA
CMOV outD, inB, inA	Conditional move (based on flag from compare) on A (flag=F) or B (flag=T)	1110011_D_BB_AA
ECMOV outD, inB, inA	External Conditional move (flag is bit 0 from input 0) on A (flag=F) or B (flag=T)	0000011_D_BB_AA
PASS outD, inA	PASS A to output	0010011_D_??_AA
PASS_SE TYPE, outD, inA	PASS A to output and sign extend the input from specified width to datapath width	TTT0011_D_??_AA
EQ outD, inB, inA	Test if A and B are equal	1101111_D_BB_AA
NEQ outD, inB, inA	Test if A and B are not equal	1011111_D_BB_AA
LTU outD, inB, inA	Test if A less than B (unsigned)	0011111_D_BB_AA
LTS outD, inB, inA	Test if A less than B (signed)	1001111_D_BB_AA
GEU outD, inB, inA	Test if A greater or equal to B (unsigned)	0111111_D_BB_AA
GES outD, inB, inA	Test if A greater or equal to B (signed)	0101111_D_BB_AA
SHLL1 outD, inA	Logic shift left (1 bit) on A	0010100_D_??_AA
SHLL4 outD, inA	Logic shift left (4 bit) on A	0010101_D_??_AA
SHRL1 outD, inA	Logic shift right (1 bit) on A	0010110_D_??_AA
SHRL4 outD, inA	Logic shift right (4 bit) on A	0010111_D_??_AA
SHRA1 outD, inA	Arithmetic shift right (1 bit) on A	0000110_D_??_AA
SHRA4 outD, inA	Arithmetic shift right (4 bit) on A	0000111_D_??_AA

Immediate Unit:

Type 5 instructions:

These instructions have the following format:

Opcode [N-1:N-1], I**M**ediate [N-1:0]

The single-bit opcode specifies if the immediate value has to be written to the output of the immediate unit.

Since the Immediate Unit (IU) is a special version of a instruction decoder (ID) the width of these instructions can be different than the instruction width for the other IDs.

The size of the Immediate (M) field scales with the instruction size (e.g a 9-bit instruction will have a 8-bit data field).

Operation	Description	Opcode and parameters
NOPI	No OPeration	0_{N-1{?}}
IMM value	write IMMediate to data network	1_{N-1{M}}

Accumulate & Branch Unit

Type 1 instructions:

These instructions have the following format (bit widths for destination and inputs might change in the future as we decide to add or remove in- or outputs from the FUs):

Opcode [11:5], **D**estination [4:4], Input**B** [3:2], Input**A** [1:0]

The 7 bit opcode describes the operation that is to be performed. For the RF, the only instruction of this type is NOP.

Operation	Description	Opcode and parameters
NOP	No OPeration	0000000_?_??_??
JR inB	Jump Relative, input B (signed) is added to the program counter	1100000_?_BB_??
JA inB	Jump Absolute, input B (unsigned) overwrites the program counter	1101000_?_BB_??
BCR inB, inA	Branch Conditional Relative, input B (signed) is added to the program counter when condition A != 0	1110000_?_BB_AA
BCA inB, inA	Branch Conditional Absolute, input B (unsigned) overwrites the program counter when condition A != 0	1111000_?_BB_AA

Type 2 instructions:

This type of instruction takes a 4-bit register address and input number as parameters. The opcode is 6 bit in size, hence the instruction format is as follows:

Opcode [11:6], Register **Y** [5:2], Input**A** [1:0]

Input A is the data input and is only used in register write operations.

? = don’t care

Operation	**Description**	**Opcode and parameters**
SRM rY, inA	Store data from input A in register YYYY	100000_YYYY_AA
LRM rY	Load data from register YYYY into the output buffer	101000_YYYY_??
ACCU rY, inA	Unsigned accumulate on register YYYY, with input A	110010_YYYY_AA
ACCS rY, inA	Signed accumulate on register YYYY, with input A	110011_YYYY_AA

Type 6 instructions:

This type of instruction takes a 6-bit immediate value and input number as parameters. The opcode is 4 bit in size, hence the instruction format is as follows:

Opcode [11:8], I**M**mediate [7:2], Input**A** [1:0]

Input A is the input specifying the branch condition and is only used in conditional branch operations.

? = don’t care

Operation	**Description**	**Opcode and parameters**
JRI value	Jump Relative Immediate, the immediate M (signed) is added to the program counter	0001_MMMMMM_??
JAI value	Jump Absolute Immediate, the immediate M (unsigned) overwrites the program counter	0011_MMMMMM_??
BCRI value, inA	Branch Conditional Relative Immediate, the immediate M (signed) is added to the program counter when condition A != 0	0101_MMMMMM_AA
BCAI value, inA	Branch Conditional Absolute Immediate, the immediate M (unsigned) overwrites the	0111_MMMMMM_AA

	program counter when condition A != 0	

Multiplier Unit

Type 1 instructions:

These instructions have the following format (bit widths for destination and inputs might change in the future as we decide to add or remove in- or outputs from the FUs):

Opcode [11:5], **D**estination [4:4], Input**B** [3:2], Input**A** [1:0]

The 7 bit opcode describes the operation that is to be performed.

The destination specifies which output register has to be written.

The input describes which of the inputs of the functional unit will be selected for a specific operand.

Operation	**Description**	**Opcode and parameters**
NOP	No OPeration	0000000_?_??_??
MULLU outD, inB, inA	Unsigned multiplication of A and B, output is the lower part	100_1000_D_BB_AA
MULLU_SH8 outD, inB, inA	Unsigned multiplication of A and B, output is the lower part, result shifted right by 8 bit.	100_1001_D_BB_AA
MULLU_SH16 outD, inB, inA	Unsigned multiplication of A and B, output is the lower part, result shifted right by 16 bit.	100_1010_D_BB_AA
MULLU_SH24 outD, inB, inA	Unsigned multiplication of A and B, output is the lower part, result shifted right by 24 bit.	100_1011_D_BB_AA
MULLS outD, inB, inA	Signed multiplication of A and B, output is the lower part	101_1000_D_BB_AA
MULLS_SH8 outD, inB, inA	Signed multiplication of A and B, output is the lower part, result shifted right by 8 bit.	101_1001_D_BB_AA
MULLS_SH16 outD, inB, inA	Signed multiplication of A and B, output is the lower part, result shifted	101_1010_D_BB_AA

	right by 16 bit.	
MULLS_SH24 outD, inB, inA	Signed multiplication of A and B, output is the lower part, result shifted right by 24 bit.	101_1011_D_BB_AA
MULU outD, inB, inA	Unsigned multiplication of A and B, complete multiplier output on port D and D+1	110_1000_D_BB_AA
MULU_SH8 outD, inB, inA	Unsigned multiplication of A and B, complete multiplier output on port D and D+1, result shifted right by 8 bit.	110_1001_D_BB_AA
MULU_SH16 outD, inB, inA	Unsigned multiplication of A and B, complete multiplier output on port D and D+1, result shifted right by 16 bit.	110_1010_D_BB_AA
MULU_SH24 outD, inB, inA	Unsigned multiplication of A and B, complete multiplier output on port D and D+1, result shifted right by 24 bit.	110_1011_D_BB_AA
MULS outD, inB, inA	Signed multiplication of A and B, complete multiplier output on port D and D+1	111_1000_D_BB_AA
MULS_SH8 outD, inB, inA	Signed multiplication of A and B, complete multiplier output on port D and D+1, result shifted right by 8 bit.	111_1001_D_BB_AA
MULS_SH16 outD, inB, inA	Signed multiplication of A and B, complete multiplier output on port D and D+1, result shifted right by 16 bit.	111_1010_D_BB_AA
MULS_SH24 outD, inB, inA	Signed multiplication of A and B, complete multiplier output on port D and D+1, result shifted right by 24 bit.	111_1011_D_BB_AA
LH outD	Load contents of the high part of the multiplier output regsiter to output D (does not do a multiplication)	010_0000_D_??_??