CGRA_HEEP

CGRA version used for compiler research experiments with USI and for HEEPocrates tapeout. This version is originally based on the cgra_v2.0.1 version. It currently has a 4x4 reconfigurable array organized in 4 columns (with shared program counter). The RCs are connected to their closest neighbors and the columns can work in synchronization. 2x2 and 3x3 CGRAs are also compatible.

CGRA CONFIGURATION WORD FORMAT		
# COLUMNS (one-hot encoding)	KERNEL START ADDRESS	# INSTR.
15:12	11:5	4:0
TOTAL: 16 bits		

Examples:

0001 0010000 01100
1 column kernel ---- kernel start address: 16 ---- kernel length: 13 instructions

0011 0101100 01111
2 columns kernel ---- kernel start address: 44 ---- kernel length: 16 instructions

CGRA INSTRUCTION FORMAT						
MUXA_SEL	MUXB_SEL	ALU_OP	RF_SEL	RF_WE	MUXF_SEL	IMM
31:28	27:24	23:19	18:17	16	15:13	12:00
TOTAL: 32 bits						

MUXA_SEL / MUXB_SEL:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	SELF	RCL	RCR	RCT	RCB	R0	R1	R2	R3	IMM	-	-	-	-	-

MUXF_SEL:

0	1	2	3	4	5	6	7
PREV	RCL	RCR	RCT	RCB	-	-	-

SELF: current RC result through own output register (i.e., result from previous clock cycle)

RC[L/R/T/B]: **R**econfigurable **C**ell [**L**eft/**R**ight/**T**op/**B**ottom] result (from previous clock cycle)

IMM: sign extended to datapath width (32 bits) immediate value. It holds the branch address for conditional branch instructions.

ZERO: hardcoded zero mux input. It can be generated from the immediate field but both might be needed at the same time, in particular for conditional branch operations: the immediate field holds the branch address and one value is compared to zero most of the time. For example BEQ IMM,R0, 0 => branch to IMM address if R0 is equal to zero.

RF_SEL: selects one of the 4 internal registers of the RC as the destination register (register to which the result is written to). In any case, the result is written to the output register that is connected to the neighboring RCs. This output register is updated only if the RC is active (i.e., not executing a nop).

RF_WE: enables the write to the register selected by RF_SEL.

ALU OPERATION		
OPCODE	**NAME**	**NOTE**
TYPE 0: opcode		
0	NOP	no operation (output register is not updated in this case and retains its previous value)
25	EXIT	use to notify the CGRA controller the kernel is finished (bypassed by branch request)
TYPE 1: opcode, rd, rs1, rs2 Example: SADD R0, IMM, RCT => R0=IMM+RCT		
1	SADD	signed addition
2	SSUB	signed subtraction
3	SMUL	signed multiplication (32bx32b => 32b)
4	FXPMUL	fixed point multiplication (1 sign bit + 16 integer bits + 15 decimal bits, 32bx32b => 32b)
5	SLT	shift left logical
6	SRT	shift right logical
7	SRA	shift right arithmetic
8	LAND	bitwise logical AND
9	LOR	bitwise logical OR
10	LXOR	bitwise logical XOR
11	LNAND	bitwise logical NAND
12	LNOR	bitwise logical NOR
13	LXNOR	bitwise logical XNOR
TYPE 2: opcode rd, rs1, rs2, fs Example: BSFA R3, PREV, RCB, RCT => R3 = RCT flag == 0 ? PREV, RCB)		
14	BSFA	operand a out if sign flag (of prev. clock cycle) equals 1 otherwise operand b out
15	BZFA	operand a out if zero flag (of prev. clock cycle) equals 1 otherwise operand b out
TYPE 3: opcode rs1, rs2, ad Example: BGE RCT, ZERO, 7 => branch to address 7 (IMM=7) if RCT==0		
16	BEQ	branch if equal
17	BNE	branch if not equal
18	BLT	branch if less than
19	BGE	branch if greater of equal
TYPE 4: opcode, rs1, rs2		
20	JUMP	jump to address rs1+rs2
TYPE 5: opcode rd Example: SWD R0 => store value from R0 to SRAM		

ALU OPERATION		
OPCODE	**NAME**	**NOTE**
27	LWD	load word direct from SRAM (using read pointer specified in peripheral register)
28	SWD	store word direct from SRAM (using write pointer specified in peripheral register)
TYPE 6: opcode rd, rs Example: LWI R3, SELF => load the 32 bits word at address SELF in SRAM to R3		
29	LWI	load word indirect
30	SWI	store word indirect

ISA/HARDWARE IMPLEMENTATION DETAILS:

LWD/SWD: each column is given an input and an output pointer to the SRAM at the beginning of a kernel through theCGRA peripheral registers. Calling LWD or SWD uses these pointers, inputpointer and output pointer respectively, to access the SRAM. Every call, the corresponding pointer is incremented by four automatically to access to the next 32 bits value at the next call.

LWI/SWI: similar to LWD/SWD but the sram address is generated by the RCs and used to access the SRAM. One RC at the time per column can make a load/store request to the SRAM but all theRCs of the column can access the data (in case of a load) using the SRAM input.

All the RCs of a column can issue a LWD/SWD or/and LWI/SWI at the same time. The requested will be pipelined to the platform bus and the execution stall until all the RCs' requests are served.Each RC can only access it's own value read from the bus and by default this value is stored in the output register and can optionally be saved in one internal register file using the corresponding bitfields of the configuration word.

Destination register:

In case no internal register (i.e., R0-R3) is used for rd (in format type 1, 2, 5, 6), in order to keep the same format, ROUT is specified as rd (which always happen anyway). For example:

SADD R0, IMM, RCT => in this case the result is written back to R0 (and also to ROUT by default). But if no internal register is used ROUT is explicitly add to keep the same format:

SADD ROUT, IMM, RCT