MUL DRAC specification version v0.1

Rubén Langarita Benítez

February 12, 2020

Contents

1 General purpose of the module

Person in Charge; Rubén Langarita

The Multiplication Unit module is placed in the execution stage and it is in charge of executing the multiplication operations. This module receives two data operands of 64 bits and some control signals. It outputs the product of both operands. The module can operate with the lowest 32 bits of the operands, in which case, it will return a 64-bit result in one cycle. When operating with operands of 64 bits, it will return the result in two cycles. When operating with 64 bits, the result is of size 128 bits, and some control bits select which part should be returned and if the operands should be interpreted as signed or unsigned.

2 Design placement

The Multiplication Unit module is placed inside the execution stage module. There is only one instance per core. All signals of the module are connected inside the execution stage module.

3 Parameters

All parameters, enums and types used in this module are defined in drac_pkg or risc_pkg.

4 Interface

Signal name	Width	Type	Description
clk_i	1	in	Clock
rstn_i	1	in	Reset
kill_mul_i	1	in	Kill the current operation
request_i	1	in	Valid request, start computation
func3_i	3	in	Control signal 000 → signed operands, low part 001 → signed operands, high part 010 → signed × unsigned, high part 011 → unsigned operands, high part
int_32_i	1	in	32-bit operation
src1_i	64	in	Operand 1
src2_i	64	in	Operand 2
result_o	64	out	Product
stall_o	1	out	Unit stalled
done_tick_o	1	out	Result ready

5 Reset behaviour

The reset signal *rstn_i* works on a negative edge. The outputs will be 0 and if some operation is being executed, it will be stopped.

6 What could not happen

- If *request_i* is 1 (means that the module should start computing) and *int_32_i* is 0, *func3_i* should take a value between 000 and 011, higher values are invalid.
- *stall_i* and *done_tick_i* can not be 1 at the same time.

7 Behaviour

7.1 Description

In this section we describe the behaviour of the Multiplication Unit module for each possible input. The meaning of the different symbols used in the next table are the following:

- x: Input without relevance or output undefined.
- -: Same input as previous line. The operation needs a second cycle to be completed.
- A^{sig}: Interprets the variable A as signed.
- A^{uns}: Interprets the variable A as unsigned.
- $A_{[31:0]}$: Selects the bits 31 to 0 from the variable A.
- $sign\_ext(A)$: Sign extend A. The most significant bit of A will be appended to the most significant side of the number until the 64 bits are full.
  $sign\_ext(A[31:0]) = A[31], A[31], A[31]...A[31], A[30:0]$.

kill_mul_i	**request_i**	**func3_i**	**int_32_i**	**src1_i**	**src2_i**	**result_o**	**stall_o**	**done_tick_o**
1	x	xxx	x	x	x	0	0	0
0	1	xxx	1	*A*	*B*	$sign\_ext((A^{sig}_{[31:0]} \times B^{sig}_{[31:0]})_{[31:0]})$	0	1
0	1	000	0	*A*	*B*	x	1	0
-	-	-	-	-	-	$(A^{sig} \times B^{sig})_{[63:0]}$	0	1
0	1	001	0	*A*	*B*	x	1	0
-	-	-	-	-	-	$(A^{sig} \times B^{sig})_{[127:64]}$	0	1
0	1	010	0	*A*	*B*	x	1	0
-	-	-	-	-	-	$(A^{sig} \times B^{uns})_{[127:64]}$	0	1
0	1	011	0	*A*	*B*	x	1	0
-	-	-	-	-	-	$(A^{uns} \times B^{uns})_{[127:64]}$	0	1

8 Special cases, corner cases

- The multiplication can overflow
- Any combination of inputs that are not defined in the previous section will produce zeros in the outputs.