Adder

Adder implements two functions:

- add
- subtract

The generic map list is presented in the following table.

name	type	value	description
g_data_width	natural	64	Base precision of the system.
g_addr_width	natural	9	The width of the address for the limbs (in this case 2^9 = 512 limbs, each g_data_width=64 wide, this gives 32768 bits = 1 BRAM in 7 series Xilinx products).
g_ctrl_width	natural	8	Control communication line width.
g_select_width	natural	5	Width of the signal to select the physical address (e.g. 18 physical registers requires 5 bits to multiplex the signal).
g_id	natural	15	Id of the module (has to be coded in no more than 6 bits).

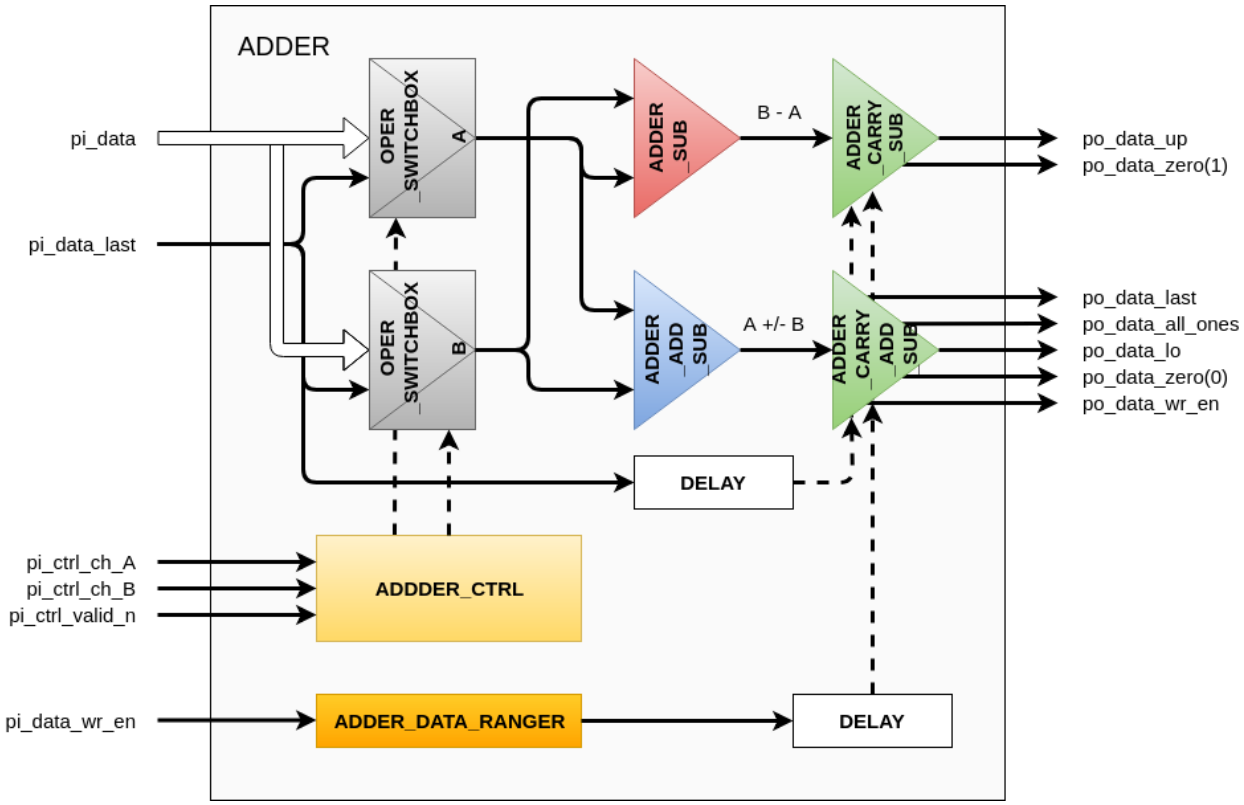
The port list is presented in the following table.

group	name	size	direction	description
general	pi_clk	1	in	Clock
	pi_rst	1	in	Reset
control	pi_ctrl_ch_A	g_ctrl_data	in	Control communication line - selects operation (+/-), source operand A
	pi_ctrl_ch_B	g_ctrl_data	in	Control communication line - selects source operand B
	pi_ctrl_valid_n	1	in	Valid signal (active low) for control data to indicate the header of the control communication stream; one signal for both control lines (A and B)
data	pi_data	g_data_width * num_of_phys_registers	in	All registers feed data to this bus. The adder contains two multiplexers to select (based on the control lines) data A and data B.
	pi_data_wr_en	1	in	Valid data (active high) for both operands simulatneously.
	pi_data_last	1	in	Last limb of the larger operand.
	po_data_up	g_data_width	out	Result data up.
	po_data_lo	g_data_width	out	Result data down.
	po_data_wr_en	1	out	Valid data (active high) for result.
	po_data_last	1	out	Last limb of the result.
	po_data_all_ones	1	out	Flag indicating all ones for po_data_lo.
	po_data_zero	2	out	Flag indicating all zeros for po_data_x: ▪ po_data_zero(1) for po_data_up ▪ po_data_zero(0) for po_data_lo

Dual result output

The output of the adder provides two results. The reason is that in some cases of addition/subtraction the result is not in the data format standard used in this project (see Data exchange format). For example subtraction of two positive numbers (3 and 5) can give (5 - 3 = 2) positive or (2-5 = -3) negative value. There is no quick and easy way to determine which operand is greater, so the sign of the result is known only at the end of the operation. In order to speed up this operation, it is run in parallel (A - B and B - A), both results are stored and finally one of the two results is taken as valid (the one that fits the used standard of data representation) and the sign is adjusted accordingly (details in CPU - sign update - [OUTDATED / WRONG]).

The block diagram of the adder is presented in the figure below.



The '-' and '+/-' blocks generate a series of limbs with one bit carrys. The carry block needs to add limbs with corresponding carry bits and, apart from the result, generate appropriate flags.

IMPORTANT: If the core operation opcode is ADD, then it can, but, it doesn't have to, translate into ADD operation in ADDER block. If one operand is negative and the other positive, then ADD operation will be executed by the ADDER as SUB. The change of operation is done by the CPU. All of the cases are presented in Sign and physical register update.

Each block is described in their own sections:

- ADDER_CTRL
- ADDER_DATA_RANGER
- ADDER_ADD_SUB
- ADDER_SUB
- ADDER_CARRY_ADD_SUB
- ADDER_CARY_SUB

Example of waveform of addition (0xBC 10 AD 56 and 0xEF BE AD DE) on the top module level is presented in the following figure.