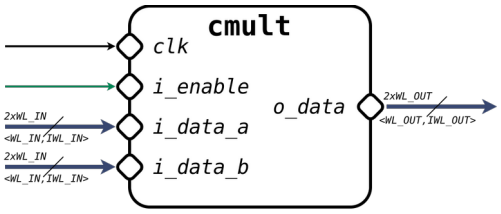
Interface



Overview

The **CMULT** block is a complex multiplier based on fixed-point arithmetic with support to rounding based on SystemC standard. Since complex multiplications are the backbones of most digital signals processing (DSP) IPs, the main purpose of this block is to be a maximum throughput adaptable block, with support to low-power techniques through "enable" logic.

Behavioral description

The block's I/O signals and parameters are detailed in tables alongside. The complex multiplicands are received at each clock cycle in i_data_a and i_data_b ports when i_en flag is asserted. Therefore, the block only operates when i_en input is in logic HIGH. When block is enabled, the o_data shall output the result of complex multiplication between multiplicands a and b after the configured latency:

$$z = (a_i+ja_q)(b_i+jb_q) = a_ib_i-a_qb_q + j(a_ib_q+a_qb_i)$$

Besides the main functionality, the block has a built-in fixed-point round blocks (`fxp_rnd`). These blocks adjust output precision according to the rounding methods defined in the IEEE 1666 SystemC standard. Rounding is applied when the output's fixed-point format **<WL_OUT, IWL_OUT>** has a shorter word length than the full-precision result. If the output word length matches full precision, no rounding is needed, and no `fxp_rnd` instances are allocated. When the output word length exceeds full precision, the extra bits are zero-padded. An example of this method is shown below for operands a *<8,1> = (24+j31) = (0.19+j0.24)* and b *<8,1> = (72 – 109) = 0.56-j0.85*:

- *z <17,3> = (5107-j384) (full precision)*
- *z <8,1> = (40 – j3) (rounded to <8,1>)*
- *z <20,5> = 10214 - j768 (zero-padded)*
- *z <7,0> = 39 – j3 (truncated to <7,0>)*

OBS: It is encouraged to the reader to check the twos-complement binary representation and verify the correctness of rounding methods.

The block's latency can be determined by the number and the presence of input and output registers set by `INREGS` and `OUTREGS` parameters, along with the configuration of the pipeline algorithm indicated by the `ALGORITHM` parameter.

The use of block-enable logic facilitates the logic synthesis tool to place clock-gating cells enabling a power-efficient approach. In scenarios where this functionality is unnecessary, the i_en input may be "hard-wired" to `1'b1`(VDD) in block instantiation. This causes the block is always enabled, disencouraging the allocation of clock gating cells in the circuit and optimizing the complex multiplier circuit by synthesis tool.

Parameters

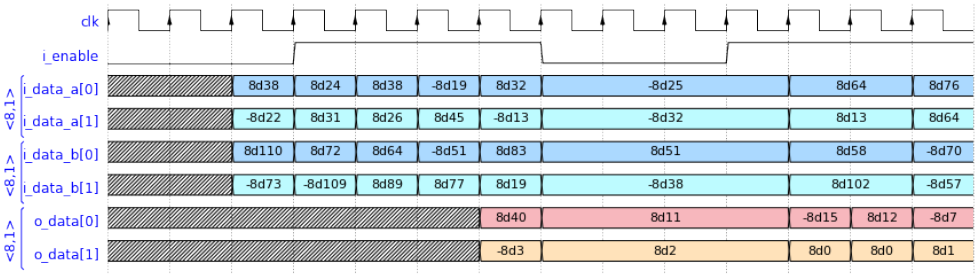
Parameter	Type	Default	Description
`ALGORITHM`	`string`	`'standard'`	Datapath multiplication method.
`INREGS`	`bit`	`1`	Enable input registers.
`IWL_IN`	`int`	`1`	Number of integer bits per input word.
`IWL_OUT`	`int`	`3`	Number of integer bits per output word.
`WL_IN`	`unsig. int`	`8`	Number of bits per input word.
`WL_OUT`	`unsig. int`	`17`	Number of bits per output word.
`OUTREGS`	`unsig. int`	`1`	Number of output registers. If this param is set 0, output data is not registered.

Interface signals

Port name	I/O	Parallelism	Wordlength	Integer bits	Description
`clk`	`I`	`1`	`1`	`1`	Clock signal for block operation.
`i_enable`	`I`	`1`	`1`	`1`	Input enable for block operation.
`i_data_a`	`I`	`1`	`WL_IN`	`IWL_IN`	Input complex data representing the multiplicand a.
`i_data_b`	`I`	`1`	`WL_IN`	`IWL_IN`	Input complex data representing the multiplicand b.
`o_data`	`O`	`1`	`WL_OUT`	`IWL_OUT`	Output complex product between multiplicands a and b.

Timing diagram

The timing diagram below exemplifies the behavior of `CMULT` block as its inputs vary. It is important to highlight that the first data (`i_data_a = 38-j22 and i_data_b = 110-j73`) is not processed, as `i_enable` is deasserted. In this scenario, a standard complex multiplier algorithm is used, with input registers and a single stage of output registers. The output is rounded to **<8,1>** precision, matching the input data format.



Microarchitecture

The microarchitecture of **CMULT** block with standard datapath (`ALGORITHM='standard'`) is presented in the block diagram of figure below. The dotted lines indicates that the presence of the structure is affected by the configuration parameter. Since this block has a "enable" logic for clock gating control, all flip-flops receives the gated clock from **CGIC** in their "clock" pin. If this functionality is not necessary, it is recommended to wire the `i_enable` port to logic HIGH. This way the synthesis tool will reduce the clock gate cell and remove the i_enable port.