439:AS
sign_b = op_b_i[31] & signed_mode_i[1];
438:AS
sign_a = op_a_i[31] & signed_mode_i[0];
399:AS
res_adder_l = alu_adder_ext_i[32:0];
444:AS
div_change_sign = sign_a ^ sign_b;
441:AS
op_b_ext = { sign_b, op_b_i };
445:AS
rem_change_sign = sign_a;
440:AS
op_a_ext = { sign_a, op_a_i };
400:AS
res_adder_h = alu_adder_ext_i[33:1];
464:AL
464:BL
multdiv_state_d = multdiv_state_q;
accum_window_d = accum_window_q;
op_b_shift_d = op_b_shift_q;
op_a_shift_d = op_a_shift_q;
op_numerator_d = op_numerator_q;
md_state_d = md_state_q;
471:IF
(mult_en_i || div_en_i)
472:CS
md_state_q
508:CA
555:CA
473:CA
528:CA
498:CA
547:CA
554:CA
503:CA
508:BL
multdiv_state_d = multdiv_state_m1;
md_state_d = (multdiv_state_q == 5'd1)? MD_LAST : MD_COMP;
473:BL
multdiv_state_d = 5'd31;
498:BL
op_a_shift_d = 1'sb0;
op_numerator_d = (sign_a)? alu_adder_i : op_a_i;
md_state_d = MD_ABS_B;
547:BL
md_state_d = MD_FINISH;
554:BS
md_state_d = MD_IDLE;
503:BL
accum_window_d = { 32'h0, op_numerator_q[31] };
op_b_shift_d = (sign_b)? alu_adder_i : op_b_i;
md_state_d = MD_COMP;
510:CS
474:CS
529:CS
549:CS
operator_i
511:CA
521:CA
516:CA
555:BS
md_state_d = 1'bX;
487:CA
481:CA
491:CA
475:CA
530:CA
538:CA
534:CA
542:CA
550:CA
551:CA
511:BL
accum_window_d = res_adder_l;
op_a_shift_d = op_a_shift_q << 1;
op_b_shift_d = op_b_shift_q >> 1;
521:BL
accum_window_d = { next_reminder[31:0], op_numerator_q[multdiv_state_m1] };
op_a_shift_d = next_quotient;
516:BL
accum_window_d = res_adder_h;
op_a_shift_d = op_a_shift_q;
op_b_shift_d = op_b_shift_q >> 1;
487:BL
accum_window_d = { 33{ 1'b1 } };
md_state_d = (equal_to_zero)? MD_FINISH : MD_ABS_A;
481:BL
op_a_shift_d = op_a_ext;
accum_window_d = { 1'b1, ~(op_a_ext[32] & op_b_i[0]), op_a_ext[31:1] & { 31{ op_b_i[0] } } };
op_b_shift_d = op_b_ext >> 1;
md_state_d = MD_COMP;
491:BL
accum_window_d = op_a_ext;
md_state_d = (equal_to_zero)? MD_FINISH : MD_ABS_A;
475:BL
op_a_shift_d = op_a_ext << 1;
accum_window_d = { ~(op_a_ext[32] & op_b_i[0]), op_a_ext[31:0] & { 32{ op_b_i[0] } } };
op_b_shift_d = op_b_ext >> 1;
md_state_d = MD_COMP;
530:BL
accum_window_d = res_adder_l;
md_state_d = MD_IDLE;
538:BL
accum_window_d = next_quotient;
md_state_d = MD_CHANGE_SIGN;
534:BL
accum_window_d = res_adder_l;
md_state_d = MD_IDLE;
542:BL
accum_window_d = { 1'b0, next_reminder[31:0] };
md_state_d = MD_CHANGE_SIGN;
550:BS
accum_window_d = (div_change_sign)? alu_adder_i : accum_window_q;
551:BS
accum_window_d = (rem_change_sign)? alu_adder_i : accum_window_q;
Leaf_464:AL
446:AL
446:BL
447:IF
(!rst_ni)
!((!rst_ni))
447:BL
multdiv_state_q <= 5'h0;
accum_window_q <= 33'h0;
op_b_shift_q <= 33'h0;
op_a_shift_q <= 33'h0;
op_numerator_q <= 32'h0;
md_state_q <= MD_IDLE;
455:BL
multdiv_state_q <= multdiv_state_d;
accum_window_q <= accum_window_d;
op_b_shift_q <= op_b_shift_d;
op_a_shift_q <= op_a_shift_d;
op_numerator_q <= op_numerator_d;
md_state_q <= md_state_d;
Leaf_446:AL
443:AS
multdiv_state_m1 = multdiv_state_q - 5'h1;
558:AS
valid_o = (md_state_q == MD_FINISH) | (md_state_q == MD_LAST) & ((operator_i == MD_OP_MULL) | (operator_i == MD_OP_MULH));
435:AS
b_0 = { 32{ op_b_shift_q[0] } };
432:AS
one_shift = { 32'b0, 1'b1 } << multdiv_state_q;
442:AS
op_remainder = accum_window_q[32:0];
431:AS
is_greater_equal = ((accum_window_q[31] ^ op_b_shift_q[31]) == 1'b0)? res_adder_h[31] == 1'b0 : accum_window_q[31];
437:AS
op_a_bw_last_pp = { op_a_shift_q[32] & op_b_shift_q[0], ~(op_a_shift_q[31:0] & b_0) };
436:AS
op_a_bw_pp = { ~(op_a_shift_q[32] & op_b_shift_q[0]), op_a_shift_q[31:0] & b_0 };
434:AS
next_quotient = (is_greater_equal)? op_a_shift_q | one_shift : op_a_shift_q;
433:AS
next_reminder = (is_greater_equal)? res_adder_h : op_remainder;
401:AL
401:BL
alu_operand_a_o = accum_window_q;
multdiv_result_o = (div_en_i)? accum_window_q[31:0] : res_adder_l;
404:CS
operator_i
407:CA
405:CA
406:CA
407:CS
md_state_q
405:BS
alu_operand_b_o = op_a_bw_pp;
416:CA
420:CA
424:CA
412:CA
408:CA
406:BS
alu_operand_b_o = (md_state_q == MD_LAST)? op_a_bw_last_pp : op_a_bw_pp;
416:BL
alu_operand_a_o = { 32'h0, 1'b1 };
alu_operand_b_o = { ~op_b_i, 1'b1 };
420:BL
alu_operand_a_o = { 32'h0, 1'b1 };
alu_operand_b_o = { ~accum_window_q[31:0], 1'b1 };
424:BL
alu_operand_a_o = { accum_window_q[31:0], 1'b1 };
alu_operand_b_o = { ~op_b_shift_q[31:0], 1'b1 };
412:BL
alu_operand_a_o = { 32'h0, 1'b1 };
alu_operand_b_o = { ~op_a_i, 1'b1 };
408:BL
alu_operand_a_o = { 32'h0, 1'b1 };
alu_operand_b_o = { ~op_b_i, 1'b1 };
Leaf_401:AL