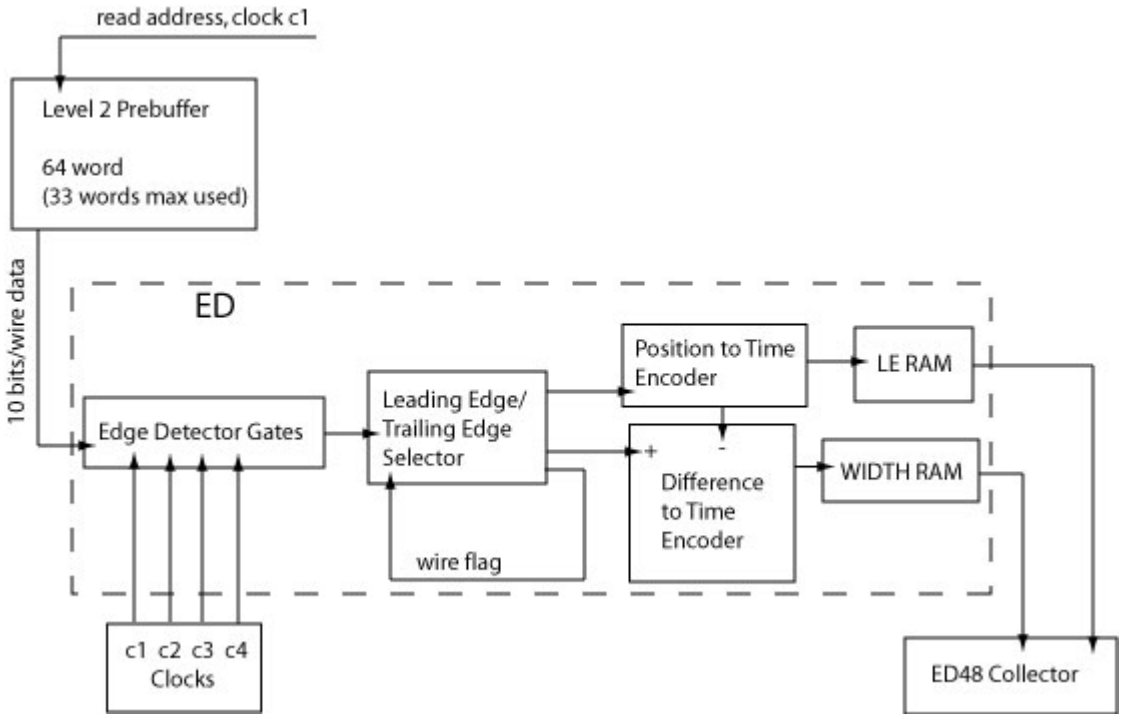
How ED and ED48 Work

The edge detector (ED) looks at the data coming out of the level 2 prebuffer and locates any transitions in the data. On a level 2 accept, data from one of the four level 1 buffers is moved to the level 2 prebuffer and the eds start examining the data. Each ed looks at the data coming from a single wire. Ed48 then combines the data from the individual eds and moves it into a single large output ram, packed for ease of the block transfer.



ED – Overview

A single ed looks for hits by working with two 10-bit words at a time. The beginning of a hit is defined as a zero followed by at least four ones or in the case of the first word, four ones in a row. The end of the hit is then defined as a one followed by four zeroes.

The diagram below shows how the data is defined in ed.

Bit Position:	[9]	[8]	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
Time Value:	0	1	2	3	4	5	6	7	8	9

Each time value represents a 1.2 ns interval, as each word covers 12 ns.

There are three possible transitions in each word and each transition needs a memory cycle. Below is a word that has three possible transitions.

Bit Position:	[9]	[8]	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
	0	***1***	1	1	***1***	0	0	0	0	***1***
Time Value:	0	1	2	3	4	5	6	7	8	9

In this case, there is a leading edge (transition #1) starting at time value 1 and ending at time value 4 (transition #2). There is also possibly another leading edge (transition #3) starting at time value 9, depending on what is in the next word. This is why ed looks at two words at a time. In the example above, it would check if the next word started with three ones.

Once a hit has been found, the data describing the hit (hit data) and the number of hits (word count) must be stored. Each ed has a little ram for storing the hit data. The hit is stored by giving the time value of the start of the hit (the leading edge) and the number of bits in the hit (the width). So, if the above example were the first word looked at, the leading edge stored would be one and the width would be four. If the example were the third word, the leading edge would be 21 (the 10 bits for the first word, 10 for the second and time value one in the third) and the width would still be four.

The maximum number of words that will be looked at for each wire is variable, but it will not exceed 33 words (which is 396/12). This is because it takes 12 ns for each word (1.2 ns per bit times 10 bits per word) and the maximum run time is 396 ns. The actual number that has to be searched is variable and is an input to ed.

The maximum number of hits stored is limited to four. Each ed has two little rams that hold the leading edges and the widths. These rams are called le_ram and width ram and each can hold four 8-bit words. Eight-bits limits the leading edges to 255 and less, but this is sufficient for the 306 ns drift time (255 x 1.2 ns). For reasons to be explained later, the largest leading edge will actually be 254. However, ed will continue to look through bits after 254 to find the width of any hits that did not finish by bit 254.

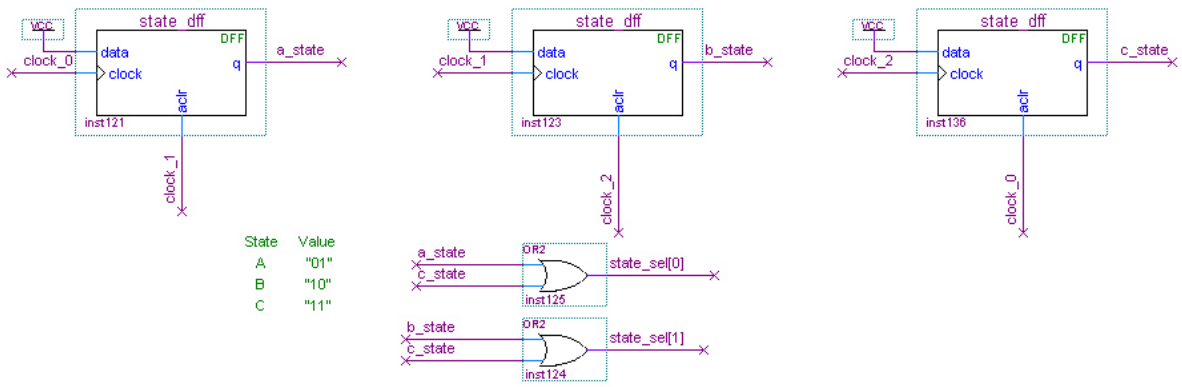
ED-The Details

As described in the overview, ed looks for transitions in three groups. The first group (A) looks for a specific pattern starting in bits 9-6, the second group (B) looks at bits 5-2 and the third group (C) looks at bits 1-0. The exact pattern that it is looking for depends on whether it successfully matched a pattern on any previous pass.

The following data will be used as an example - the hits are shown in bold. The small bold, underlined numbers are the 1.2 ns time values that would be stored (in le_ram) as the leading edges of the hits.

	[9]	[8]	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
first_word	0	0	**1**	**1**	**1**	**1**	0	0	0	0
	0	*1*	***2***	*3*	*4*	*5*	*6*	*7*	*8*	*9*
next_word	0	0	0	**1**	**1**	**1**	**1**	**0**	**1**	**0**
	10	*11*	*12*	***13***	*14*	*15*	*16*	*17*	*18*	*19*
	1	**1**	0	0	0	0	0	0	0	**1**
	20	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	***29***
	1	**1**	**1**	0	0	0	0	0	0	0
	30	*31*	*32*	*33*	*34*	*35*	*36*	*37*	*38*	*39*

In order to look at hits that might wrap around the edges of words, ed must look at two words at a time. It has two buffers called first_word[9..0] and next_word[9..0] that it uses to hold the two words. It also has a state machine that it uses to determine which group of bits it is searching. The states are called a_state, b_state and c_state.

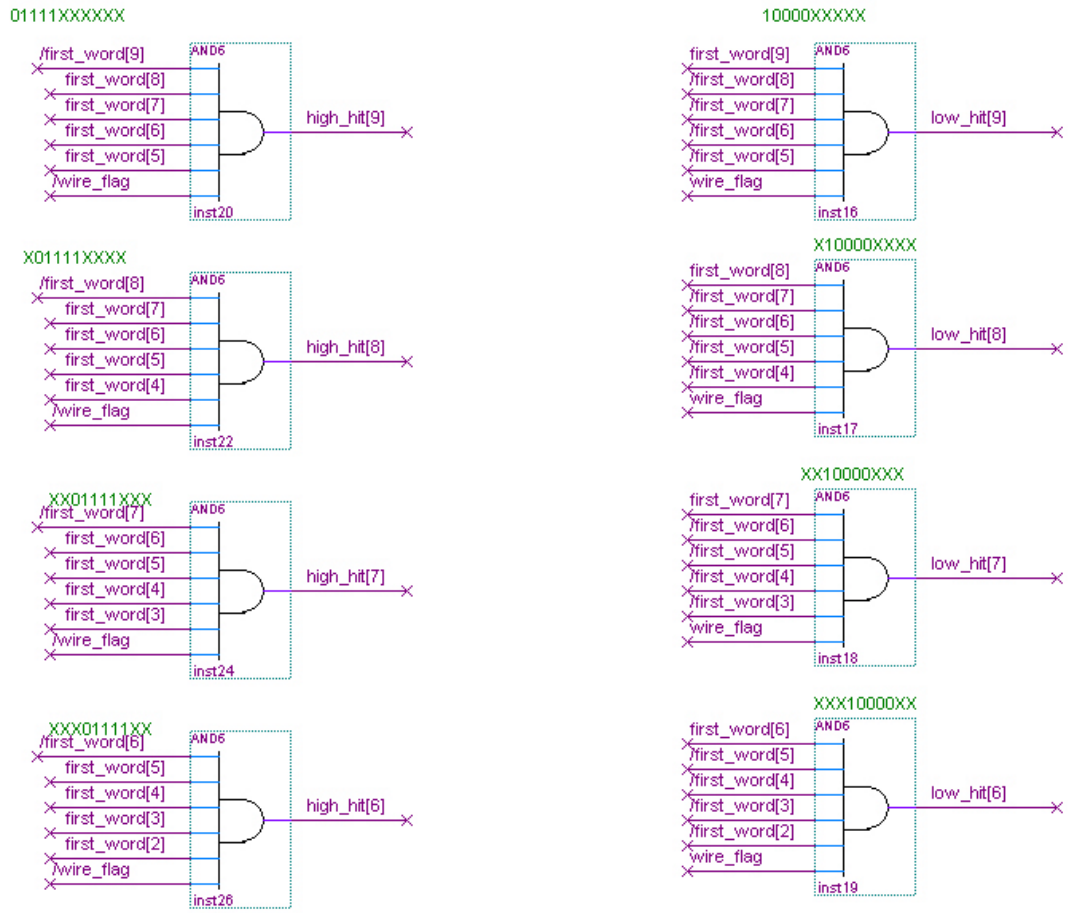


State machine for ed. Group A bits are examined during the a_state, group B during the b_state and group C during the c_state.

In the example data, ed first searches for the pattern starting in the group A bits, which are underlined below.

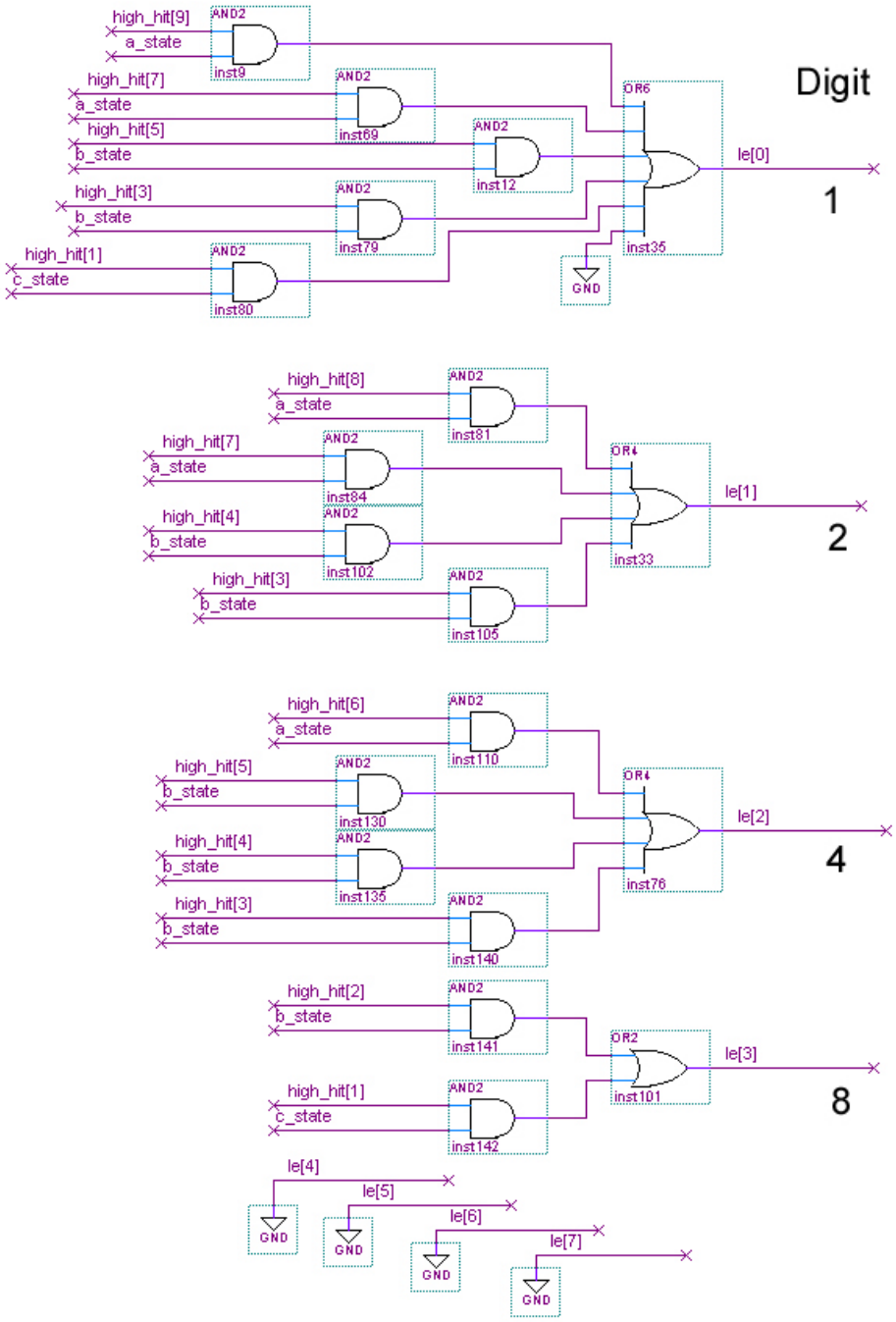
	[9]	[8]	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
first_word	0	0	**1**	**1**	**1**	**1**	0	0	0	0
next_word	0	0	0	**1**	**1**	**1**	**1**	**0**	**1**	**0**
	1	**1**	0	0	0	0	0	0	0	**1**
	1	**1**	**1**	0	0	0	0	0	0	0

The following shows how the pattern is searched for in group A.



Leading edge and trailing edge searching in group A. Note that the signal wire_flag is '0' when searching for a leading edge and '1'when searching for a trailing edge.

Since this is the first word, the pattern that is being searched for is the one that represents the leading edge. This pattern is 01111, and it is found starting at first_word[8]. The signal high_hit[8] is now set and used with the other high_hit[X] signals to decode the ones digit of the hit. In this instance, the value (which is 2) is stored in the vector le[X]. The following diagram shows how the position is decoded to a binary value, which is stored in le[X].

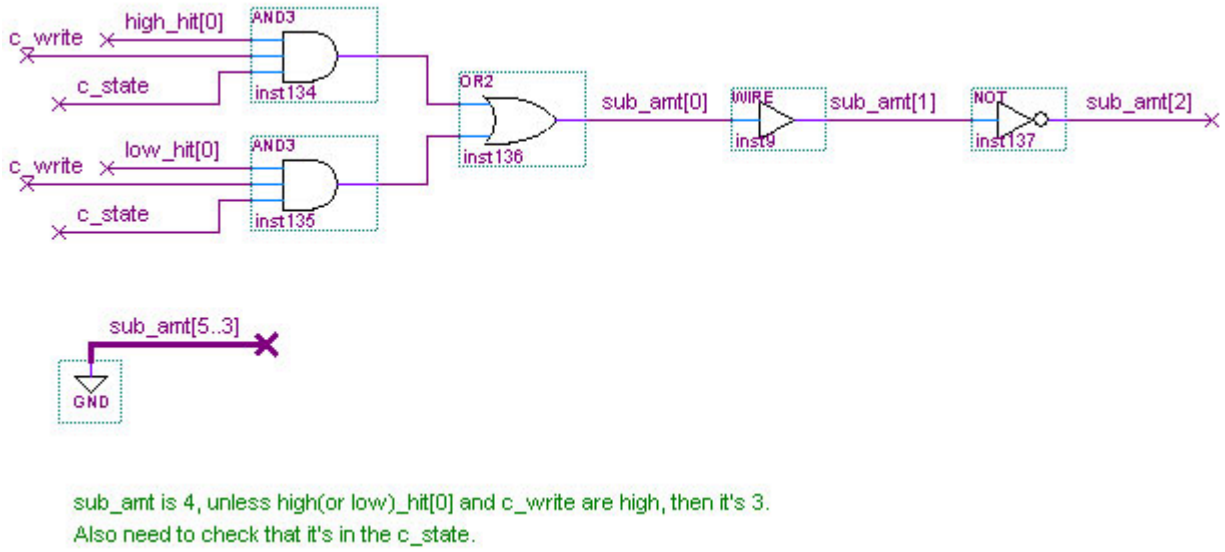


Decoding the hit data to the correct time value inside a single word.

Since this hit is in the first word of the example, the signal le[X] contains the correct value for the time value of the leading edge. However, if this hit were in any other word, this value would be incorrect. The correct time value of the hit is found by multiplying the address of the word by ten and adding the value in the signal le[X]. In the case above, the first word address is zero: 0 x 10 + 2 = 2, which is the value that will be stored in the leading edge ram.

It's slightly more complicated in the schematic because the read_address signal is what is being used for the word address. The data that comes from the level 2 prebuffer ram is registered, which means that read_address will be delayed. Plus, the data then first goes to the next_word buffer and then to the first_word buffer. So, four must first be subtracted from the read_address signal to get the correct value. And there is a special case where the hit is

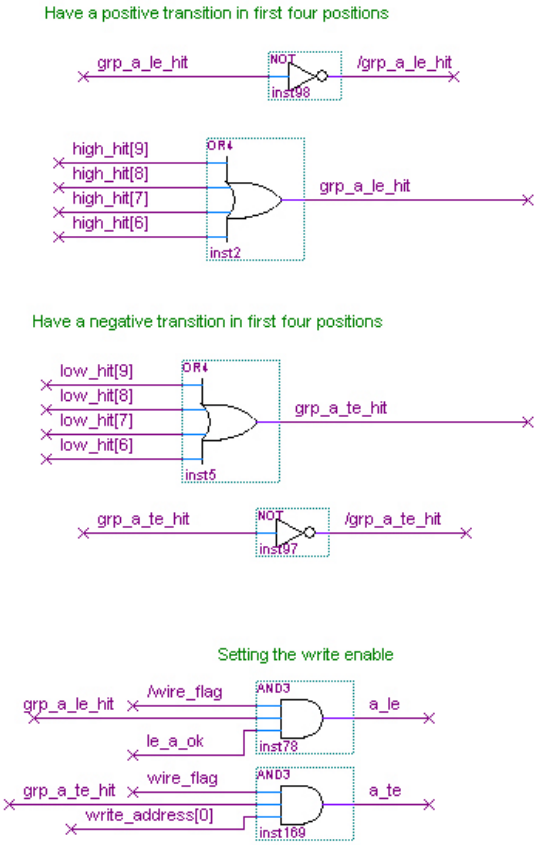
found in the first four bits of next_word. In this case, only three is subtracted. The following diagram shows how the value of the signal called sub_amt[5..0] is decoded.



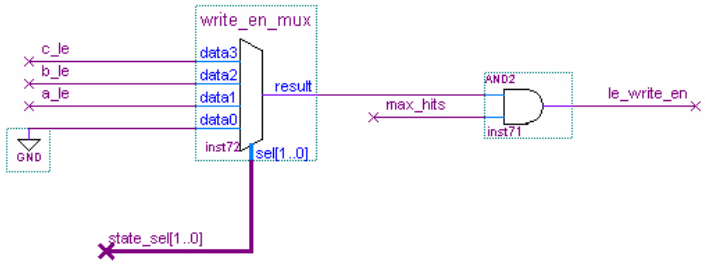
Decoding of value to be subtracted from the read_address. Note that the signal c_write means a hit was found in the group C bits.

Once the leading edge is calculated, it will be stored in the ram (le_ram) on the next clock. The write enable for the ram must be set and it is, using the state machine and some other signals in the group A section of the schematic. The leading edge is also stored in a signal called subtract_edge, so that it can be subtracted from the trailing edge value to find the width of the pulse. Finally, a flag is used to say that a leading edge has been found and now ed should be looking for the trailing edge pattern, 10000.

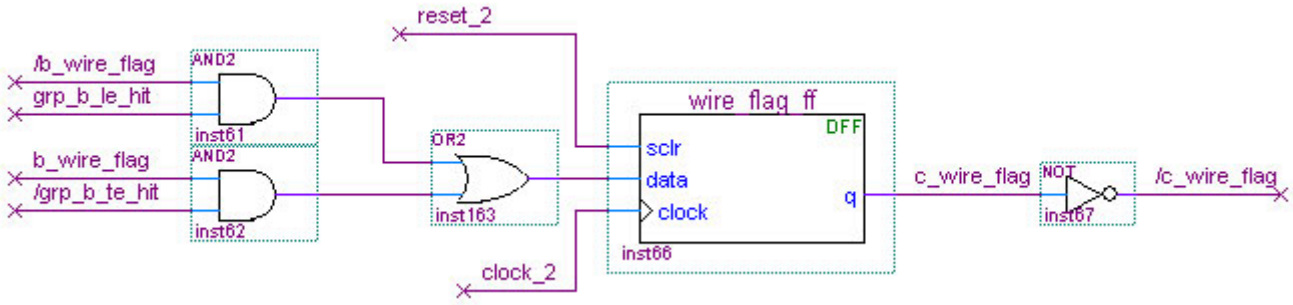
The following diagrams show the section of the schematic (for group A only) that control the write enables for the rams and set the wire flag for the next state (b_state). In the actual schematic, there is a section similar to this for each group.



Logic shows there is either a leading edge hit (grp_a_le_hit) or trailing edge hit (grp_a_te_hit) in group A. This then sets the signals a_le or a_te, which are used to control the write enables on the leading edge and width rams. Note that signal le_a_ok is used to make sure no leading edge is found after word 25, which is the last word in which a leading edge can be found. Also note that signal write_address[0] is used to make sure that a trailing edge cannot be found unless a leading edge was already written to ram.



*Multiplexer that controls the write enable on the ram holding the leading edges. Note that max_hits makes sure that, at most, four hits are written to the ram. The signals *_le mean that a leading edge was found in the group (A,B or C).*



Setting c_wire_flag

b_wire_flag	grp_b_le_hit	grp_b_te_hit	c_wire_flag
0	0	0	0
0	1	0	1
0	0	1	0
1	0	0	1
1	1	0	1
1	0	1	0

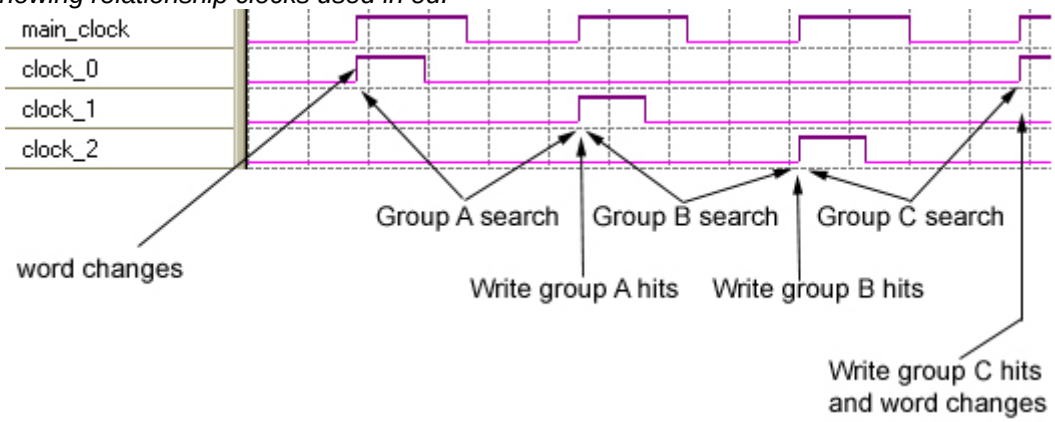
Function of x_wire_flag
0 = Look for trailing edge
1 = Look for leading edge

Logic and truth table for setting the wire flag for the following group.

There are four different clocks that are used in ed. Their characteristics are given in the table below and shown graphically in the diagram below.

Clock Name	Clock Period	Phase	Duty Cycle
main_clock (c4)	22 ns	0	50%
clock_0 (c1)	66 ns	17.6	10%
clock_1 (c2)	66 ns	39.6	10%
clock_2 (c3)	66 ns	61.6	10%

Waveform showing relationship clocks used in ed.



Waveform showing tasks performed during the different clock cycles.

Back in the example, ed is now in b_state and is looking for a trailing edge starting in the following data.

	[9]	[8]	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
first_word	0	0	1	1	1	1	0	0	0	0
next_word	0	0	0	1	1	1	1	0	1	0
	1	1	0	0	0	0	0	0	0	1
	1	1	1	0	0	0	0	0	0	0

The pattern 10000 is found starting at first_word[4]. Using the same techniques that were used in group A to find the leading edge, the trailing edge is found. The only extra step is that the leading edge must be subtracted from the trailing edge to determine the width of the hit, which is what is stored.

The diagram below shows how the data to write for either edge is determined.

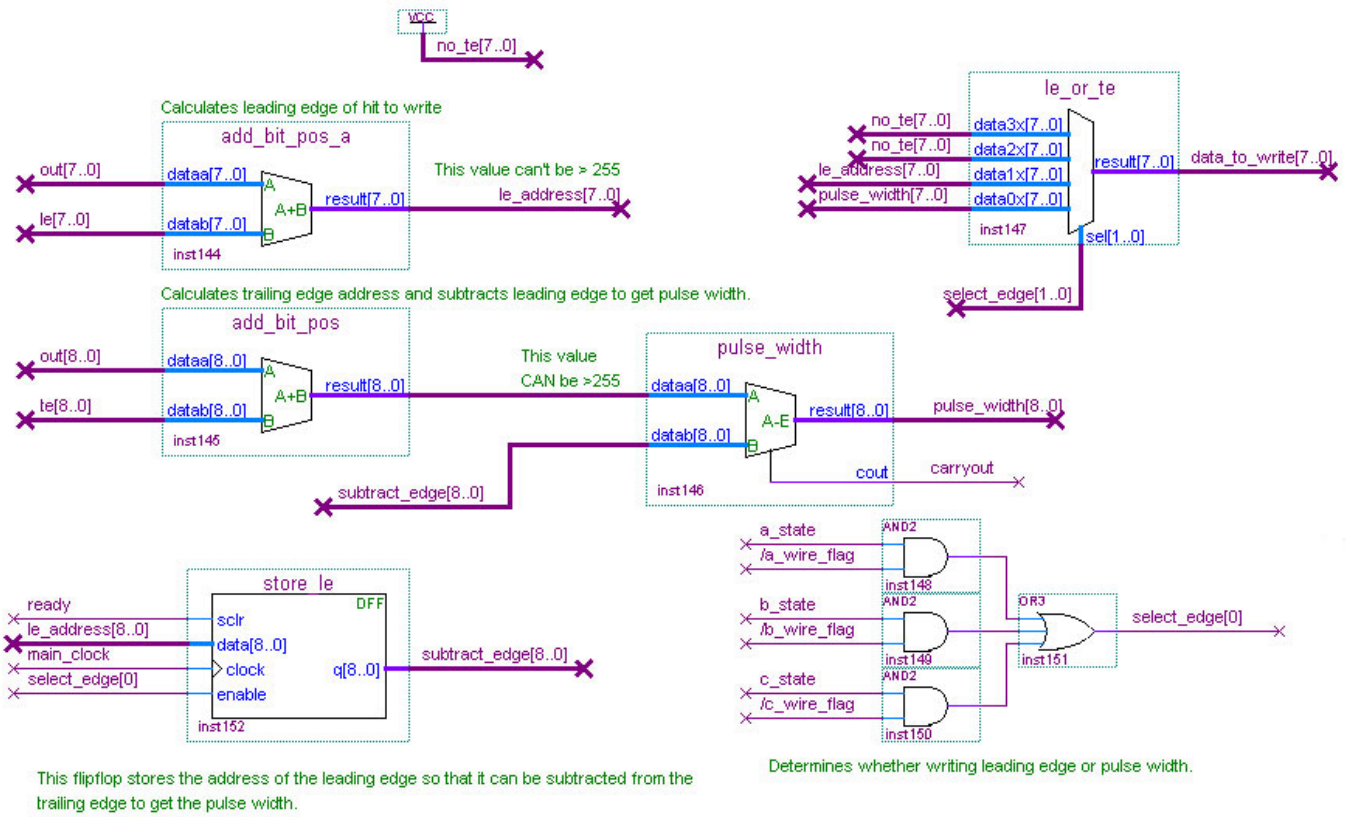


Diagram showing now data to write to ram is determined. In the multiplexer le_or_te, the value at data0x is the width of the pulse, the value at data1x is the leading edge of the pulse and the values at data2x and data3x are for the possibility that a leading edge is found, but no trailing edge is found. The store_le flip-flop is used to hold the leading edge time value while looking for a trailing edge.

On the next clock, the data is written to the ram, the state machine changes to c_state and ed looks for another leading edge in the following data.

	[9]	[8]	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
first_word	0	0	**1**	**1**	**1**	**1**	0	0	0	0
next_word	0	0	0	1	1	1	1	0	1	0
	1	1	0	0	0	0	0	0	0	1
	1	1	1	0	0	0	0	0	0	0

As the pattern is not found here, no data is written, the words change and the process repeats. The data now examining is the following.

	[9]	[8]	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]
	0	*0*	***1***	***1***	***1***	***1***	*0*	*0*	*0*	*0*
first_word	0	0	0	1	1	1	1	0	1	0
next_word	1	1	0	0	0	0	0	0	0	1
	1	1	1	0	0	0	0	0	0	0

This process repeats until the signal stop_reading goes high. At this point, if ed is looking for a trailing edge, it writes all ones as a width to signal that it didn't find a trailing edge.

A few notes on ed. The two memories, le_ram and width_ram, are separate so that when writing their contents to a bigger memory (in ed48) a 16-bit bus can be used, instead of an 8-bit bus. Along with the rams, there is an output bus called word_count[3..0] that gives a final total for how many hits were found. It is calculated by using the write address for the ram. At most, the number of hits is going to take three bits, there is an extra bit that can be used for anything. It is connected to the pin named wire_off, and can be used to tell whether a wire was on or off during the run. The largest value of a leading edge is 254 because the last possible time value of a hit in group A is three. Any leading edge with a value of 255 would start in group B. But if this were allowed, values 256, 257, etc. could also possibly occur. In order to prevent time values that cannot fit in 8-bits, it was decided that the largest leading edge time value would be 254.

ED48 – Overview

The job of ed48 is to take the data from each ram in each ed and pack it into two bigger rams. One ram, called hit_count_dualport, holds the number of hits that were found on each wire and the other ram, called hit_dat_dualport, holds the hit data.

ED48 – Details

Ed48 starts with a level 2 accept. Then, if it is not already reading data from the level 2 pre-buffer, it starts to do so. If it is already reading data from the level 2 pre-buffer, the level 2 accept is ignored. The signal then gets sent through various flip-flops, which change the signal from working on the cdf_clk to clock_0 and give a delay to allow data to move to the level 2 pre-buffer.

The signal called ready is sent to each ed. This signal is used to clear the counter that keeps track of how many words are written. The signal called set_go is then sent to a T-flip-flop that is used to tell ed48 that it should start reading data from the level 2 pre-buffer. Data is read until it reaches the address specified on the pin named words_to_read. Note that it takes four clocks for the data to get from the little rams in each ed, so this signal is delayed by four clocks.

Until the signal called stop_reading goes high, ed48 does nothing. This is the time while each of the eds is looking for hits in its data. When stop_reading does go high, it means that all of the eds are finished looking at their data and nothing else should be read from the level 2 prebuffer.

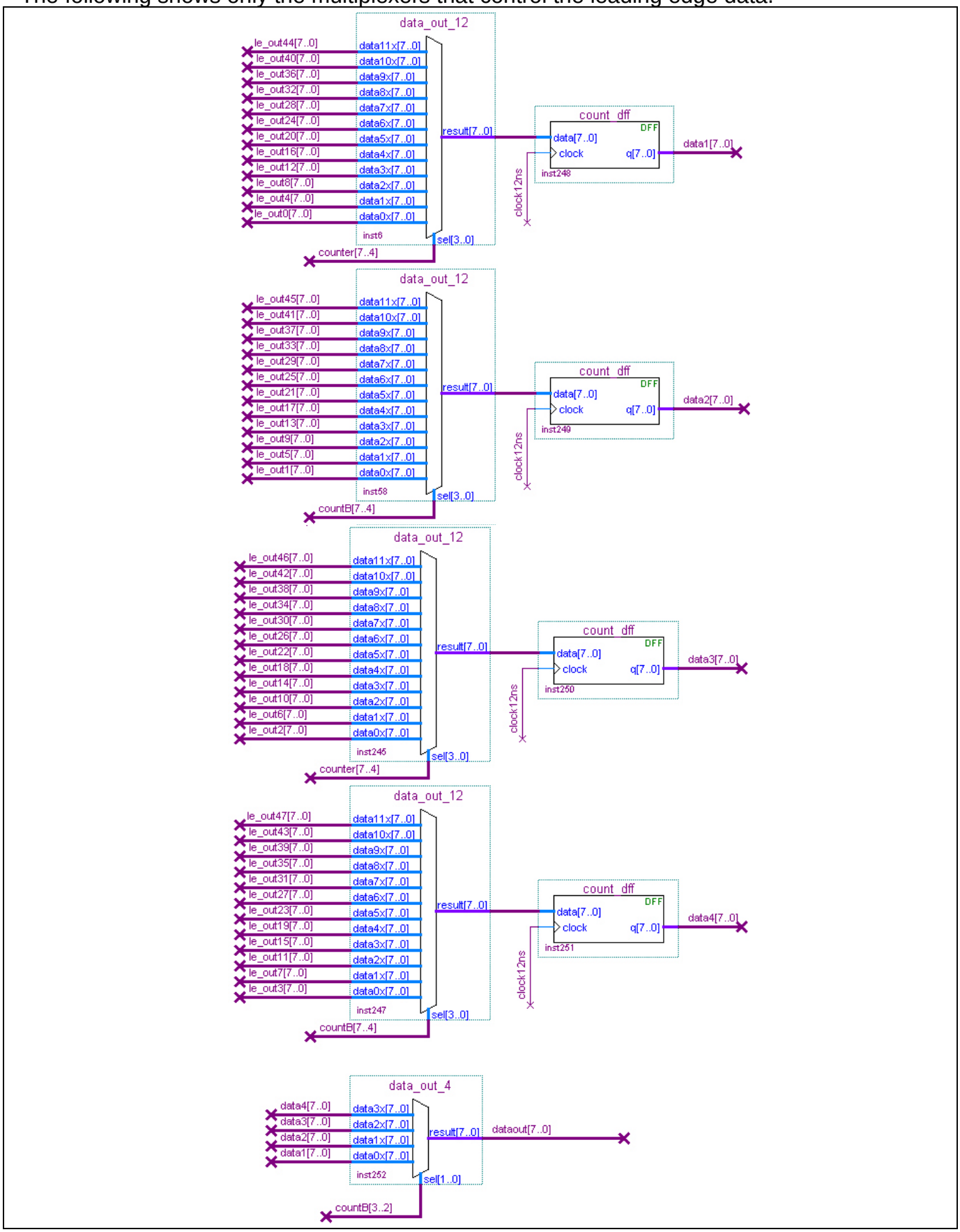
Next, ed48 totals up all the word counts from each ed. This sum is the number of 16-bit words will be written in the hit_dat_dualport ram. Since the data will be taken out of the ram on a 32-bit bus, the 16-bit word count is divided by two. This gives the number of 32-bit words that will

need to be read to get all the data. This value is output on the pin called number_of_hit_words[6..0].

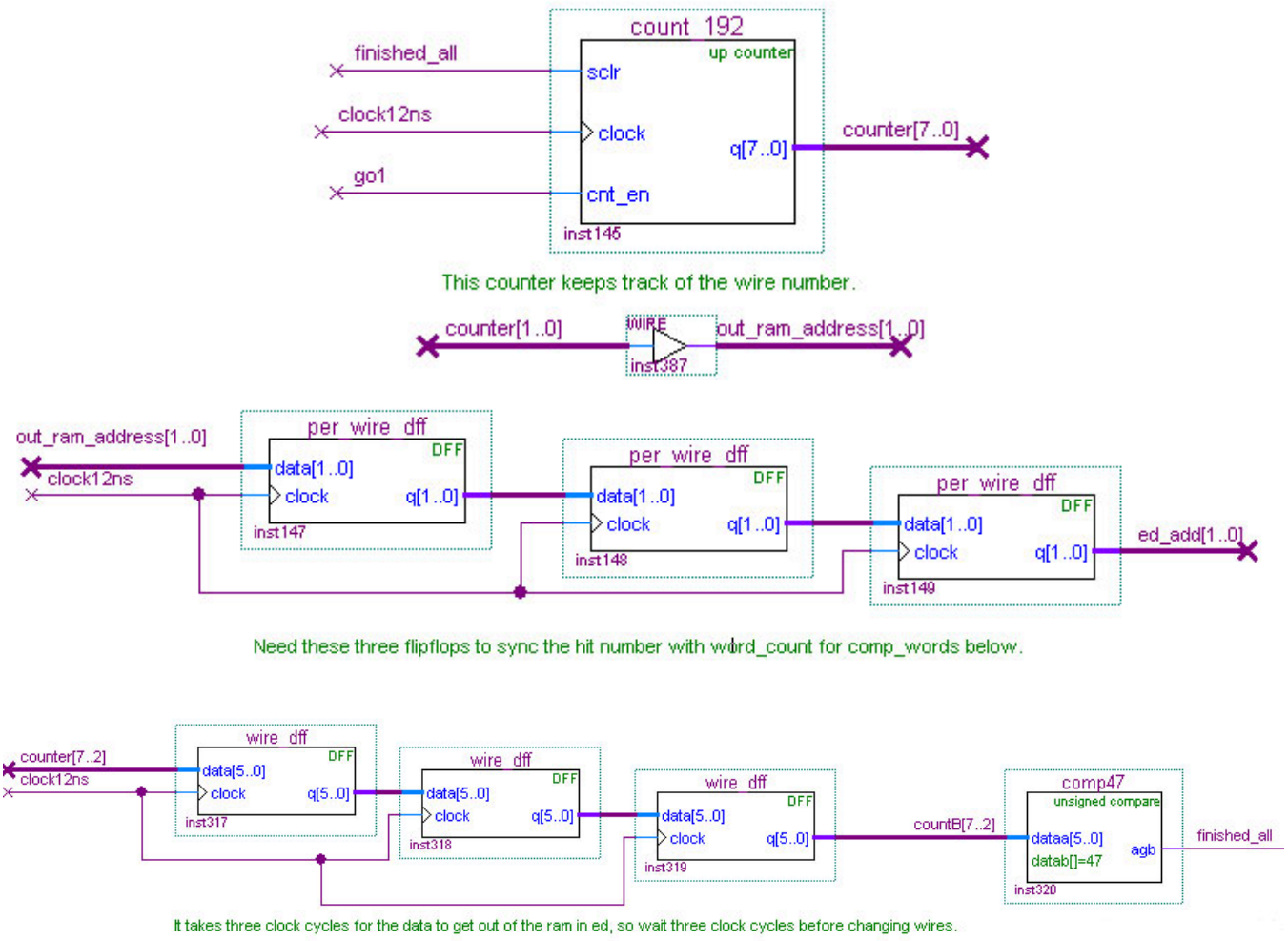
Also, when signal stop_reading goes high, an 8-bit counter is started. Its job is to count to 192, which is 4 (maximum number of hits) times 48 (total number of wires). The output signal of this counter is used to control an array of multiplexers which manage the hit data coming out of the eds and going into the single ram.

There are twelve 12-input and three 4-input multiplexers used to control the flow of data from the eds to the rams in ed48. One 4-input and four of the 12-input multiplexers are used each for the leading edge data, the widths and the word counts.

The following shows only the multiplexers that control the leading edge data.



The diagram below shows how the counter is used to control the flow of data through the multiplexers.



Counters and flip-flops used to control the selectors on the multiplexers.

The buffer counter[7..2] is used as selectors on the multiplexers, while the buffer [1..0] is sent to the rams in each of the eds as the address to read (out_ram_address[1..0]). The following table shows how the counter is used to control all the multiplexers.

000000*00* *Digits 1..0 are sent to the ed rams as the read address*
000000*01*
000000*10* Digits 3..2 are the selector for the 4-input multiplexer
000000*11*
000001*00* **Digits 7..4 are the selector for the 12-input multiplexer**
000001*01*
000001*10*
000001*11*

Since the all the data moving through the circuit is registered, the write enable the hit_dat_dualport ram is delayed by three clock cycles. Also, notice that the selectors on every other data_out_12 multiplexer are called countB[7..4], and the others are counter[7..4]. The

signal countB[7..0] is counter[7..0], delayed by three clock cycles for the same reason as the write enable.

Counter[1..0] (which is the address sent to the little rams in ed) is delayed through three flip-flops (called ed_add[1..0] after these flip-flops), added to one and then compared to the word_count[2..0] signal to control the write enable on the hit_dat_dualport ram. The diagram below shows how the write enable (wire_enable signal) for the hit_dat_dualport ram is controlled.

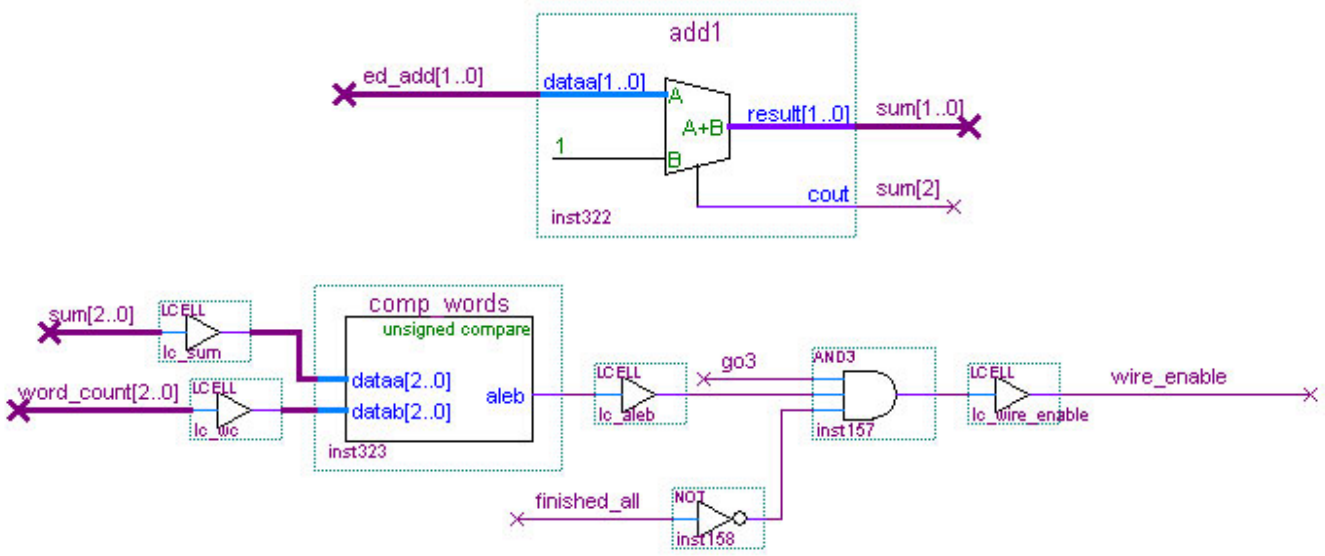


Diagram showing the logic used to control the signal wire_enable, which is used to control the write enable on hit_dat_dualport.

A second counter in ed48 is used for the write address on hit_dat_dualport. Its counter enable is controlled by the wire_enable signal and its synchronous clear is controlled by the finished_all signal.

When finished getting data from each ed, the signal finished_all is set. This signal is used to clear most counters and flip-flops in ed48. It is also used to set the signal tdc_done, which means that the data in the ram is ready to be read via vme.

In ed48, there is also a section with signals called save_wc, write_wc and clear_wc. These all have to do with the word counts that are written to the hit_count_dualport ram. On a stop_reading signal (when the data can be moved from the little ed rams to the hit_dat_dualport), the hit count for each wire is known and can be written to the hit_count_dualport. So one clock after stop_reading, the word counts are registered onto the bus to hit_count_dualport and on the next clock, they are written to the ram, along with the geographic address of the card, the board serial number, the board type and the B0 bunch. When the finished_all signal is received, this bus is cleared.

Updates:
June 2004
The data in hit_dat_dualport is stored as follows:

Width hit #2—Leading Edge hit #2—Width hit #1—Leading Edge hit #1
Width hit #4—Leading Edge hit #4—Width hit #3—Leading Edge hit #3
…

It was requested that the data instead be stored as follows:
Leading Edge hit #1—Width hit #1—Leading Edge hit #2—Width hit #2
Leading Edge hit #3—Width hit #3—Leading Edge hit #4—Width hit #4

Instead of changing how the data is stored, the assignment of the bus was changed as shown below: