Conceptual Design of Front-End Readout Electronics for the GLAST Silicon-Strip Tracker

R.P. Johnson, SCIPP
Draft: November 13, 1998

Introduction

This is a conceptual design for the GLAST silicon-strip front-end electronics. It is based on my note "Straw-Man Conceptual Design of Data Acquisition for the GLAST Tracker" from October 23, 1996. Some significant changes have been made in the front-end chips, however, in order to simplify the design. In particular, most (but not all) of the zero-suppression function has been moved from the readout chips to the controller chips. This may result in less data throughput, because the outputs of all of the channels in any chip having at least one hit must be clocked into the controller chips during a readout sequence.[1] Nonetheless, the principal design objectives remain the same:

1. Redundancy—if a single chip fails, it is still possible to read out all other chips in the system. This is achieved by placing a controller chip at each end of the hybrid, such that data from the string of readout chips can be shifted in either direction, and each controller can send commands to any of the chips. At least two failures would be needed in order to knock out a complete layer or a complete tower.
2. Minimize the number of signal lines on the hybrid and going up and down the tower. The design avoids output busses, so that a single chip with a bad driver cannot bring down the entire bus, and serial protocol is used as much as possible. The chips within a hybrid and the layers within a tower are daisy-chained, so that at the bottom of the tower all of the data can be obtained from a single serial line. Redundancy is achieved by duplicating the daisy chain.
3. Sparse readout—the data come out of the bottom of the tower already formatted as a set of addresses of hit strips.
4. Low power—clocking of the front-end chips is minimized. During data taking, the clock lines are active only during readout. Optionally, if a particular hybrid does not produce a fast-OR signal for a given event, then it is not read out at all. As long as nothing is broken and as long as the extra data throughput is not needed, half of the controller chips could be turned off by gating off their clocks.

Some assumptions have to be made before beginning the design.

1. Between 8 and 31 layers per tower side, so that up to 5 bits are necessary in order to specify a layer number (with one bit combination reserved as a wild-card, to which all layers respond in the case of a command).
2. 64-channel readout chips.
3. 25 readout chips per hybrid, in order to accommodate a plane of four-by-four 8 cm wafers or five-by-five 6.4 cm wafers with 195 μm strip pitch. Therefore, 5 bits are

[1] However, with the anticipated noise performance, only a few chips out of the 25 in a given readout section should have any data, so the readout speed should still be much greater than if all 64 channels of all 25 chips had to be read.

necessary in order to specify a chip address (with one bit combination reserved as a wild card).

4. 63 hits per hybrid is a reasonable maximum (any beyond that limit are discarded).

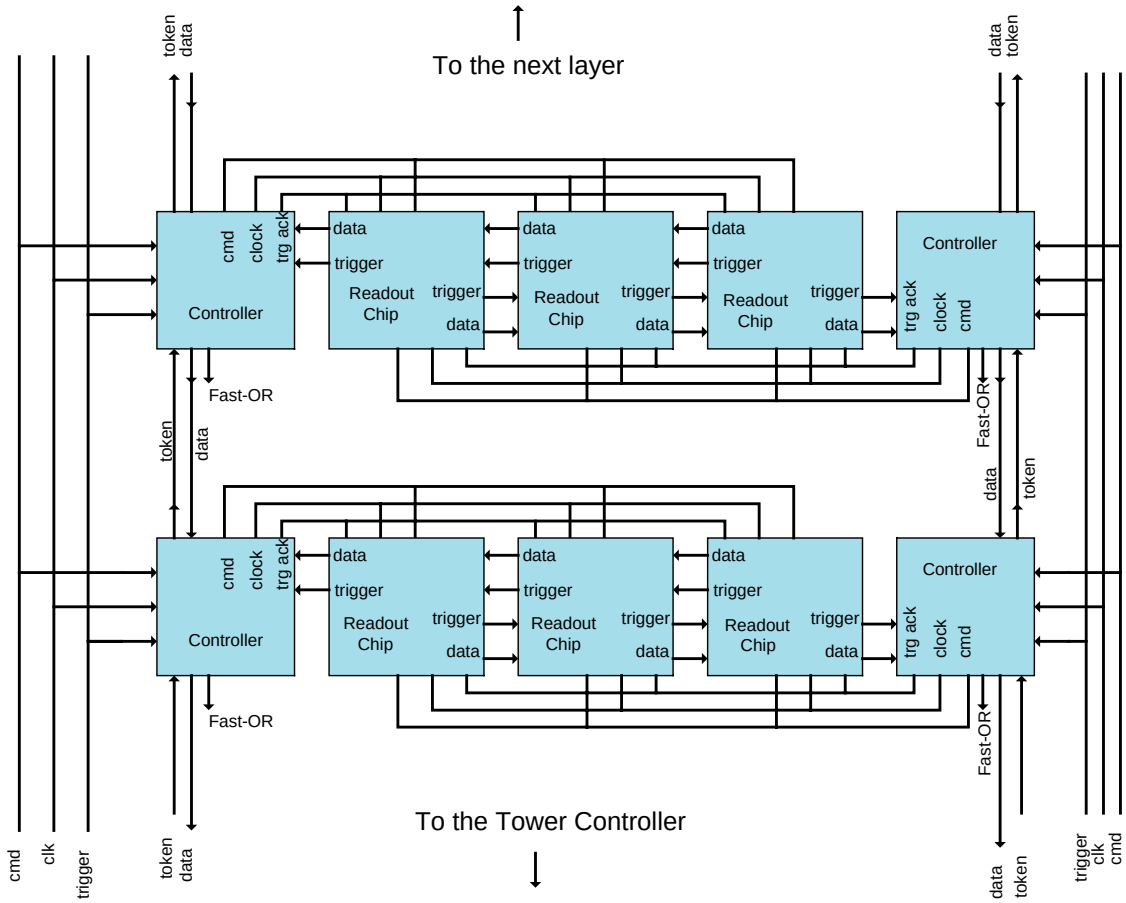

Figure 1. Simplified block diagram of two readout layers in the tracker, with only 3 of 25 readout chips shown.

Figure 1 shows a simplified block diagram of the chips on a single hybrid. The token and data lines connect the eight hybrids on every other plane of a given tower side (*x* or *y*) into a daisy-chain readout.

Data Format

Each controller of each hybrid sends a data packet consisting of a header and a sequence of hits. The beginning and end of a packet can always be recognized, so arbitrary gaps between packets are okay. Each packet consists of a start bit followed by a variable number of 11-bits words:

- Start bit—a single 1.
- Layer address (5 bits), followed by number of hits (6 bits).

- Error flag (1 bit), followed by time-over-threshold (10 bits).
  - ◊ Error flag bit: did an error occur during the readout? 1=error.
- Sequence of hits, each hit consisting of an 11-bit channel address (5 bits for the readout-chip number and 6 bits for the channel within the chip).
- 11-bit check-sum. This can be turned off when not needed, to improve the readout speed.

When the control register of a controller chip or a front-end readout chip is loaded, the previous register contents will appear on the data output line with a start bit, but without any header words. The registers would never be loaded during active data acquisition, however, so there is no danger of confusing the register contents with actual data.

Trigger Protocol

At initialization time, each front-end chip on each hybrid is told (by loading its control register) whether it should pass data and fast-ORs to the left-hand controller or the right-hand controller. Depending on the setting, when a hit occurs in any one or more of its unmasked channels, it sends a fast-OR signal to the chip on the right or the left, after making a logical OR of its own fast-OR with the one it received from the previous chip. Thus each controller receives on its fast-OR input a logical OR of the outputs of all unmasked comparators on the front-end chips for which it is responsible. That fast-OR must be immediately (asynchronously) sent down the tower to the tower controller.

The fast-OR input is also used to clear and start a time-over-threshold counter in the controller. The counter stops when the fast-OR input goes low again. At the same time, a trigger latency counter is started, which stops counting either when a trigger signal is received or the maximum trigger latency period expires. If the trigger latency counter times out before a trigger signal is received, then the time-over-threshold counter is stopped and cleared to be ready for the next fast-OR. If a second fast-OR is received from the front-end chips before the latency counter from the previous trigger has expired, then it is ignored and is not sent to the tower controller. (Note, however, that it is not possible to receive a second fast-OR until the previous signals have all passed below threshold, which usually takes much longer than the trigger latency time.)

The tower controller takes all of the fast-OR signals from the tower and generates the final trigger. When such a trigger occurs, a 1-bit trigger signal is sent to all controller chips in the entire GLAST tracker. The controllers on the hybrid process the trigger as follows:

- When the signal is received, a trigger signal is immediately fanned out to the front-end chips on the hybrid to latch their comparator outputs.
- The contents of the time-over-threshold counter are loaded into a FIFO buffer that has the same depth (8) as the FIFO buffers on the front-end chips. Of course, this can only occur after the time-over-threshold count stops or times out.
- A readout flag is loaded into the FIFO (in parallel with the time-over-threshold) that specifies whether the latency counter was active when the trigger signal was received.

The front-end chips respond to a trigger signal by latching the outputs of their comparators into their event FIFO buffers. The *tower* controller must keep track of how many trigger signals versus read-event commands have been sent, in order to be able to disable the trigger whenever the event and time-over-threshold buffers are full. Also, the

trigger logic must respond quickly enough such that the trigger signal reaches the front-end chips while the comparator outputs are still high (about 1.3 μs minimum time-over-threshold).

Readout Protocol

A readout sequence is initiated by sending a read command to the controller chips. Each of them responds by pulling the time-over-threshold and readout flag from the FIFO buffer. If the readout flag is set, then the controller sends a read-event command to the front-end chips. That command causes an event to move from the front-end-chip FIFO into the data output shift register, after which the clock is directed to the shift register to move the data toward the controller chip. Another command, end-read-event, must be sent to turn off the clock to the shift register. (During all of that time, the controller must supply a clock to the front-end chips, of course.) If the readout flag is false, then a clear-event command is sent to clear the event from the FIFO. How many clocks are needed to shift the data out depends on where the front-end chips are divided between the two controllers on the hybrid: a minimum of 10 clocks plus the number of data bits to be clocked in. As the data are clocked in, the controller keeps a chip counter and a hit counter going and latches the counts each time a hit is detected, for up to a maximum of 63 hits. Those hits are stored in one of two 63-deep buffers, alternating from one buffer to the next with each new event.

The format of the data that are shifted out of the front-end chips is

- Header bit: 1 if the chip has any hits, 0 if no hits.
- Data bits: none if the chip has no hits, 64 if the chip has at least one hit.

The controller chip first looks at the header bit. If it is zero, then it increments the chip counter and looks at the header bit from the following chip. On the other hand, if it is a one, then it starts up the hit counter and counts while clocking in the next 64 data bits, latching the hit counter each time a one if detected. Then it increments the chip counter and goes on to the next header bit, and so forth.

Movement of the data from the controller chips into the tower controller is initiated by the tower controller by sending a token to the first controller chip, which responds by sending its data to the tower controller as soon as it has finished reading from the front-end chips. When if has finished sending its data out, it passes the token to the controller chip on the layer above, giving it permission to send its data out on the serial line. Each controller has two event buffers, so that while it is sending data it can receive another read-event command and begin clocking the next event out of the front-end chips. Therefore, a second read-event command may be sent before the previous event has been read out of the controller chip. The controller chips will hold the pending read-event command until the data from the previous event have all been clocked out of the front-end readout chips. It will then begin clocking out the next event. The tower controller is responsible for keeping track of whether or not a buffer is free in the controller chip. What that means is that it should never send a *third* read-event command until all of the data from the first one have been received.

Also, there may be a delay between the time that the token is sent to a controller chip and the data start coming out, in the case that the chip has not finished clocking the bits from the front-end chips. The controller chip must hold the token until it is ready.

When a controller has finished sending its data packet, it passes the token on up to the next layer. When it is not in possession of the token, whatever it sees on its data input port (from the next layer) it passes on to its data output port on the following clock cycle. If a chip has no data for the given event, it is still required to send the first 11-bit word, consisting of the layer number and the number of hits, which is zero in such a case. This allows the tower controller to know when the readout is complete—it just has to wait for the last layer to report its data. The tower controller can strip out the empty data packets or otherwise reformat the data to reduce the data volume, if necessary.

It is the responsibility of the tower controller to ensure that an event has been completely read out before a token is sent to initiate the readout of the next event. The requirement that each controller chip always send a data packet even if it found no hits makes this possible. The two buffers in the controller chip are managed in as simple way as possible. Each time a new read-event command is received, the chip changes buffers into which the data are read, and each time a new token is received, it changes buffers from which the data are read.

The Front-End Readout Chip

The front-end readout chip has a minimum of logic circuitry, the digital part consisting mainly of memory and shift registers. It does *not* have a continuously running global clock. It is initialized by loading a single long shift register, different parts of which serve different functions. It is controlled during data taking by a serial command line and a trigger strobe input. All digital inputs and outputs are differential, thus requiring two pads each. The major components, illustrated in Figure 2, are

- 64 amplifiers and discriminators, each essentially equivalent to what has already been prototyped and tested in the 32-channel beam-test chip (there are some small detailed changes in the shaper and comparator designs). The channels are numbered from 0 through 63, with channel 0 being on the left when looking from the chip toward the detector.
- 64-bit calibration mask, used to mask the calibration pulse such that only the specified channels are stimulated. The amplitude of the calibration pulse is derived from an internal DAC, while the timing of the pulse is determined by a command string input into one of the command decoders.
- 7-bit threshold DAC, used to set the discriminator threshold. It is really a 6-bit DAC, with a 7th bit used to select one of two ranges. The low range is for normal operation, while the high range is for running threshold scans with relatively large input signals. The step size of the low range is 6 mV (about 0.05 fC), while that of the high range is 24 mV.
- A second, almost identical 7-bit DAC, used to set the level of the calibration pulse. The step size of the low range is 6 mV (roughly 0.3 fC charge injected), while that of the high range is 4 times as large.
- 64-bit trigger mask, used to mask individual channels from contributing to the fast-OR.
- 64-input logical OR of the trigger mask outputs to generate the fast-OR.
-

flag in the control register is set to pass data and fast-ORs to the right.

- Yet another OR gate combines this with the fast-OR output of the adjacent chip on the right-hand-side and sends the result to the chip on the left, but only if the left-right flag in the control register is set to pass data and fast-ORs to the left. Thus when 25 chips are put together in series, we get two 1600-channel OR's of the discriminator outputs, one exiting at each end of the chain, but only one enabled at a given time.
- 64-bit channel mask, used to mask off the discriminator outputs of noisy channels to prevent them from loading into the FIFO.
- A 64-input OR of the channel-mask output, used to determine whether the chip has any data to send out for a given trigger.
- 8-deep FIFO buffer, used to latch the output of the discriminators upon receipt of a trigger and to buffer the data until a read-event signal is received. A 65^{th} bit of each row serves to latch the output of a 64-fold logical OR of the channel-mask outputs.
- Two shift registers, into one of which the data from the FIFO are transferred upon receipt of a read-event signal. One of the registers shifts from left to right and the other from right to left. The registers connect into the neighboring chips to form two 1625-bit registers. A bit in the control register determines which one is used. A 65^{th} bit in each output register serves to hold and pass on the information as to whether the chip had any hits. A multiplexer is used to route the data from the previous chips around the shift registers in the case that this chip has no hits. A chip with no hits outputs only a single zero bit.
- 207-bit control shift register, which includes the channel mask, calibration mask, the settings for the DACs, and the flag controlling the readout direction. It can be loaded via either of the two command decoders. Each bit has a default upon power-up. As it is being written, it's previous contents are clocked out onto a special single-ended CMOS tri-state output. Those outputs are bussed on a single trace on the hybrid that goes to the controller chip. (This would never be used during data acquisition, so there are no noise issues associated with using this single-ended signal.) The output is enabled only if the chip is addressed with its unique address. It is not enabled in case that the wild-card address is used to load the same control-register information into all chips, to prevent multiple chips from simultaneously trying to drive the output line.
- Two redundant command decoders, one driven from the left-hand controller and the other from the right-hand controller. These each have a serial input and a clock input. The clock is pulsed one time for the command start bit, five times for the chip address, three for the command bits, and 207 times (in the case of the load-control-register command) to input the control bits. The decoder is initialized by the power-on reset or the external reset pad. Once initialized, upon receiving additional clock pulses the decoder looks for a high bit on the command line to signal the start of a new command. Each controller chip can load the control register of any front-end chip at any time by sending a clock and appropriate command to the corresponding decoder. Clearly the two controllers should never try to do it at the same time.
- Differential output drivers and differential receivers. All of the outputs send data only as far as the neighboring chip, via short bond wires, except for the last chip, which has to send the signals over short (about 1 cm) traces to the controller chip. Simple differential CMOS signals are used for the data, which swing essentially from

rail to rail (0 to ≈3 V). The fast-ORs use a differential low-voltage-swing output, with a comparator as receiver. The inputs for the clock, commands, and trigger are on the back of the chip and must receive low-voltage-swing differential signals from the controller chips. The fast-OR inputs bias to logic zero (no fast-OR) if not connected. Unused drivers and receivers (*e.g.* the left-going ones in the case that the chip is programmed to read out to the right) are switched off to save power. Both clock and command inputs are always alive, however, to allow the chip to be reprogrammed through either.

- Power-on reset. A reset signal is generated and distributed throughout the chip a short time after the power is turned on.

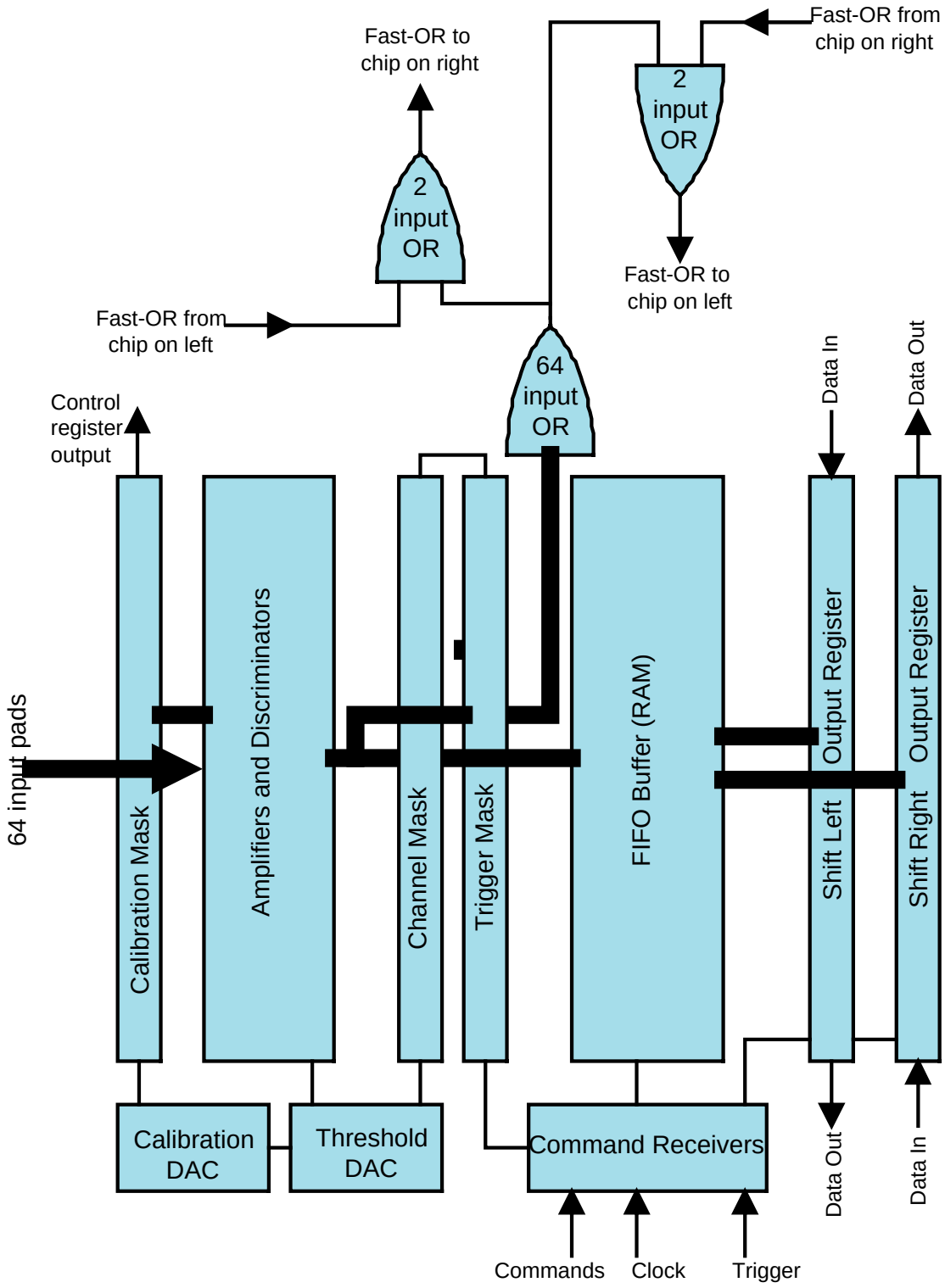

Figure 2. Simplified block diagram of the front-end readout chip.

List of bonding pads for the front-end chip (not counting internal probe pads):

- Front edge of chip (detector side, 64 pads):
  - ◊ 64 input pads.
- Left-hand and right-hand edges of chip. Each edge has 8 pads:
  - ◊ Fast-OR output. 2 pads.
  - ◊ Fast-OR input. 2 pads.
  - ◊ Data shift register output. 2 pads.
  - ◊ Data shift register input. 2 pads.
- Back edge of chip (32 pads).
  - ◊ IREF pad. Reference current for amplifier bias.
  - ◊ AVDD2 pad (analog 2 volts). 2 pads, one on each end.
  - ◊ AVDD pad (analog 5 volts). 2 pads, one on each end.
  - ◊ QVDD pad (analog 5 volts for input transistor N-well).
  - ◊ DVDD pad (digital power). 2 pads, one on each end.
  - ◊ DGND pad (digital ground). 2 pads, one on each end.
  - ◊ AGND pad (analog ground). 2 pads, one on each end.
  - ◊ Hard-wired address. 5 pads.
  - ◊ Command input, left and right. 4 pads.
  - ◊ Clock for the command input, left and right. 4 pads.
  - ◊ Trigger strobe for left and right. 4 pads.
  - ◊ Control register output. 1 pad. This output is single-ended and tristate, since it would only be used for diagnostic purposes.
  - ◊ RESET. Force a reset of the command decoders. 1 pad.
  - ◊ CAL. Input an external calibration signal to override the internal calibration. For bench testing only.

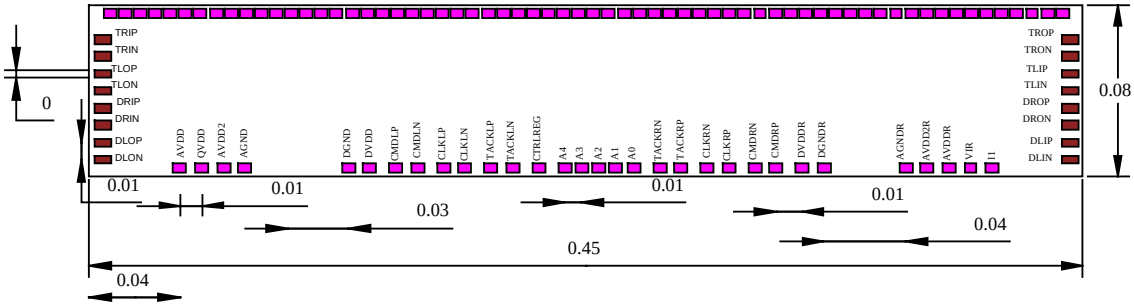

Figure 3. Rough pad layout of the front-end chip.

Control register of the front-end readout chip. Bit-0 is the first bit to be clocked in when loading the register.

- 64-bit calibration mask. Bit zero is for channel 0, and bit 63 is for channel 63. The power-on default is channel 0 set, channels 1 through 3 not set, channel 4 set, channels 5 through 7 not set, and so forth. Only every forth channel is set.
- 64-bit channel mask. Bit 64 is for channel 63, and bit 127 is for channel 0. Every bit is set at power on.

- 64-bit trigger mask. Bit 128 is for channel 0, and bit 191 is for channel 63. Every bit is set at power on.
- 7-bit calibration DAC. Bit 192 is the range select (set=high range). Bits 193 through 198 set the voltage in binary, with bit 198 being the lowest-order bit. The voltage is 6.26 mV × (DAC setting + 1) in the low range. In the high range, the voltage change per step is four times greater. The default is the low range and a setting of 001111, or 100 mV.
- 7-bit threshold DAC. Bit 199 is the range select (set=high range). Bits 200 through 205 set the voltage in binary, with bit 205 being the lowest-order bit. The voltage is set the same as for the calibration DAC, but the power-on default is different: low-range 010111, or 150 mV.
- Left-right control bit, or Bit 206. If the bit is set, then the chip reads out to the right. Otherwise it goes to the left. The default is left.

Here is a list of serial command codes for the front-end chip. Each code is preceded by a start bit (a 1) and followed by a 5-bit chip address, with 1F being the wild card address. Thus a hybrid with up to 31 chips can be handled. With the exception of the 001 command (load control register), the chip ignores commands from the command input that has not been selected. Loading the control register is, of course, necessary for making or changing that selection. The least-significant bit is always first in time. At least one zero must separate any two commands. The clock may be turned on well before sending the command, or the first clock edge may appear immediately after the first rising edge of the command.

- 000: no-op.
- 001: set the control register input mux and clock the next 207 bits into the control register. Obviously the chip should not receive this command from both left and right at the same time. This write command also can serve as a read-control-register command, since the previous contents are clocked out as it is loaded. If the power-up reset doesn't work, it may be necessary to send this command, with address zero and followed by a few hundred clocks, to ensure that the command decoder ends up in the proper state for receiving a command (*i.e.* with all the internal counters reset to zero). The minimum number of clocks needed is 218.
- 010: read event—move data from the FIFO to the output registers and then direct the clock to the correct output register. Needs to be used in conjunction with the end-read-event command. Needs at least 10 clocks plus the number of data bits to be shifted. The minimum number of data bits that can be shifted is 11. If it is desired to shift no data bits, then the read-event command cannot be sent (use clear-event instead).
- 011: calibration strobe. Requires a minimum of 522 clocks.
- 100: clear the first event from the FIFO. Needs at least 12 clocks.
- 101: reset the chip, including all registers and the FIFO. Needs at least 12 clocks.
- 110: reset just the pointers in the FIFO. Needs at least 12 clocks.
- 111: end read event. This disables the clock to the output shift register, so that it will not shift while the next command is being interpreted.

The Hybrid Controller Chip

The controller is a synchronous chip, with a clock (20 MHz) that is always running. It must handle the following functions:

- Receive and decode commands and receive trigger signals.
- Send initialization information to the front-end chips. This is done by passing a load-control-register command on to a given front-end chip, along with the appropriate data.
- Determine whether the hybrid has any data for a given trigger. In order to do this, the controller must start a latency counter each time its hybrid generates a fast-OR. The counter should be active for only the maximum trigger latency (about 1.3 μs). Then, if that counter is not still active when a trigger is received, the controller can assume that its hybrid did not contribute to that particular trigger. If the counter is active, then a flag is set in an 8-deep buffer, as discussed in the following point.
- Keep an 8-deep FIFO list of flags, parallel to the event buffers in the front-end readout chips. A flag is set if the corresponding event is to be read out. If a read-event command is received and the flag at the top of the buffer is not set, then the data from that hybrid are not shifted out. Instead, a clear-event command is sent to the front-end chips.
- When a read command is broadcast to the controllers, send a read command to the front-end chips along with clock pulses to shift the data into the controller chip, or else send a clear-event command if the readout flag is not set. In the case of a read command, after the appropriate number of clock pulses an end-read command is sent and the clock then turned off. The number of clock pulses needed is specified by bits in the control register: at least 10 plus the maximum number of data bits to be shifted in.
- Count the bits coming in to generate a list of hits (up to 63 hits in length, each with 11 bits). Two buffers are needed for this, so that the list can be read out while another is being accumulated.
- Calculate an 11-bit check-sum for each data packet and include it in the packet, if the corresponding enabling bit is set in the control register.
- Send data down to the next lower layer in the tower in a token-controlled protocol. Pass data from the next higher layer if the token is not present.
- Send calibration and initialization commands to the front-end chips, including the appropriate number of clock pulses.
- Calculate the time-over-threshold of the fast-OR signal and send it along in the header. A first guess for the appropriate counting speed is 5 MHz, but that needs more thought put into it. 5 MHz with 10 bits would give a range of 204 microseconds, which is good for up to about 50 MIPS.
- Two such controller chips are present on each hybrid, one at each end. Either one can initialize any or all of the front-end chips on the hybrid, and either can read out all of the chips, or any contiguous set of them that includes the chip next to the controller.

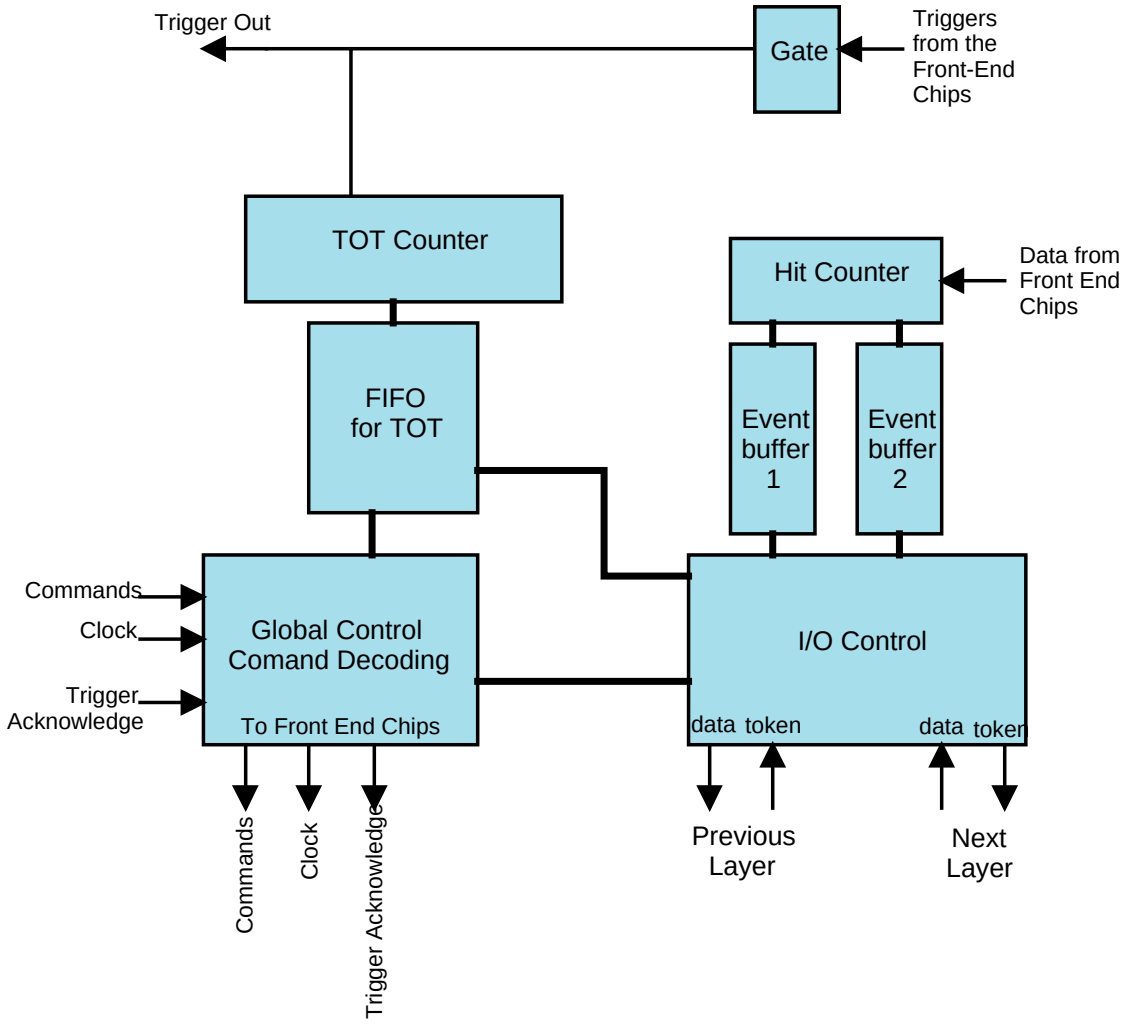

Figure 4. Simplified block diagram of the controller chip. Many internal connections are not shown.

Here is a list of commands to be interpreted by the controller chip. The format is *saaaaacccddd*…, where *s* is the start bit, *a* is the address of the layer, *c* is the command code, and *d* is the data string. Not all commands include data. The highest-order bit of the command is sent first.

1. Load control register of the controller chip. This also results in the previous contents of the control register being sent out the data line. (000)
2. Send a clear event command to the front-end chips. This would be used, for example, in case that calorimeter information is used to reject an event after the trigger signal has already been sent to the front-end chips. (001)
3. Read event. This initiates clocking of the front-end data register into the controller chip and suppression of the zeroes, after the read-event command has been passed on to the front-end chips. An end-read-event command is generated automatically by the controller chip. If the read flag is not set, then a clear-event command is sent instead of read-event. Normally addressed to all chips at once. (010)

4. Load control register of a front-end chip. This can only be addressed to a single controller chip and a single front-end chip at a time. The data *d* hold a command string to be passed to the front end chip (chip address plus the front-end chip read-control-register command). The contents of the control register come back on a single-ended CMOS line driven by tri-state buffers. The controller chip should input those bits and pass them out onto the data output line. (011)
5. Turn on the clock to the front-end chips. (100)
6. Send calibration strobe command to the front-end chips, along with at least 522 clock pulses. (101)
7. Send the data as a command to the front-end chips. In this case, the data include the full command string for the front-end chips, including the chip address. The only commands that should be sent this way to the front-end chips are RESET and RESET FIFO. (110)
8. Reset the controller chip. (111)

Control register of the controller chip:

1. Bits 0–4. Number of chips to read from the front-end data register. 5 bits. (Default: 5 chips.)
2. Calculate and include an 11-bit check sum. (1=default=yes)
3. No longer used.
4. Read out the layer even if it did not generate a fast-OR? 1 bit. (1=default=yes)

Pad count for the controller chip:

◊ Fast-OR input from front-end chips. 2 pads.
◊ Data input from front-end chips. 2 pads.
◊ VDD power pad.
◊ GND ground pad.
◊ Hard-wired address. 5 pads.
◊ Output commands to the front-end chips. 2 pads.
◊ Output clock for commands to the front-end chips. 2 pads.
◊ Trigger strobe to front-end chips. 2 pads.
◊ Read token in from previous layer. 2 pads.
◊ Read token out to next layer. 2 pads.
◊ Data out to previous layer. 2 pads.
◊ Data in from next layer. 2 pads.
◊ Commands in. 2 pads.
◊ Trigger strobe in. 2 pads.
◊ Clock in. 2 pads.
◊ Fast-OR output. 2 pads.
◊ Hard reset. 1 pad.