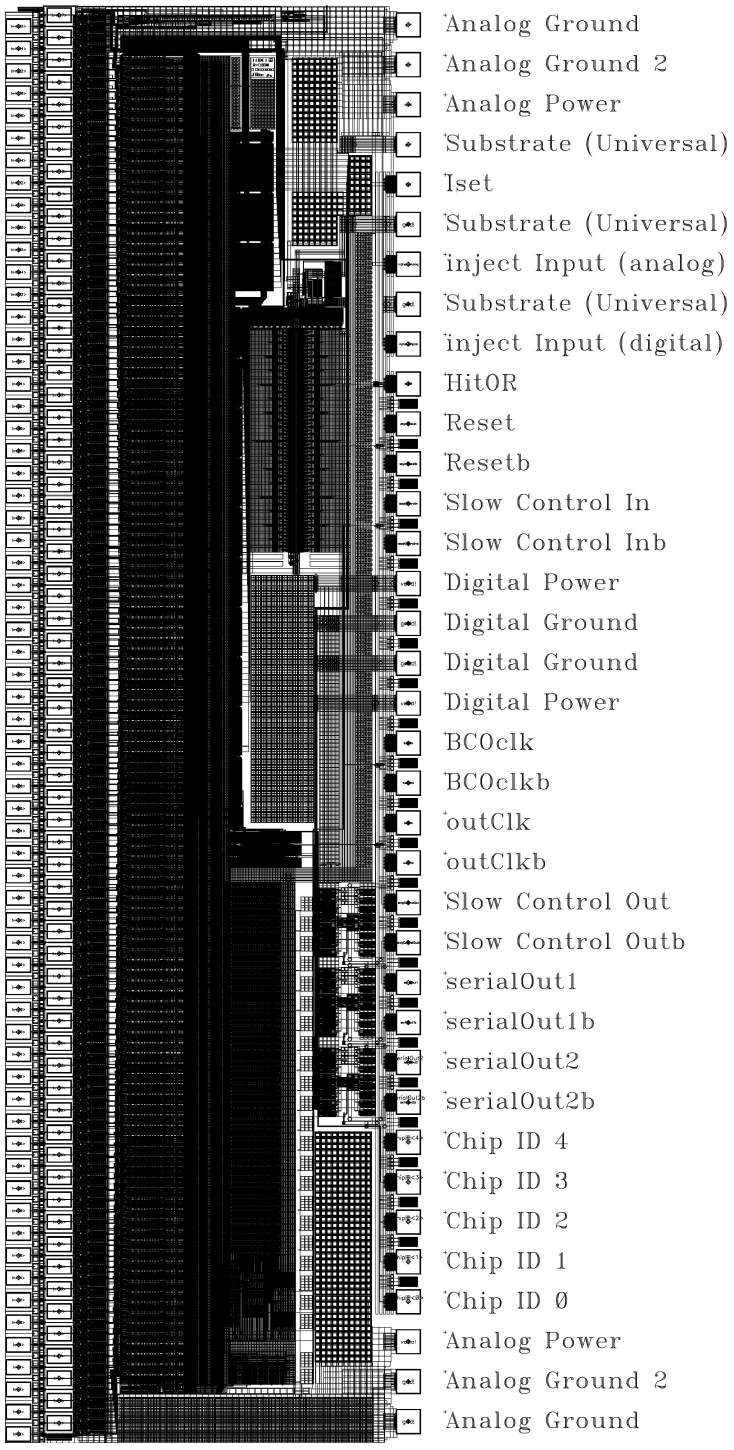
FPHX Chip Documentation

Tom Zimmerman
Jim Hoff
Fermilab

6/4/09



The FPHX chip

Introduction

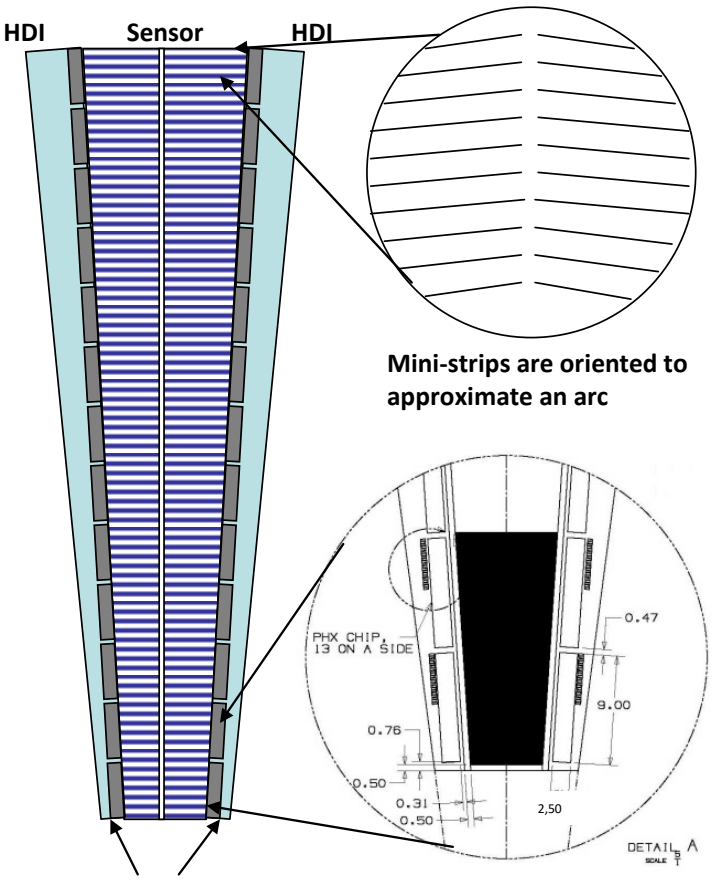
The FPHX is the custom readout chip designed for the FVTX Silicon Sensor Wedge in the PHENIX experiment, as shown in Figure 1. A first prototype chip was tested beginning in 2008 and found to be functional. A second prototype chip, containing some bug fixes and design modifications, was submitted in June 2009.

The experiment will use a 106 ns beam clock. Each FPHX chip integrates and shapes (CR-RC) signals from 128 channels of mini-strips, digitizes and sparsifies the hit channels each beam crossing, and serially reads out the digitized data. The chip is designed only for positive charge (hole) collection from p-on-n detectors. The strips used by the experiment will have different lengths, resulting in a varying distribution of input capacitance across different chips. For the most part, activity is rare, but when there is an event, it is expected that there will be an average of 2.8% hit pixels per chip, with the possibility of much higher occupancy. Long latency cannot be tolerated, and it is therefore required that four hits will be read out within four beam cross-over periods of an event. As shown in Figure 1, a number of FPHX chips will be controlled on a single High Density Interconnect (HDI). Space on the HDI is limited, which is why the data from events must be serialized onto one or two pairs of LVDS digital lines per chip. In addition, the FPHX must be as "self-sufficient" as possible and provide its own internal bias voltages and currents with minimal external support circuitry. The user must be able to control internal parameters and biases, therefore a digital slow control interface is provided on each chip to enable programming.

The FPHX is a mixed-mode chip with two major and distinct sections, the front-end and the back-end. Figure 2a is a block diagram of the complete chip, and Figure 2b is the top level full chip schematic, with the two major sections outlined. The mostly analog front-end contains the 128 channels of integrators, shapers, and comparators, in addition to several programmable bias circuits and DACs that are used for setting internal parameters. The output of each front-end channel is simply an 8-bit digital word from 8 comparators (forming one hit discriminator output and seven thermometer-coded ADC outputs, which results in 3-bit magnitude information). Each comparator's threshold is independently programmable, effectively allowing a custom non-linear ADC. The hit and ADC information serves as the input to the Core Logic section of the back-end, which processes the data for readout. Also part of the back-end is the Slow Controller, which accepts a serial data stream to program the chip after power-up.

This document is organized into two sections, one for the front-end portion of the chip and one for the back-end.

FVTX Silicon Sensor Wedge (Prague, UNM, LANL)



Sensor

- 2 columns of strips
- 1664 strips per column (large)
- 640 strips per column (small)
- strip length ~3.45 mm to ~11.2 mm (large)
- strip length ~3.45 mm to ~6.8 mm (small)
- 75 micron spacing
- 48 wedges per disk (7.5˚/sensor, 15˚/wedge)
- 0.5 mm overlap with adjacent wedges

FPHX Chip

- 1 column readout
- 128 channels
- ~ 70 microns channel spacing
- Dimensions – FNAL ~9mm x 2.5 mm

Figure 1.

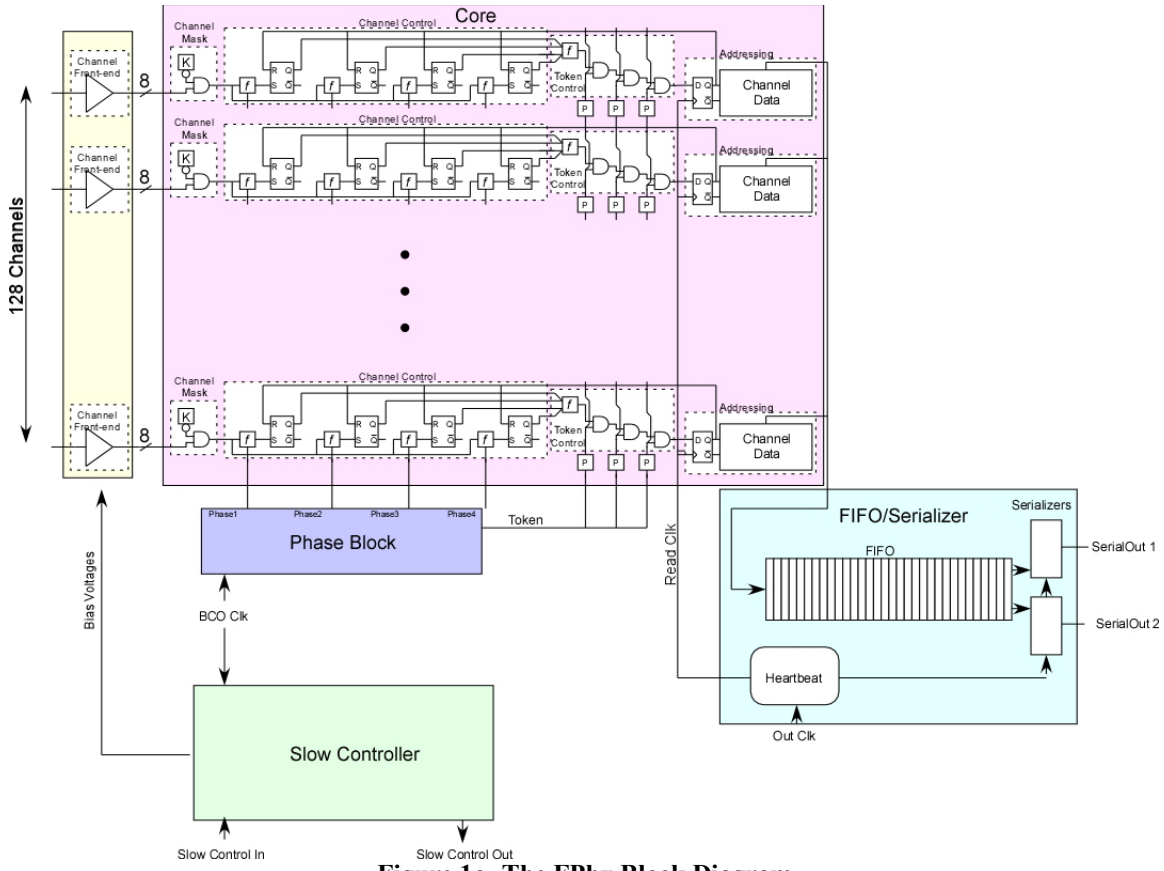


Figure 1a- The FPhx Block Diagram

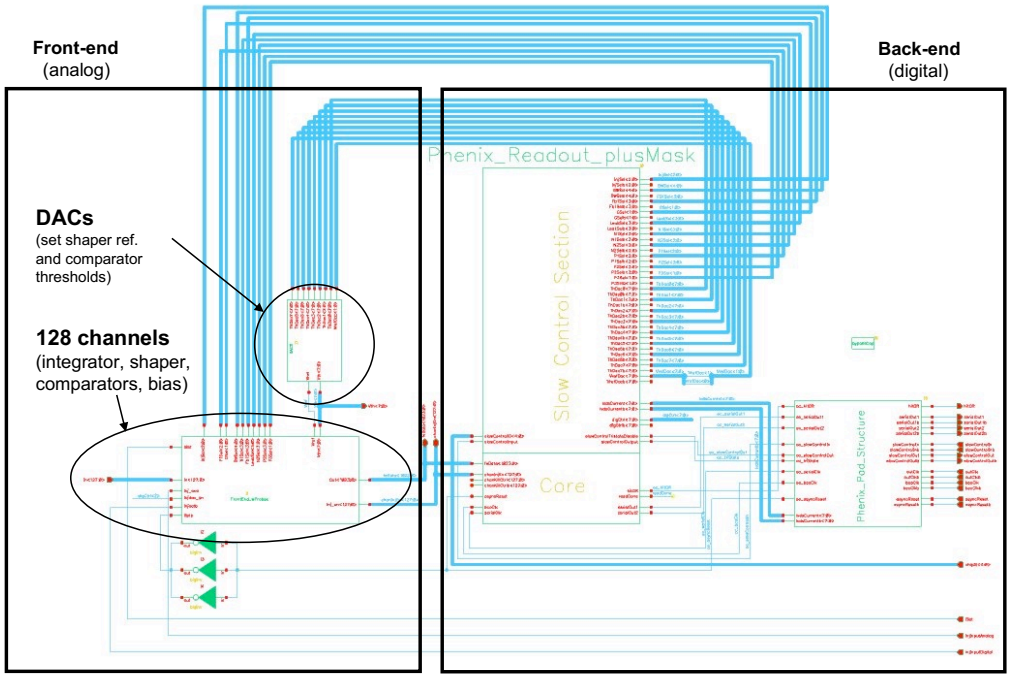


Figure 2b. FPHX full chip schematic

The FPHX front-end

This section serves to document the front-end section of the FPHX chip. Specifications, schematics, programmable parameters, simulation results, and the recommended HDI layout will be covered.

Front-end Specifications

- **Configuration:** 128 identical channels. Each channel contains a charge integrator, shaper, programmable threshold discriminator, and an 8-comparator thermometer ADC with fully programmable thresholds.
- **Input charge signal polarity:** positive (holes)
- **Input capacitance:** the design is optimized for an external detector capacitance of approx. 1.5pF (and estimated chip parasitic C of 0.5pF)
- **Detector leakage current compensation:** programmable maximum of 0 – 50nA
- **Effective Channel gain:** 46, 50, 60, 67, 85, 100, 150, or 200 mV/fC (programmable)
- **Output:** 3-bit digital code per channel (digitized shaper output pulse height). No analog output.
- **Output pulse dynamic range (shaper output)**: >800mV (25ke to 100ke, depending on the gain setting)
- **Nominal output pulse peaking time:** 60 ns (programmable, set by the shaper)
- **Output pulse fall time:** programmable, set by the integrator fall time adjust
- **Noise at shaper output:** (approximate from simulations)
- At max. input transistor bias of 38uA: 115e + 134e/pF (linear for total Cin>=2pF, including chip parasitics of approx. 0.5pF)
- At input transistor bias of 14uA: 110e + 196e/pF
- **Power:** Approx. 140 uW per channel for the maximum input transistor bias current of 38uA, 70 uW per channel for minimum input transistor bias current of 10uA.
- **External biasing components required:** one resistor to ground to set master Iref
- **2 test pulse inputs:** one direct analog, one controlled by a DAC and digital input.

Schematics

Figure 3 is the "128 channels" schematic, which includes everything in the front-end except for the DACs that generate the comparator thresholds. The core of this schematic is the block containing 128 channels of integrator, shaper, and comparators, whose output is 1024 digital lines that feed the back-end. A bias generator block provides the necessary biases for the integrators, shapers, and comparators. The bias generator is programmable to allow for adjustment of critical analog parameters, through downloading to the program register in the back-end Slow Controller. A "clean digital"

buffer powered by the analog supply is used to prevent any back-end digital noise from entering the bias generator. A test pulse inject circuit is provided to allow a pulse of programmable magnitude to be injected to any number of the 128 channels simultaneously. A programmable inject enable for each channel also comes from the back-end program register to allow for injection of arbitrary channel patterns.

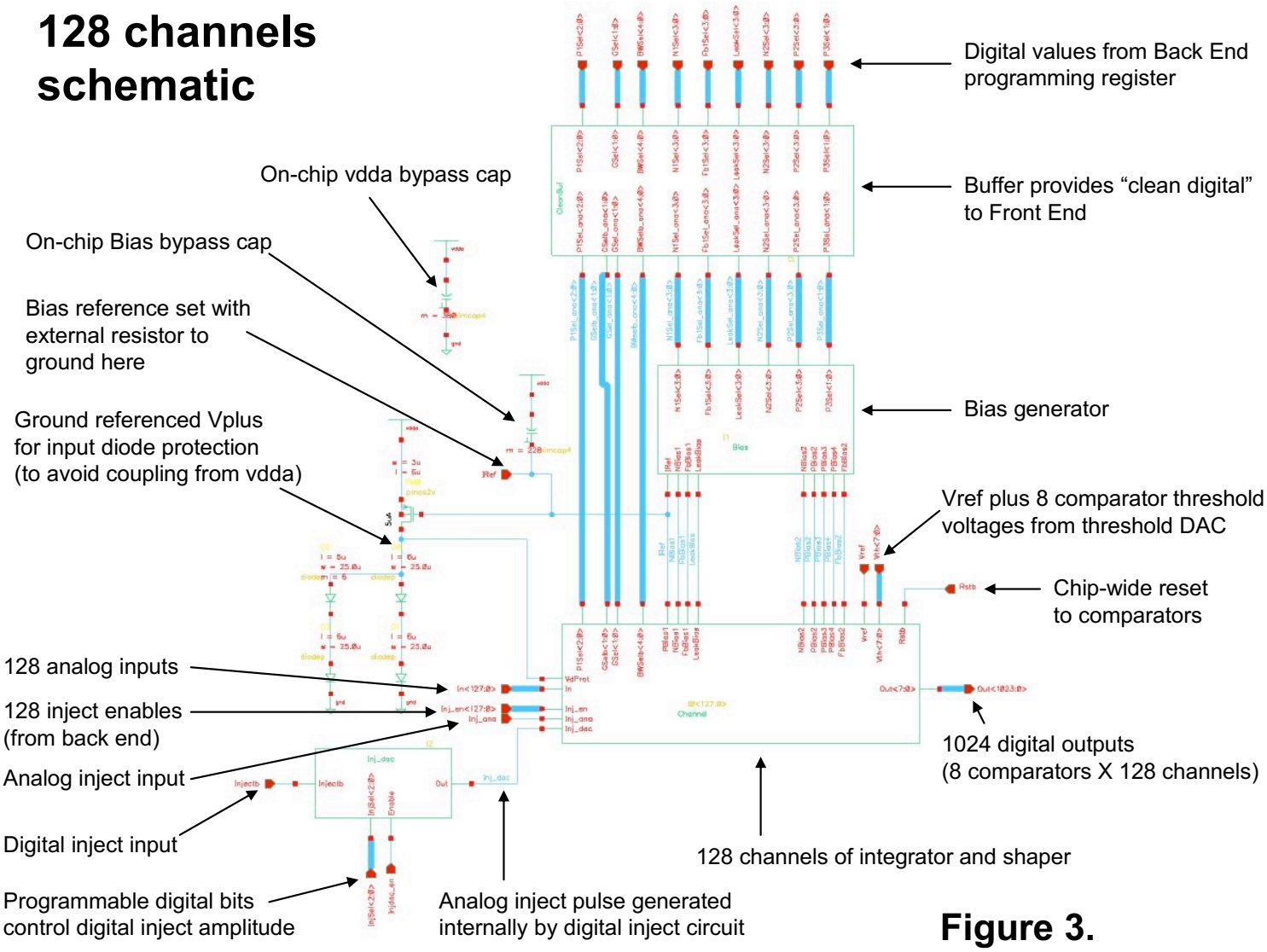


Figure 3.

Figure 4 is the same schematic with “test pads” added. The integrator and shaper output of channel 0 are buffered in order to be able to drive an oscilloscope FET probe for testing on the prototype chip. Some comparator outputs on several channels are also buffered. These circuits may or may not be included on a production chip.

128 channels with test pads schematic

Test pads added for diagnostic purposes (on prototype run only)

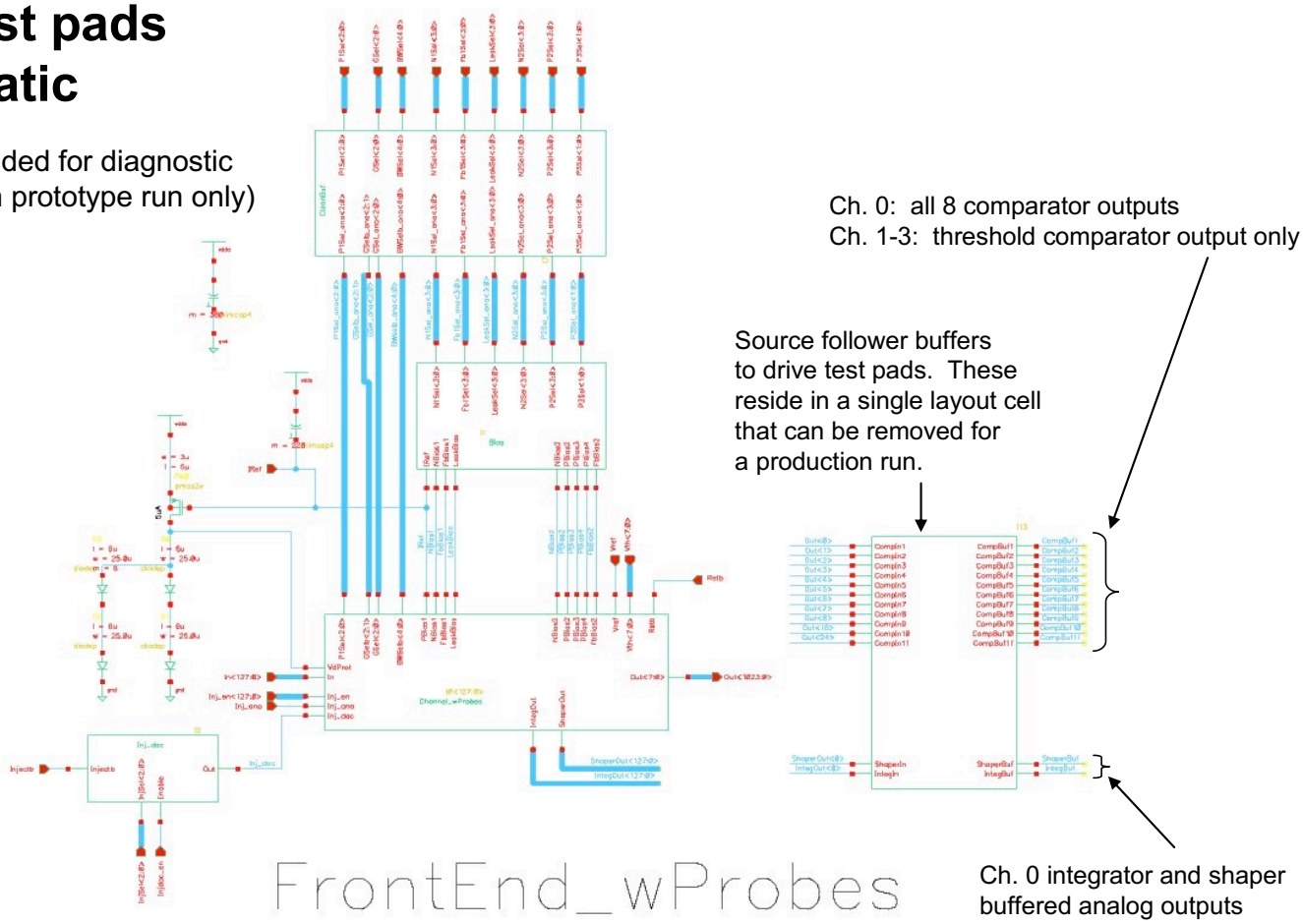


Figure 4.

A top-level schematic of one individual channel is shown in Figure 5. It consists simply of an integrator driving a shaper, which in turn drives comparators with programmable thresholds. Figure 6 is a more conceptual diagram of this channel. When the integrator receives a charge input impulse, it produces a fast (relative to the shaper peaking time) output step followed by a programmable discharge time, giving the "CR" portion of the CR-RC response. The integrator bias level, gain, and bandwidth are programmable in order to allow optimization of noise and response time for a variety of total input capacitance values (typically from 0.5 pF to 2 pF). The shaper has a fixed gain and limits the signal bandwidth, which determines the signal rise time and peaking time. The nominal peaking time of the shaper is intended to be about 60 ns, so that there will be no ambiguity about which 106 ns beam clock period produced the signal. The shaping time can be adjusted via the programmable shaper bias.

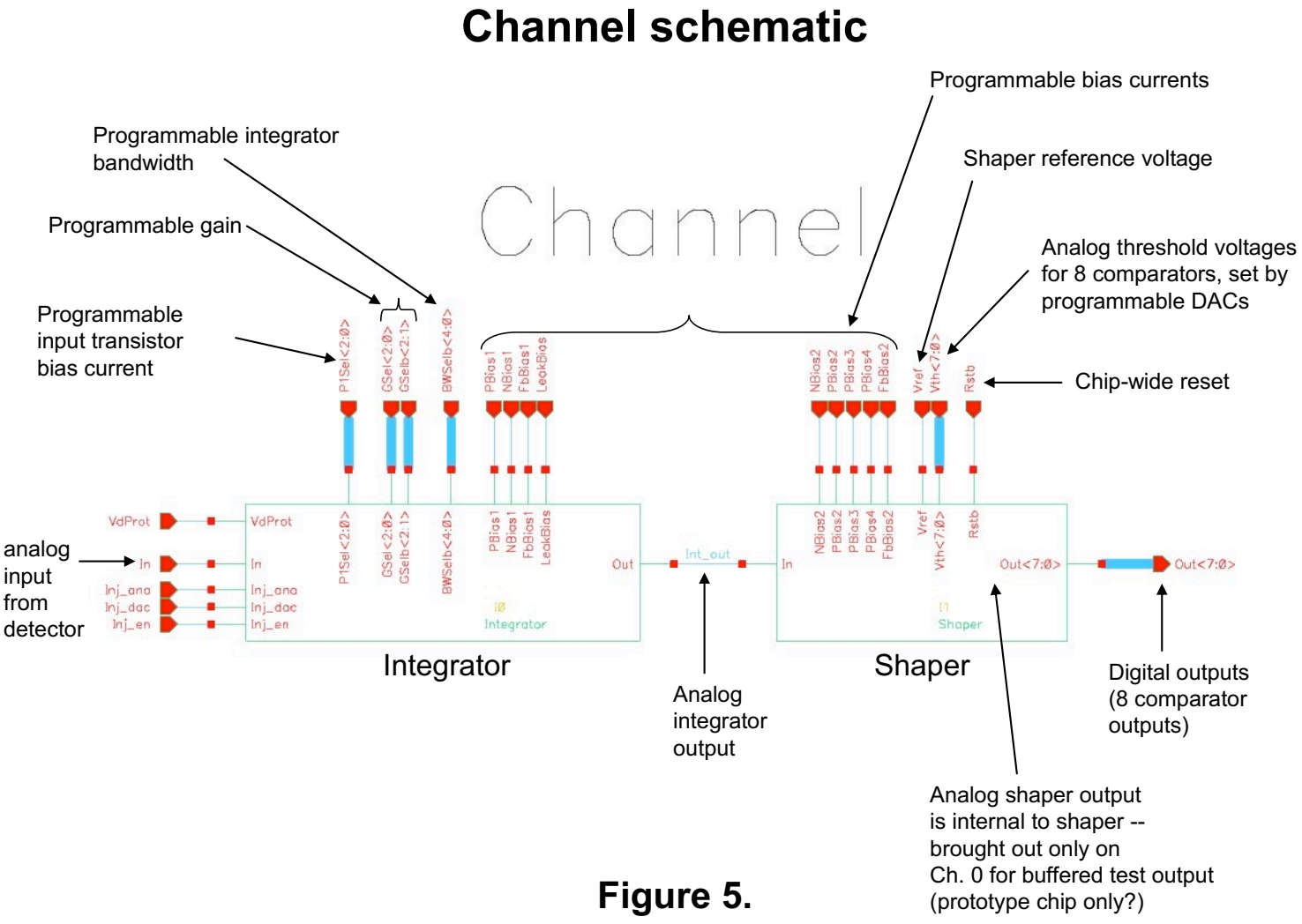


Figure 5.

A new type of continuous reset and leakage compensation circuit has been designed for the integrator. In Figure 6, a programmable current labeled Ifb sets the effective reset resistance. Mfb acts exactly like a resistor of value 1/gm for small signals, where gm depends directly on Ifb since Mfb is in weak inversion. This programmed resistance, in conjunction with the feedback capacitor, sets the RC fall time (return to baseline) of the shaped signal. For large signals, when the output swings more than a hundred mV or so, Mfb no longer looks like a resistor, but a constant current source. The return to baseline then becomes linear until the output approaches the reset level, at which point it becomes an exponential RC decay again. The current through Mfb is supplied by one leg of a differential pair that is also used to absorb any DC detector leakage current at the input. The programmed current labeled Imax determines the maximum value of available leakage current compensation. With this configuration, the circuit response is very insensitive to changes in DC leakage current. The equivalent resistance of the shaper feedback is referenced to the integrator feedback circuit, and is set to have much higher resistance so that it has little impact on the output pulse shape. The shaper baseline is set with a programmable reference voltage (Vref), and the output pulse feeds eight programmable-threshold comparators, the first of which is used as a hit discriminator. The other seven comparators, along with the hit discriminator, form a simple thermometer-code "flash ADC." The thresholds are referenced to the shaper baseline Vref.

Simplified conceptual diagram of one channel

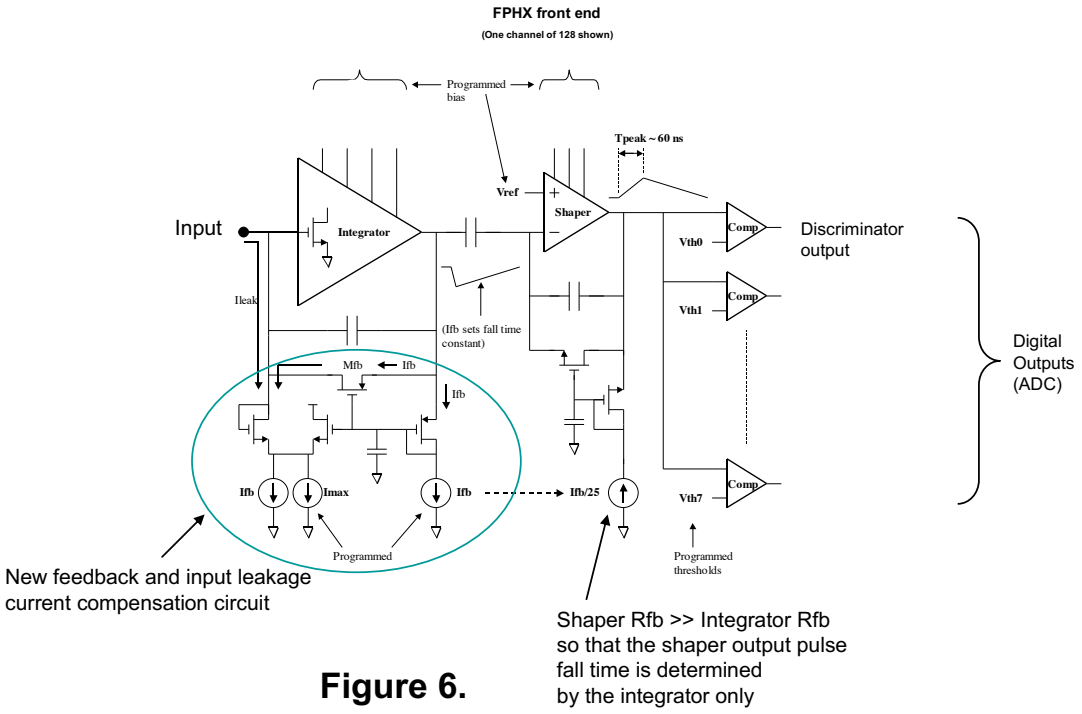


Figure 6.

The integrator is shown in detail in Figure 7. The input transistor feeds a regulated cascode (to keep the open loop gain high), and is biased by a cascode current source. Additional input transistor bias current is programmed through switched current sources. The integrator bandwidth is adjustable via switched capacitors on the amplifier dominant node. The transfer gain is set with programmable feedback capacitors. A source follower drives the integrator output.

Integrator schematic

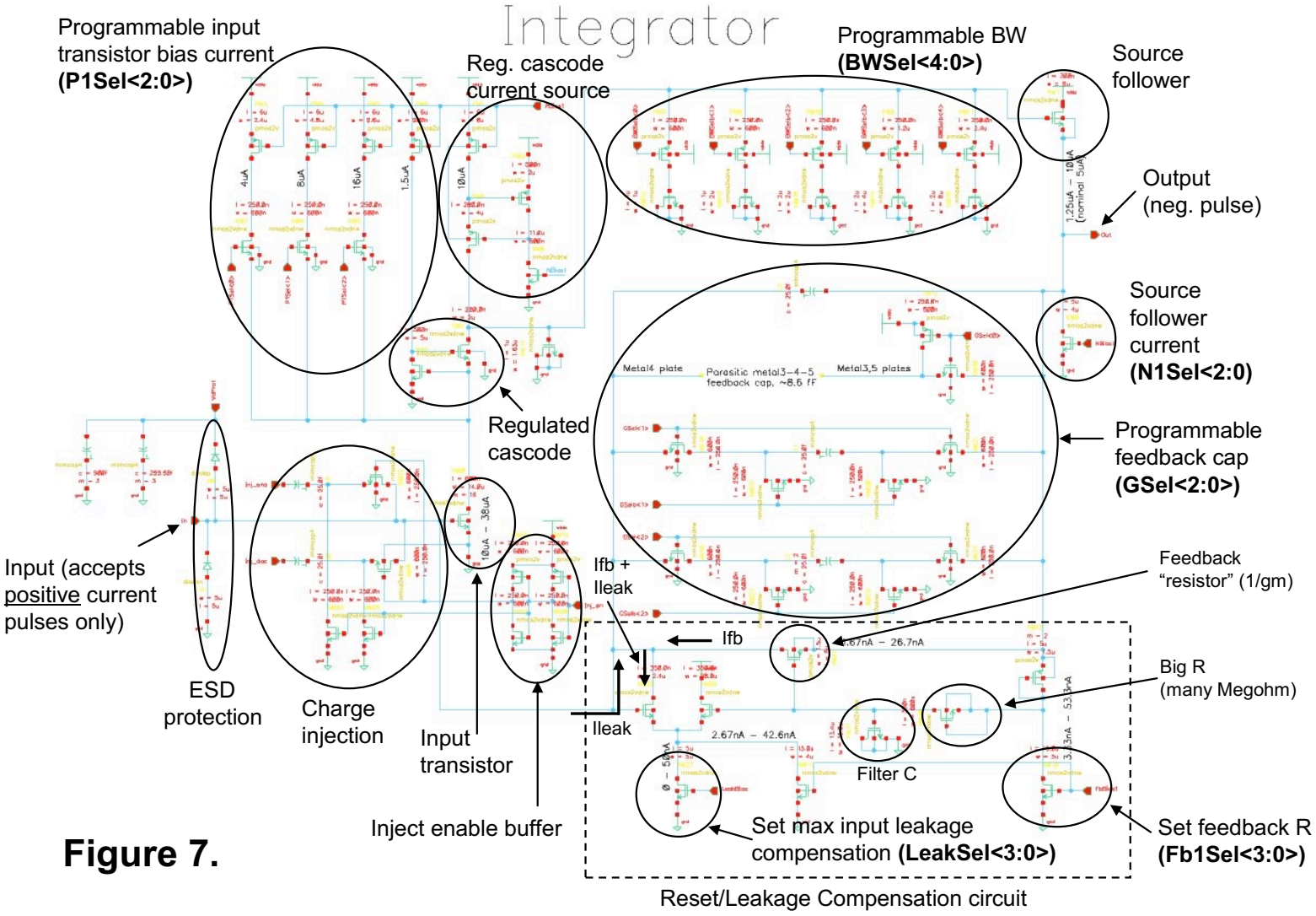


Figure 7.

The shaper is shown in detail in Figure 8. The heart of the shaper is a simple differential opamp structure, with programmable bias to set the bandwidth. The shaper voltage gain is set to approximately 5 by the input/feedback cap ratio. The shaper drives eight comparators (which also reside on this schematic), so requires a source follower at the output. Because the comparators represent a load that varies with signal amplitude, the source follower must be inside the shaper feedback loop to preserve linearity. The shaper reference, or DC operating point, is set by the Vref input. Note that the Vref voltage does not go directly to the opamp input, but is first buffered and level shifted by a PMOS source follower. The actual operating point of the shaper is then Vref + Vgs(PMOS). (This will be an important point in understanding the operation of the comparators).

The shaper has a high resistance across the feedback capacitor for continuous reset, which is derived from the integrator feedback bias. This resistance is ratioed to the integrator feedback resistance so that it is much higher than the effective integrator feedback resistance. This insures that the shaper feedback resistance does not significantly affect the signal fall time. Eight identical comparators, shown as a block on the schematic, are driven directly by the shaper output.

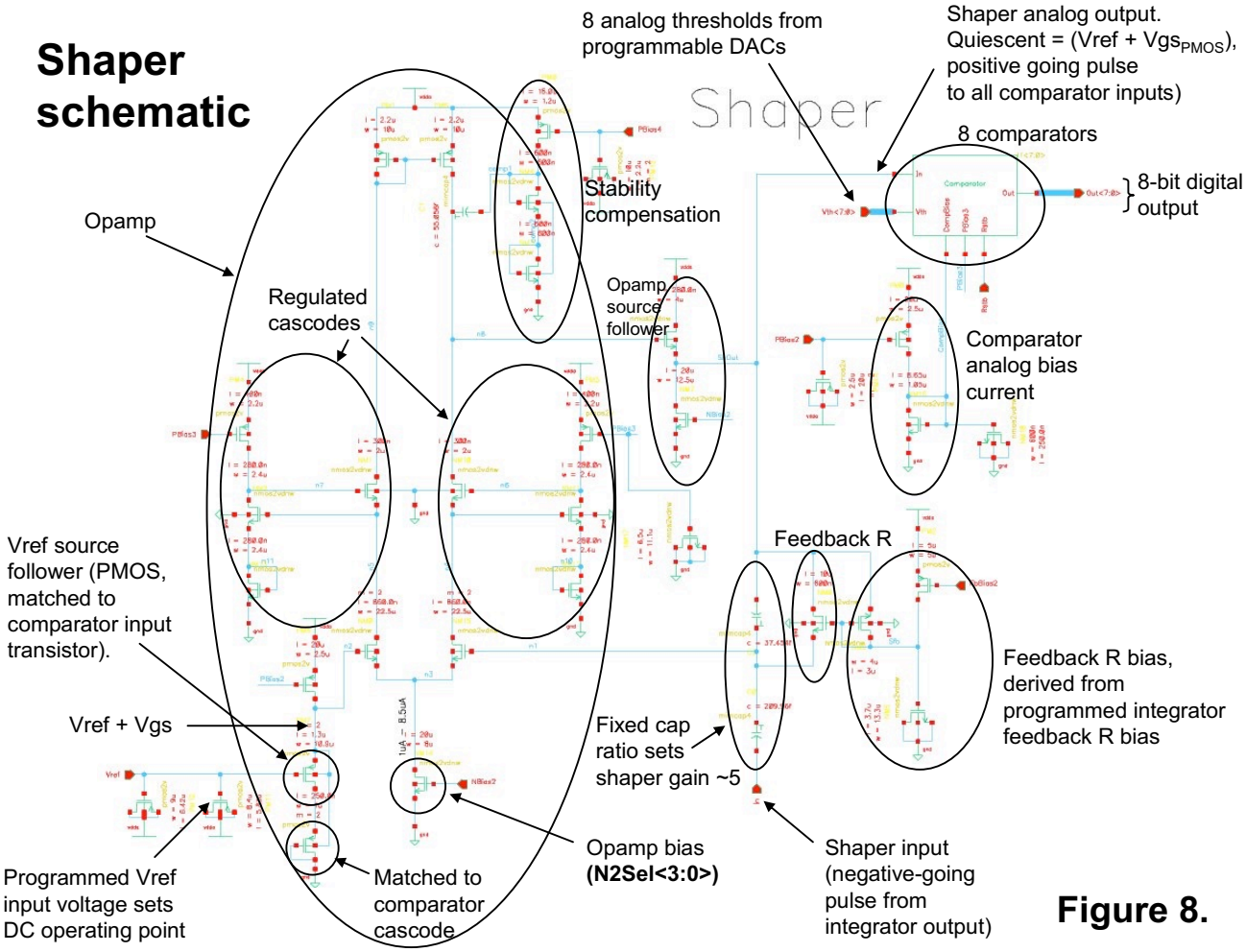


Figure 8.

The comparator design is shown in Figure 9. The heart of the comparison operation is essentially a single PMOS transistor (in the upper left corner of Figure 9), with the input applied to the source and the threshold applied to the gate. The threshold is set to a programmable amount above Vref. The comparator input sits quiescently at Vref + Vgs(PMOS) due to the source follower at the shaper input, as explained previously. Since the comparator consists of a PMOS matched to the source follower, the Vgs(PMOS) is theoretically subtracted out, and an excursion on the shaper output of more than (Vthreshold – Vref) turns on the comparator transistor. One advantage of this configuration is that with no signal, the comparator PMOS transistor draws no quiescent current – it only turns on temporarily during the time that the signal exceeds threshold.

The comparator transistor is biased with a simple NMOS current source. This current is programmable and affects the comparator delay. The output of this simple stage feeds more inverter gain stages, the final ones being powered by the digital supply. The threshold is not applied directly to the comparator transistor gate, but is passed through a switch so that the gate can be disconnected from the threshold when the comparator flips. This serves two purposes – kickback from the comparator to the threshold voltage is avoided, which keeps the threshold clean, and the act of switching also introduces some hysteresis, which makes sure that the comparator flips completely.

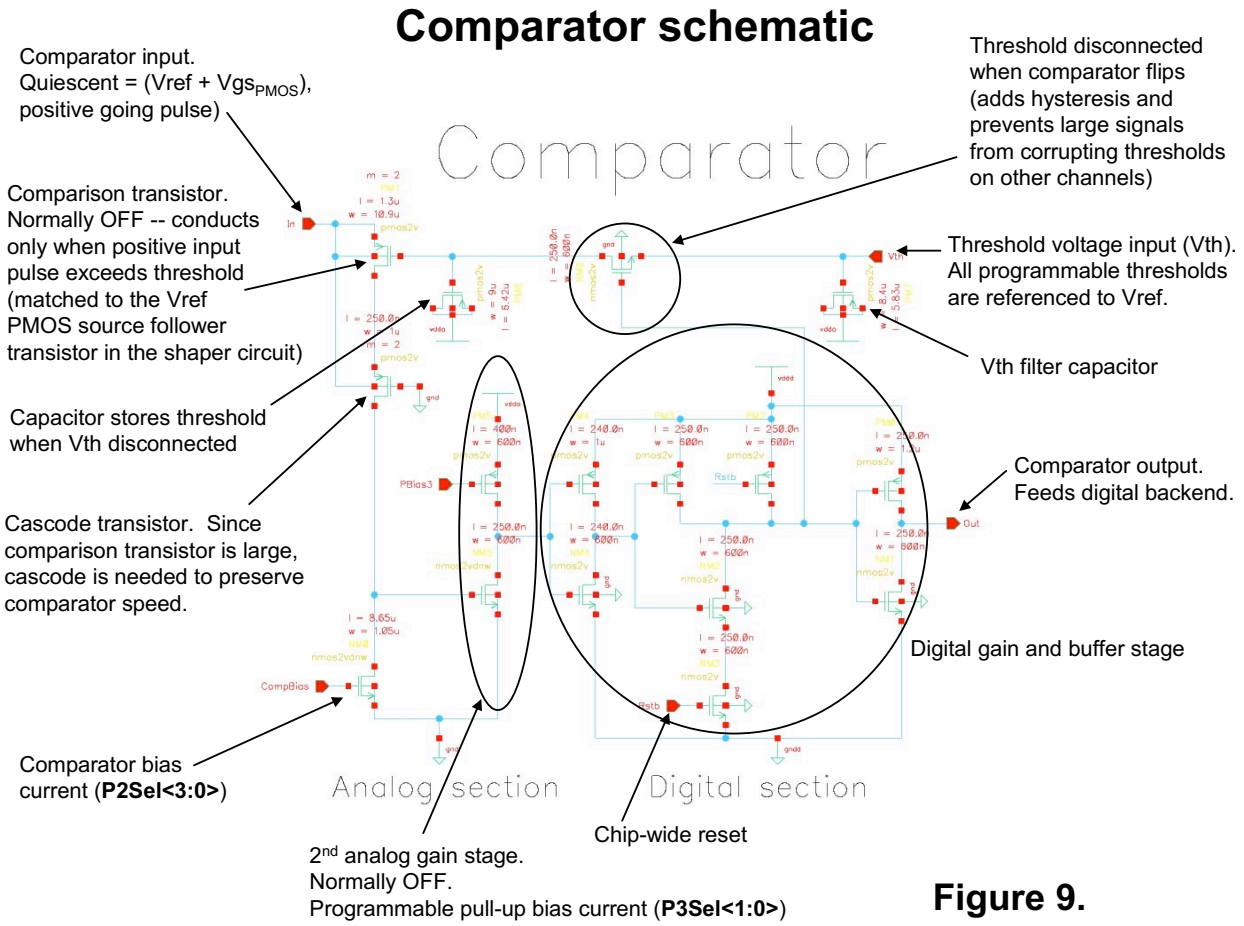


Figure 9.

Figure 10 is a schematic of the programmable bias section, which sets various biases in the integrator and shaper. Digital program bits from the back-end simply control the magnitude of current sources that are used to generate the required bias currents. Only one external resistor is needed to establish a master reference current.

Bias schematic

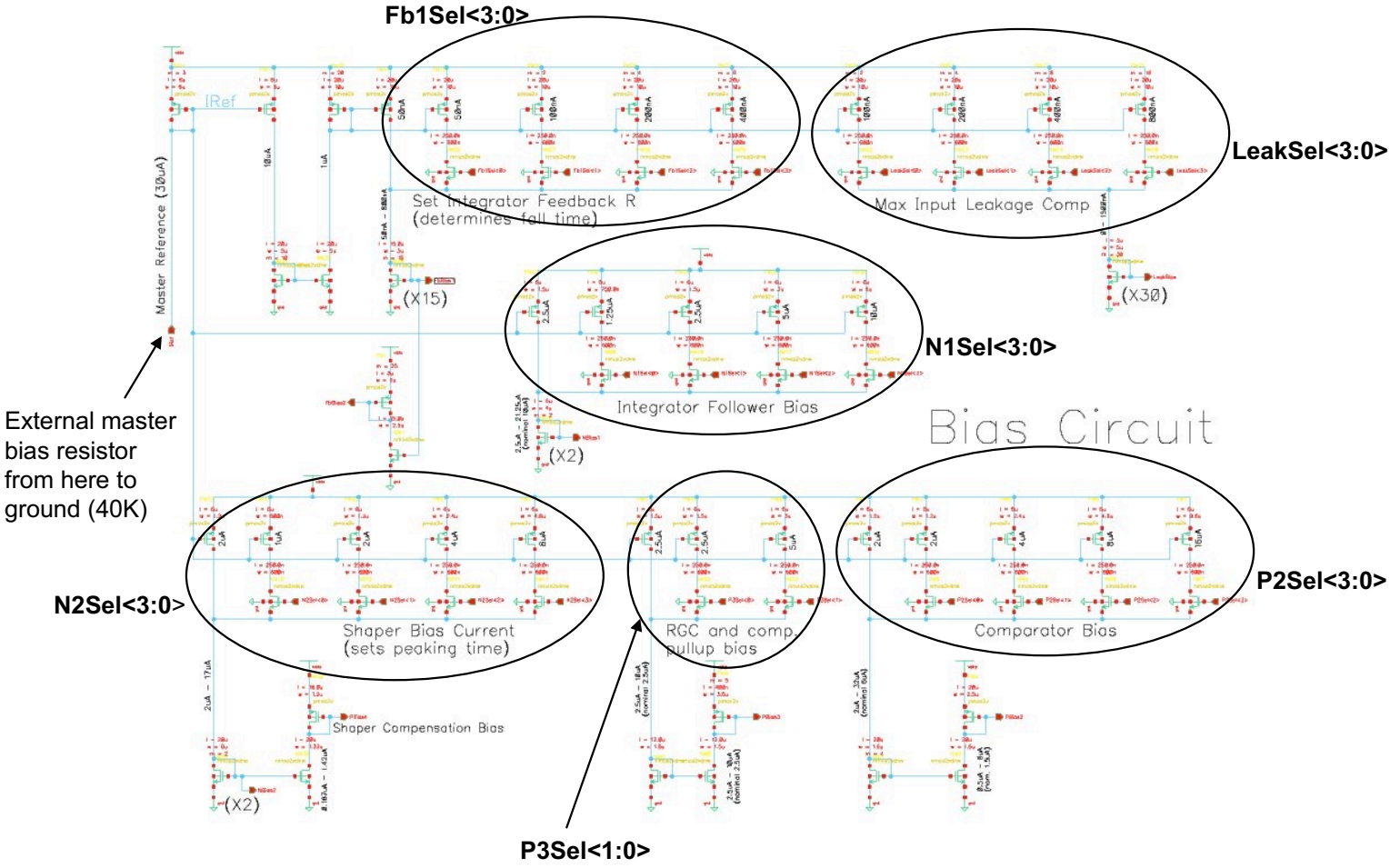


Figure 10.

As mentioned previously, a test charge injection circuit is provided in order to inject a programmable magnitude charge into a selected set of channels. This circuit is shown in Figure 11. A resistor ladder generates a reference voltage and a set of eight DC voltages that are binary weighted above the reference. A 3-bit digital code selects one of the binary weighted voltages with an 8-transistor analog multiplexer. A digital inject input pulse causes a second analog multiplexer (2-to-1) to switch from the reference voltage to the selected one-of-eight voltage, yielding an analog voltage step of programmed magnitude. This is then buffered by a source follower so that it can drive a 25fF injection capacitor on the input of each channel (25fF X 128 = 3.2 pF total).

Inject DAC schematic

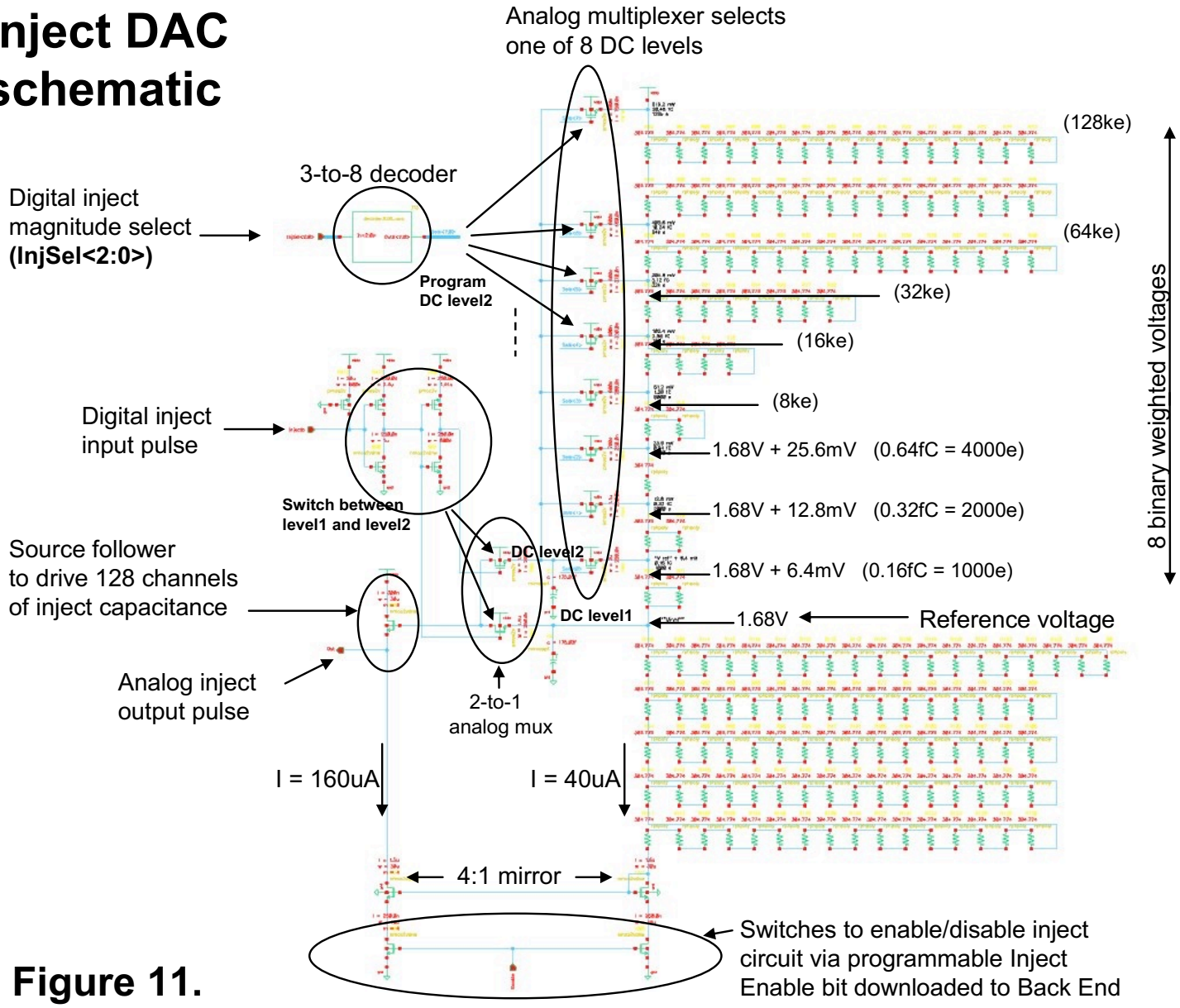


Figure 11.

The only remaining block in the front-end is the DAC that generates the shaper reference voltage (Vref) and the comparator threshold voltages (Vth0-7). The overall DAC is made of several subcircuits and is shown in Figure 12. A bandgap reference generates a reference voltage of approximately 1.34V. A resistor string divides up this voltage in such a way as to generate a programmable Vref and another 255 voltages above Vref with steps of 4 mV. These 256 voltage taps feed eight identical 256-to-1 analog multiplexers, each of which accepts an 8-bit code to program the threshold for one of the eight comparators in each channel. Figure 13 shows the bandgap schematic, which is a standard "textbook" circuit.

Vref and Vth DACs schematic

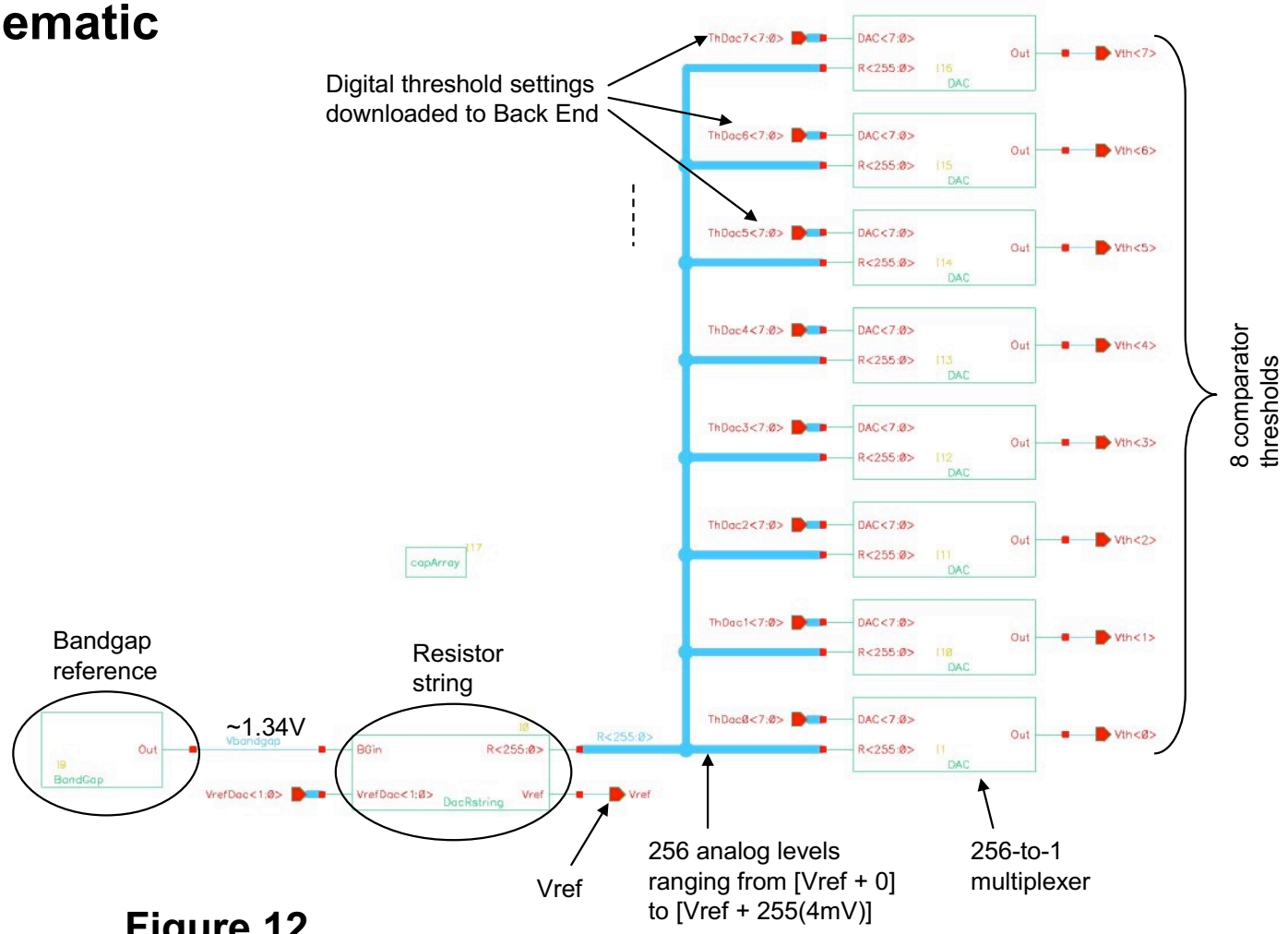


Figure 12.

Bandgap schematic

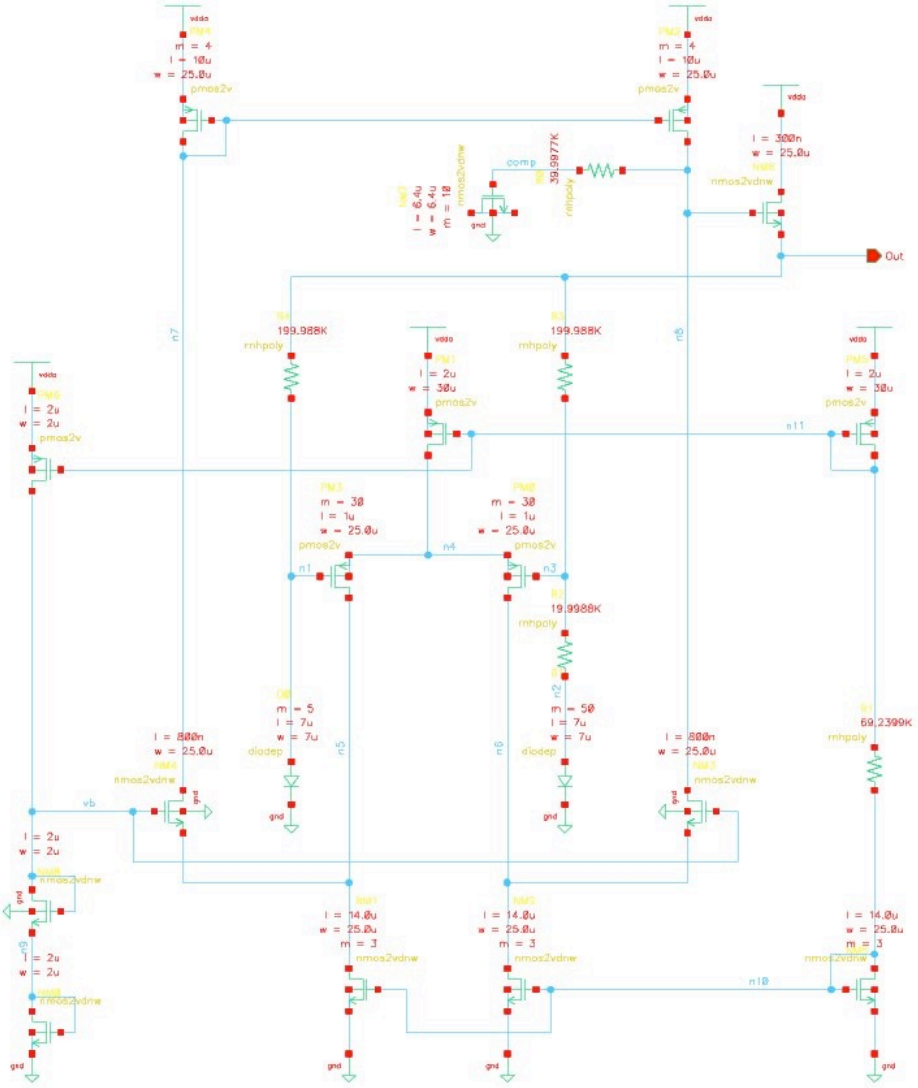


Figure 13.

The schematic of the resistor string block is shown in Figure 14. 255 resistors create 256 voltage taps of approximately 4 mV, giving a DAC range of about 1V. The lowest of these tap voltages serves as Vref. A set of resistors and switches above and below the 255 DAC resistors enables the 1V range to "slide" up or down, allowing Vref to be programmed to the lowest voltage that keeps the shaper in its linear region. This will yield the maximum shaper dynamic range. Figure 15 is the schematic of a 256-to-1 multiplexer block, which is simply an array of analog switches.

Resistor string schematic

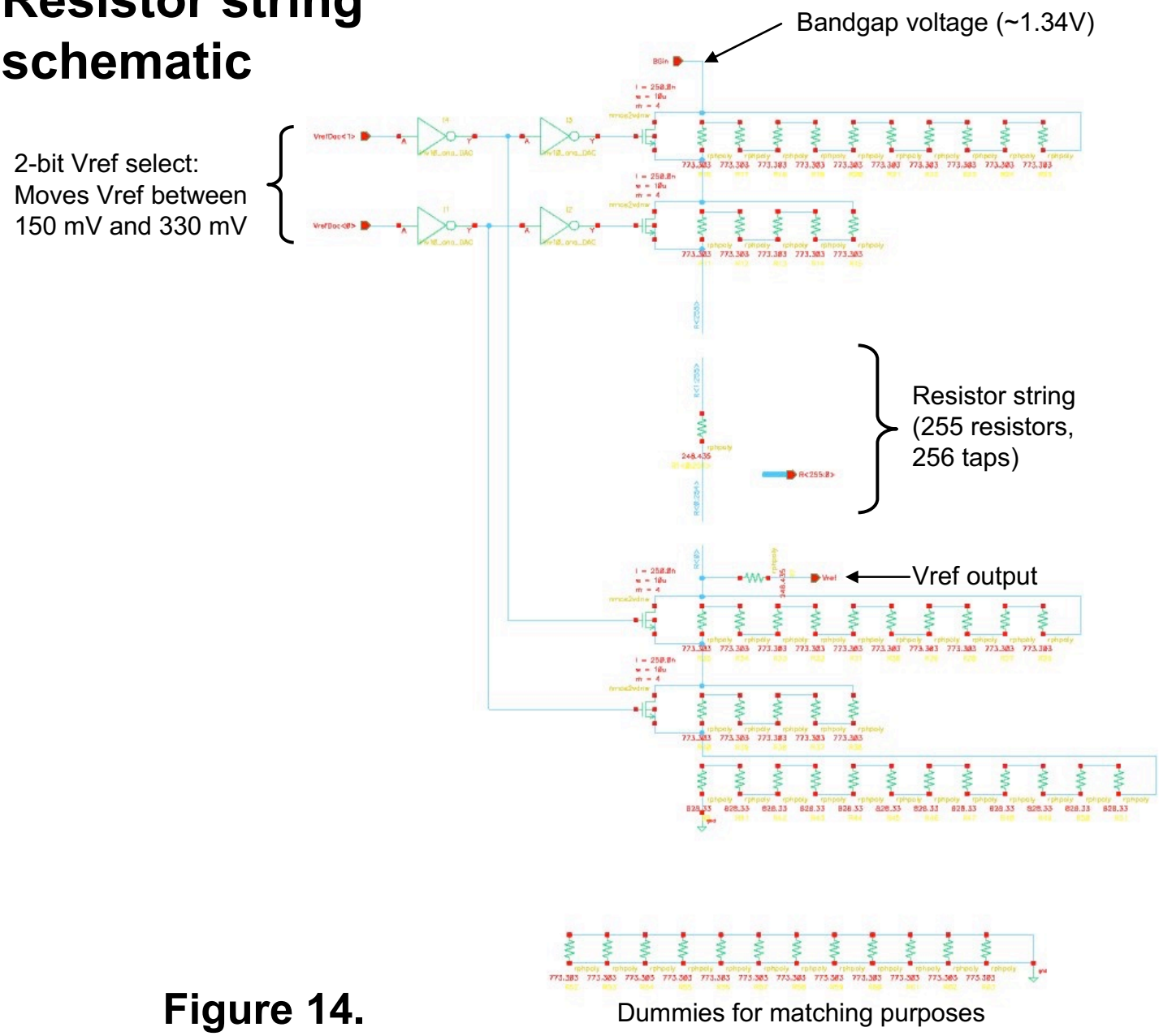


Figure 14.

DAC 256-to-1 multiplexer schematic

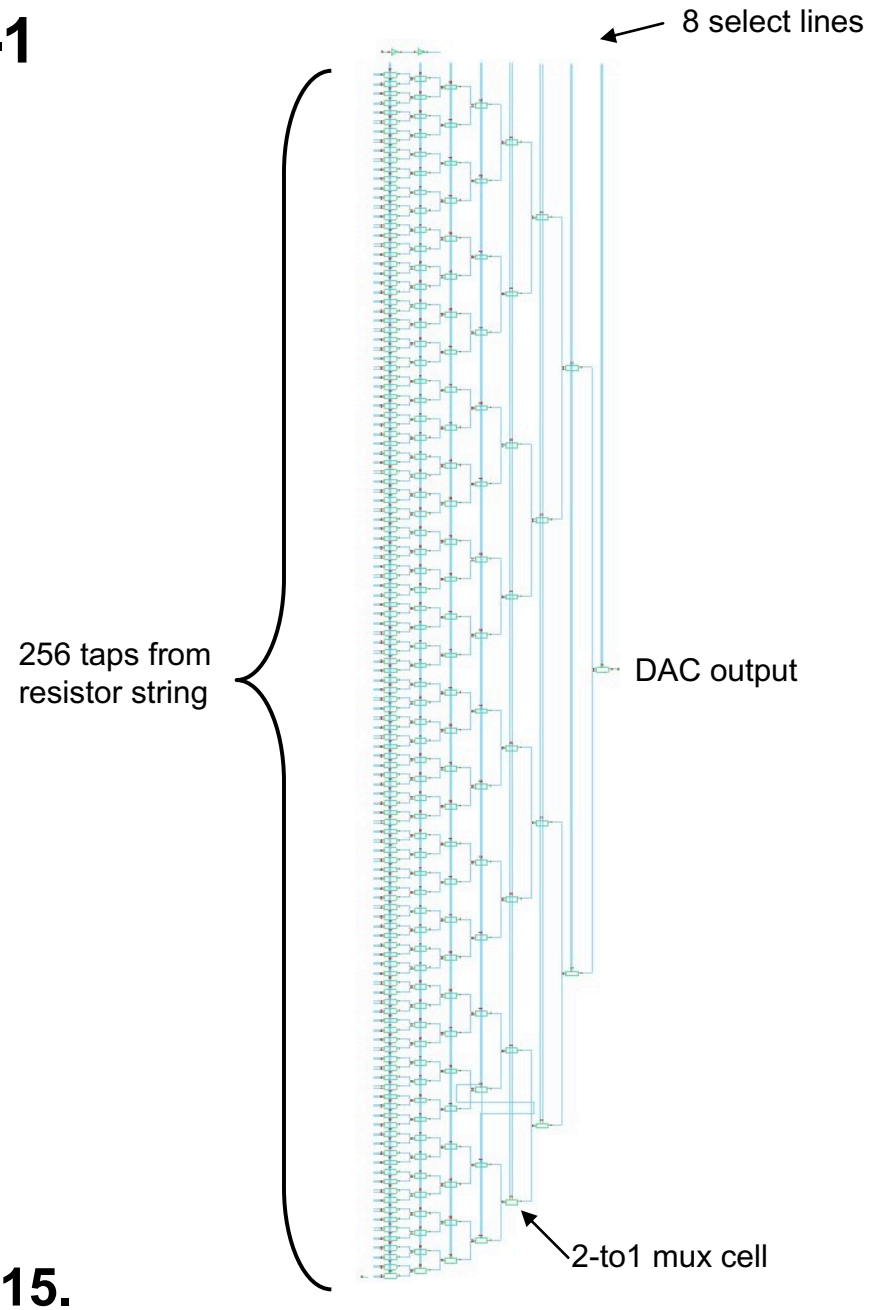


Figure 15.

Front-end Program Bits

- **P1Sel<2:0>:** sets input transistor bias current = 10uA + [(4uA)(P1Sel)]. Default = 101. Higher bias current gives lower noise.
- **Fb1Sel<3:0>:** sets integrator feedback transistor bias current Ifb = 1.67nA + [(1.67nA)(Fb1Sel)]. Default = 0100. This current defines the equivalent feedback resistance (Rfb ~ 0.05/Ifb), which determines the output pulse fall time (in conjunction with the programmable integrator feedback capacitor Cfb, yielding time constant RfbCfb).
- **N1Sel<3:0>:** Integrator source follower bias = 1.25uA + [(0.625uA)(N1Sel)]. Default = 0110.
- **LeakSel<3:0>:** maximum DC input leakage current compensation = [1.6(Ifb) + (3.33nA)(LeakSel)]. Default = 0000.
- **GSel<2:0>:** determines integrator feedback capacitance Cfb = 25fF + [(8.6fF)(GSel<0>) + (25fF)(GSel<1>) + (50fF)(GSel<2>)]. Default = 010.
- With a fixed shaper gain of about 5, the nominal system transfer gain can then be set to approximately 46, 50, 60, 67, 85, 100, 150, or 200 mV/fC.
- **BWSel<4:0>:** adds capacitance to the dominant node of the integrator. Explicit added capacitance = 10fF + [6fF(BWSel)]. Default = 00100. Used to compensate for the effects of varying input capacitance and gain setting on the integrator response.
- **N2Sel<3:0>:** Shaper bias current = 1uA + [(0.5uA)(N2Sel)]. Default = 0100. This is used to set the bandwidth of the shaper, which determines the output pulse rise time (peaking time). A shaper bias of 3.5uA results in a peaking time of approximately 60ns. Higher settings result in faster peaking times.
- **P2Sel<3:0>:** Comparator bias current = 0.5uA + [(0.25uA)(P2Sel)]. Default = 0100. Higher settings reduce comparator delay.
- **P3Sel<1:0>:** Comparator second stage pullup bias current. Default = 00. Mainly affects comparator trailing edge delay – higher settings reduce the delay.
- **InjSel<2:0>:** Digital inject circuit DAC setting. Charge inject magnitude = (1Ke)(InjSel + 1). Default = 000 (1Ke).
- **Vref<1:0>:** Shaper DC reference voltage = 150mV + [(60mV)(Vref<1:0>)]. Default = 01. For maximum dynamic range, set Vref as low as possible while still maintaining linearity.
- **Vth0<7:0>** (Threshold ADC 0)**:** 0th ADC comparator (discriminator) threshold = Vref + [(4mV)(Vth0)]. Default Vth0<7:0> = 00001000 (Vref + 32 mV).
- **Vth1<7:0>:** 1st ADC comparator threshold = Vref + [(4mV)(Vth1)]. Default Vth1<7:0> = 00010000 (Vref + 64 mV).
- **Vth2<7:0>:** 2nd ADC comparator threshold = Vref + [(4mV)(Vth2)]. Default Vth2<7:0> = 00100000 (Vref + 128 mV).
- **Vth3<7:0>:** 3rd ADC comparator threshold = Vref + [(4mV)(Vth3)]. Default Vth3<7:0> = 00110000 (Vref + 192 mV).
- **Vth4<7:0>:** 4th ADC comparator threshold = Vref + [(4mV)(Vth4)]. Default Vth4<7:0> = 01010000 (Vref + 320 mV).

- **Vth5<7:0>:** 5th ADC comparator threshold = Vref + [(4mV)(Vth5)]. Default Vth5<7:0> = 01110000 (Vref + 448 mV).
- **Vth6<7:0>:** 6th ADC comparator threshold = Vref + [(4mV)(Vth6)]. Default Vth6<7:0> = 10010000 (Vref + 576 mV).
- **Vth7<7:0>:** 7th ADC comparator threshold = Vref + [(4mV)(Vth7)]. Default Vth7<7:0> = 10110000 (Vref + 704 mV).
- **Inj_en:** Enable charge injection feature (all channels). Default = 0 (off).
- **Mask<127:0>:** Mask bits to disable charge injection on a per-channel basis. Default = 0 (all off).

Programming Strategy

- First determine **GSel** based on the desired approximate transfer gain (approx. 50 – 200 mV/fC).
- Then depending on the value of the input capacitance (Cin), program **BWSel** to achieve fast integrator response without ringing.
- For Cin = 2 pF (1.5 pF detector + 0.5 pF parasitic):
  - If **GSel** = 000, set **BWSel** = 00000
  - If **GSel** = 010, set **BWSel** = 00100
  - If **GSel** = 100, set **BWSel** = 01000
  - If **GSel** = 110, set **BWSel** = 01100
- For Cin = 1.5 pF (1 pF detector + 0.5 pF parasitic):
  - If **GSel** = 000, set **BWSel** = 00001
  - If **GSel** = 010, set **BWSel** = 00110
  - If **GSel** = 100, set **BWSel** = 01010
  - If **GSel** = 110, set **BWSel** = 01110
- For Cin = 1 pF (0.5 pF detector + 0.5 pF parasitic):
  - If **GSel** = 000, set **BWSel** = 00010
  - If **GSel** = 010, set **BWSel** = 01000
  - If **GSel** = 100, set **BWSel** = 01110
  - If **GSel** = 110, set **BWSel** = 10101

- Program **Fb1Sel** based on the desired fall time constant
- Then program **N2Sel** to achieve the desired rise time/peaking time (rise and fall times will have some effect on the transfer gain)
- Program **Vref** to the lowest value possible that allows linear response (nominally 00)
- Based on the transfer gain, set the desired comparator thresholds

Simulations

Figure 16 shows the results of running a (noiseless) simulation with all nominal parameters and a 2000e input charge. The integrator has a faster leading edge response than the shaper, but determines the trailing edge fall time. The shaper bandwidth sets the peaking time. The input pulse in this example is just large enough to trigger the discriminator. The switching of this comparator causes a slight disturbance on the shaper output, but the magnitude of this effect is well below the noise.

Simulation results

With default parameters (N2Sel = 0100, Fb1Sel = 0100, GSel = 010, BWSel = 00100). Cin = 2 pF, Qin = 2000e.

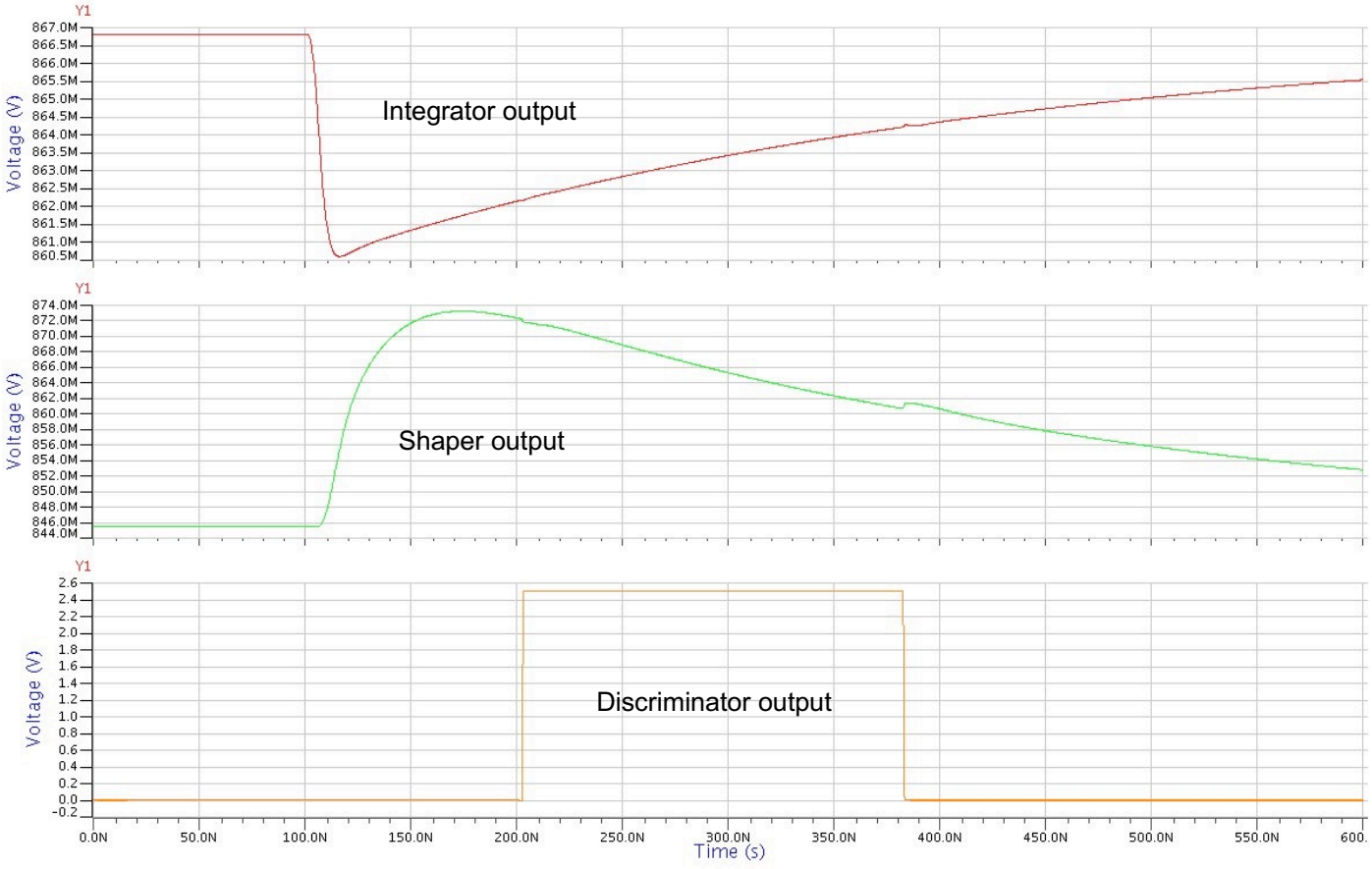


Figure 16.

Figure 17 shows the effect of varying the **BWSel** parameter to adjust the integrator response.

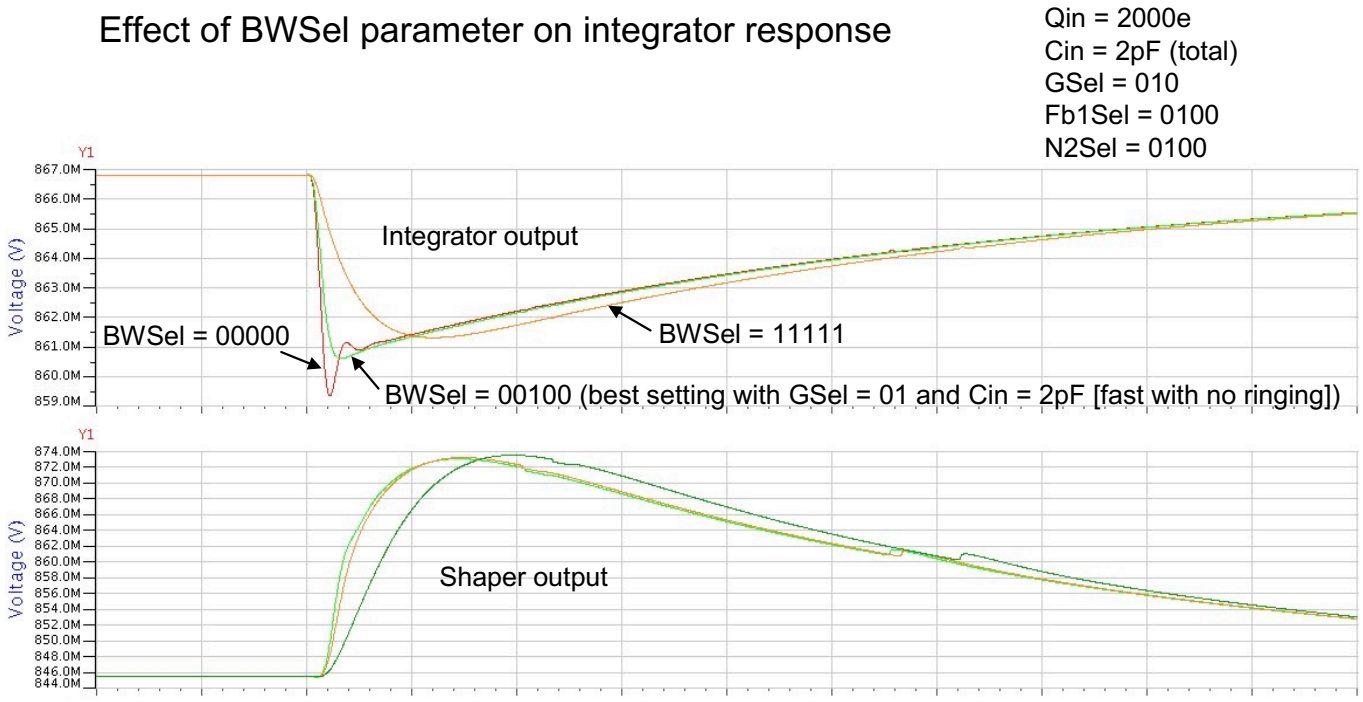


Figure 17.

The effect of adjusting **Fb1Sel** (effective integrator feedback resistance) with Cfb = 25 fF is simulated in Figure 18. In Figure 19, Cfb is changed to 50 fF, and in Figure 20 to 100 fF. Note that for a given setting of **Fb1Sel**, the RC decay time is proportional to Cfb.

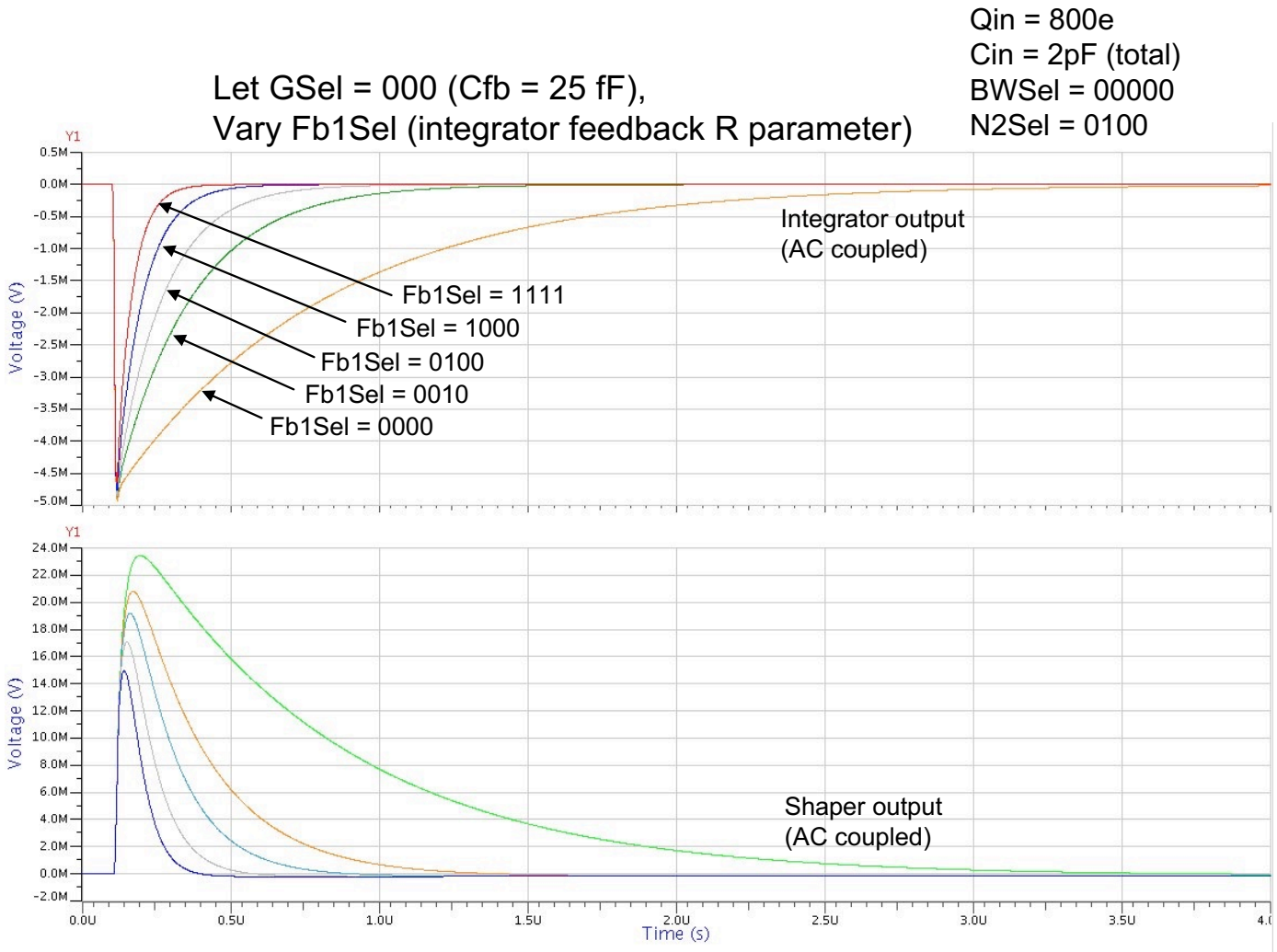


Figure 18.

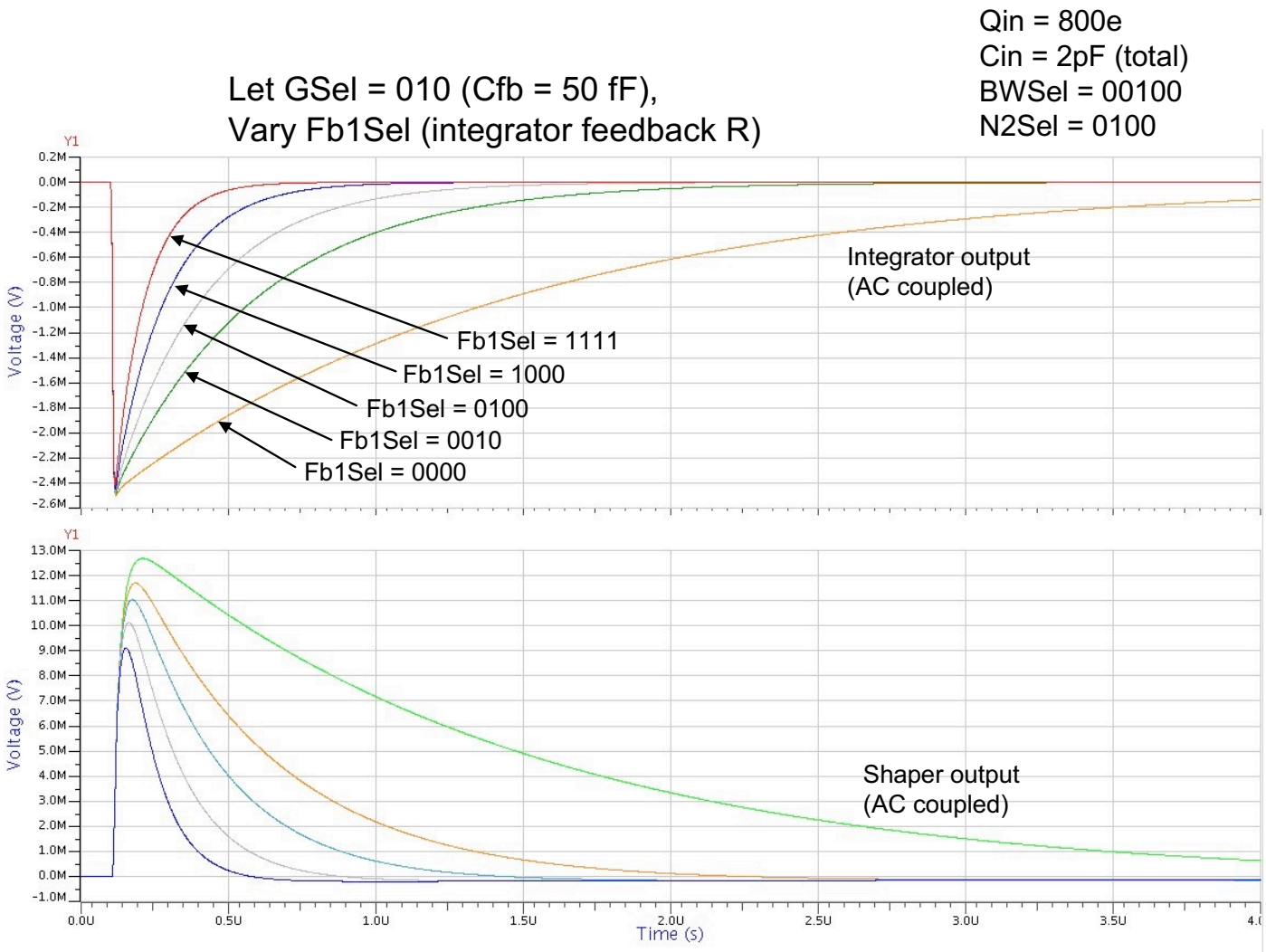

Qin = 800e
Cin = 2pF (total)
BWSel = 00100
N2Sel = 0100
Let GSel = 010 (Cfb = 50 fF),
Vary Fb1Sel (integrator feedback R)
Integrator output
(AC coupled)
Fb1Sel = 1111
Fb1Sel = 1000
Fb1Sel = 0100
Fb1Sel = 0010
Fb1Sel = 0000
Shaper output
(AC coupled)
Voltage (V)
Time (s)

Figure 19.

Qin = 800e
Cin = 2pF (total)
BWSel = 01100
N2Sel = 0100

Let GSel = 110 (Cfb = 100 fF),
Vary Fb1Sel (integrator feedback R)

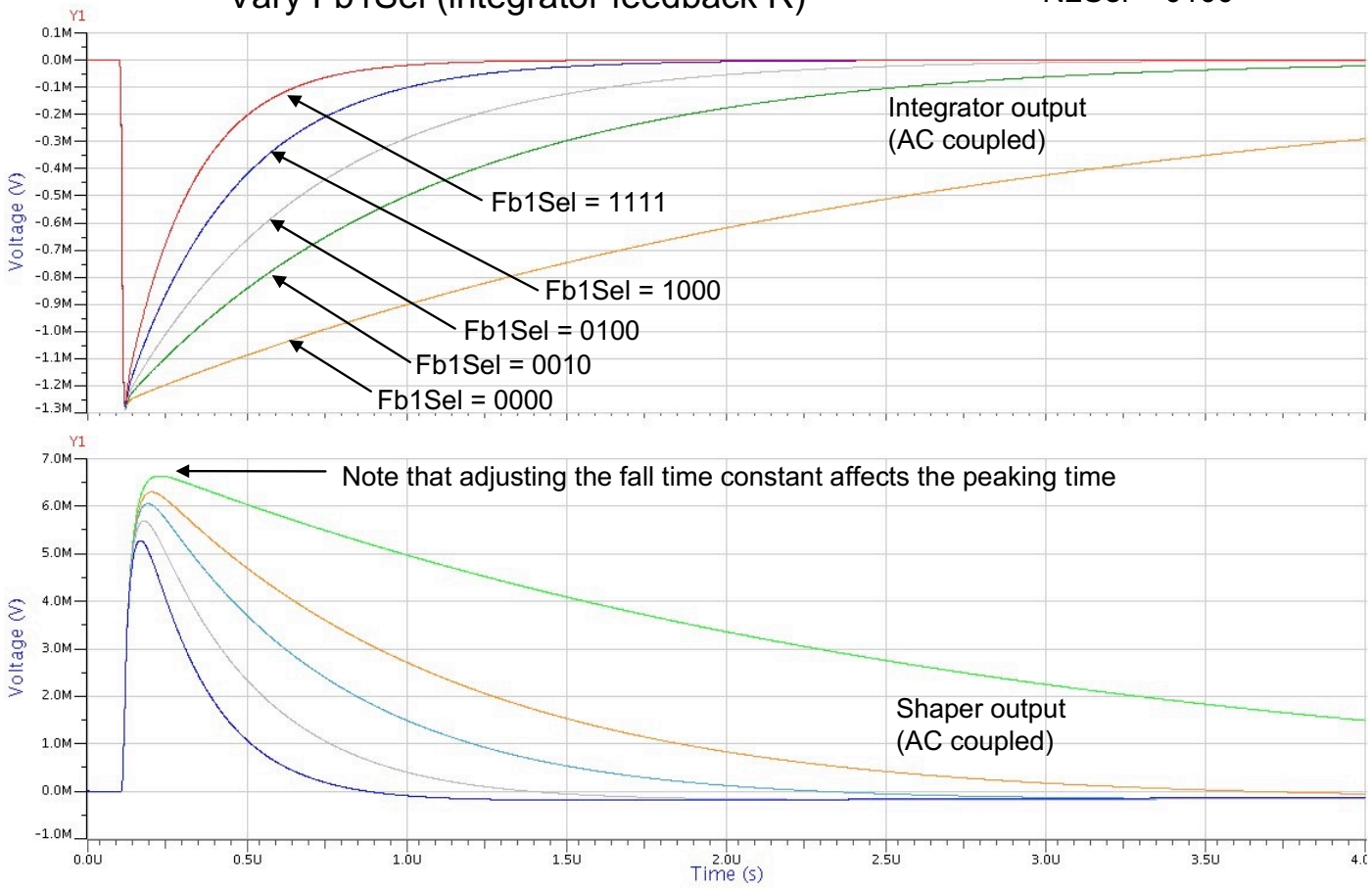


Figure 20.

Figure 21 shows the effect of adjusting **N2Sel** on the shaper peaking time (for a given integrator decay time).

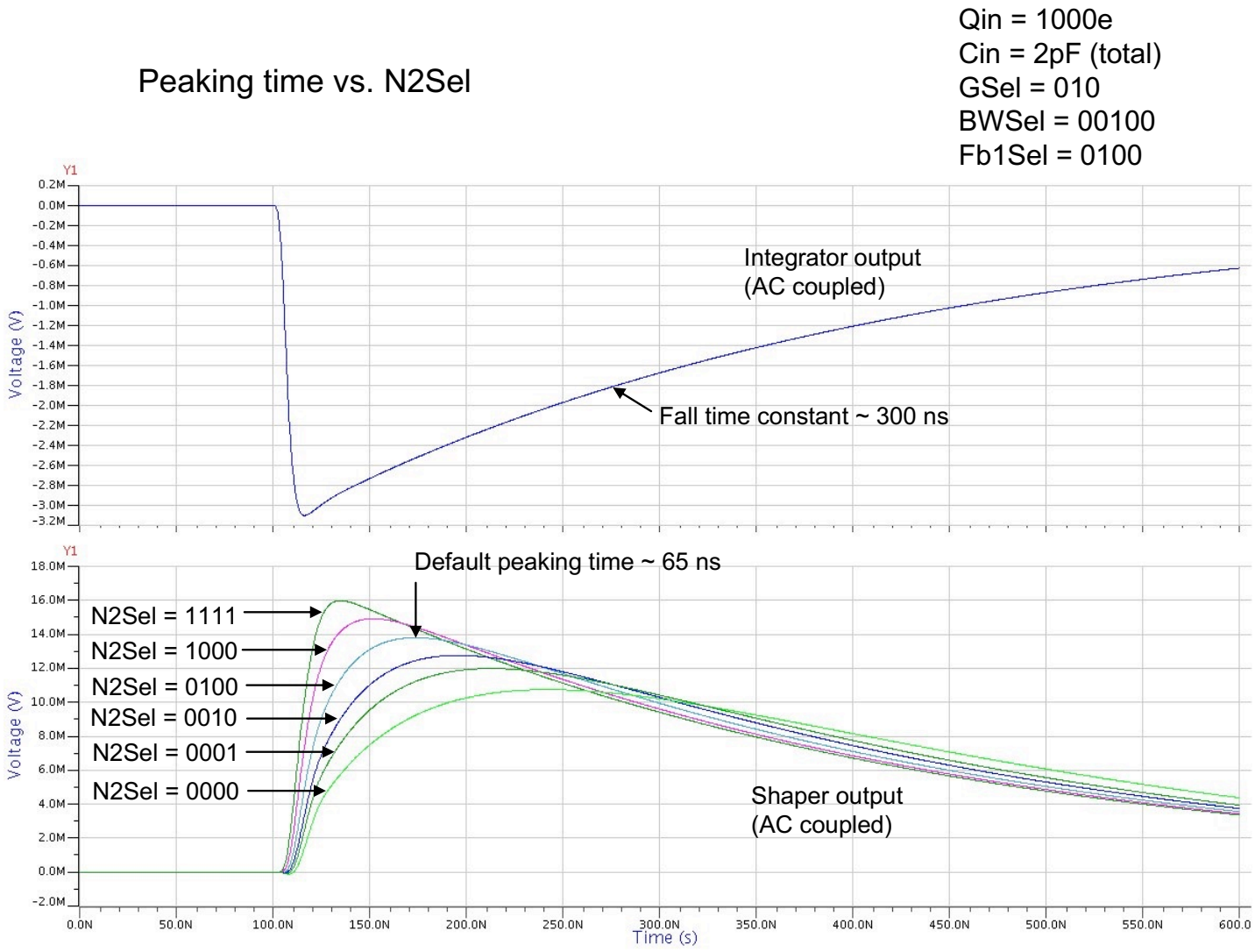
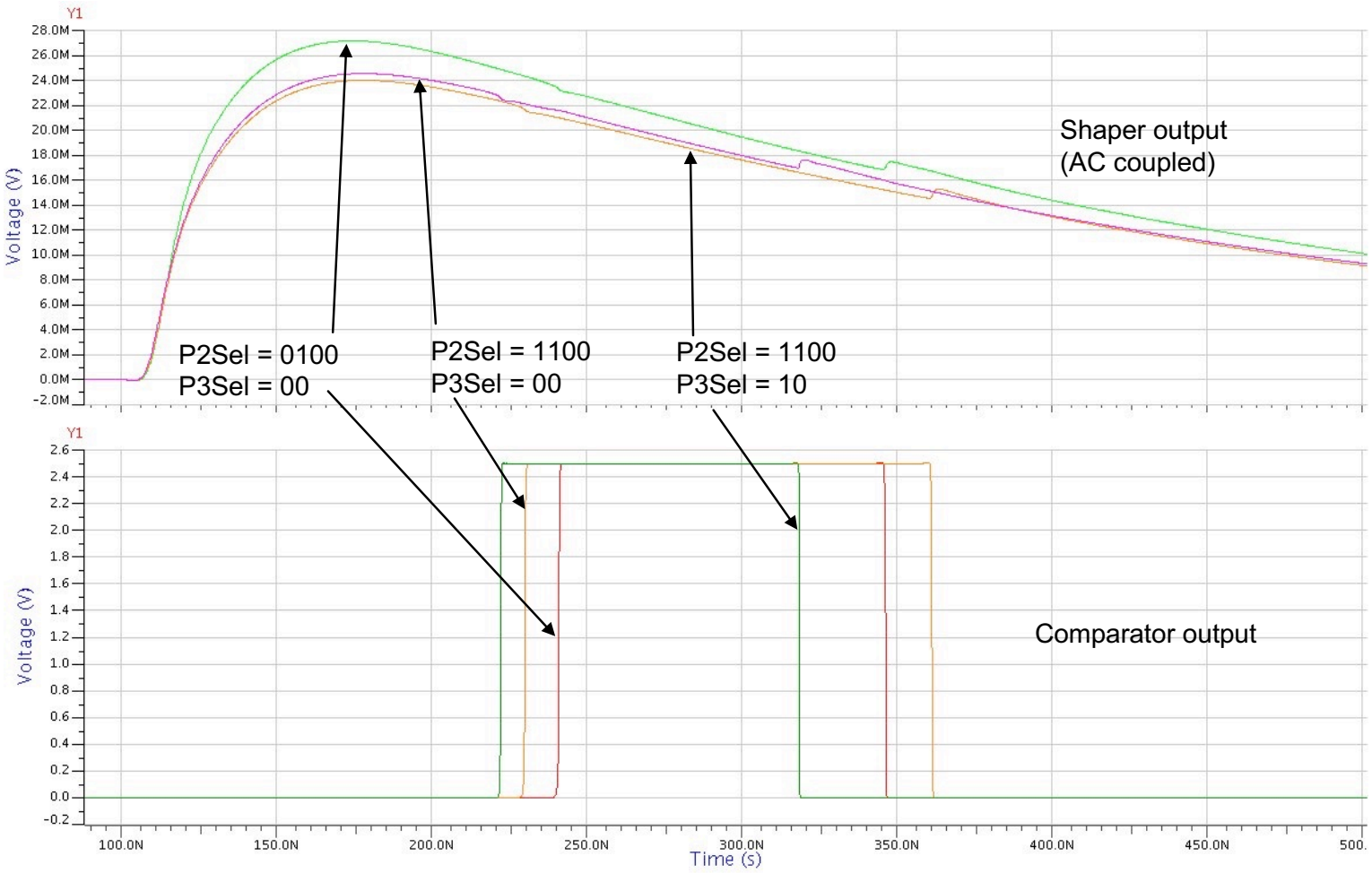


Figure 21.

Figure 22 shows how **P2Sel** and **P3Sel** can be used to change the comparator delay. Figure 23 shows that changing the comparator delay affects the minimum signal that will flip the comparator for a given threshold setting.

Comparator delay vs. P2Sel and P3Sel

In each case, set the input pulse to the smallest magnitude that gives a comparator output pulse



Note that increasing both P2Sel and P3Sel decreases the leading and trailing edge delays of the comparator.

Figure 22.

Effect of adjusting the comparator bias

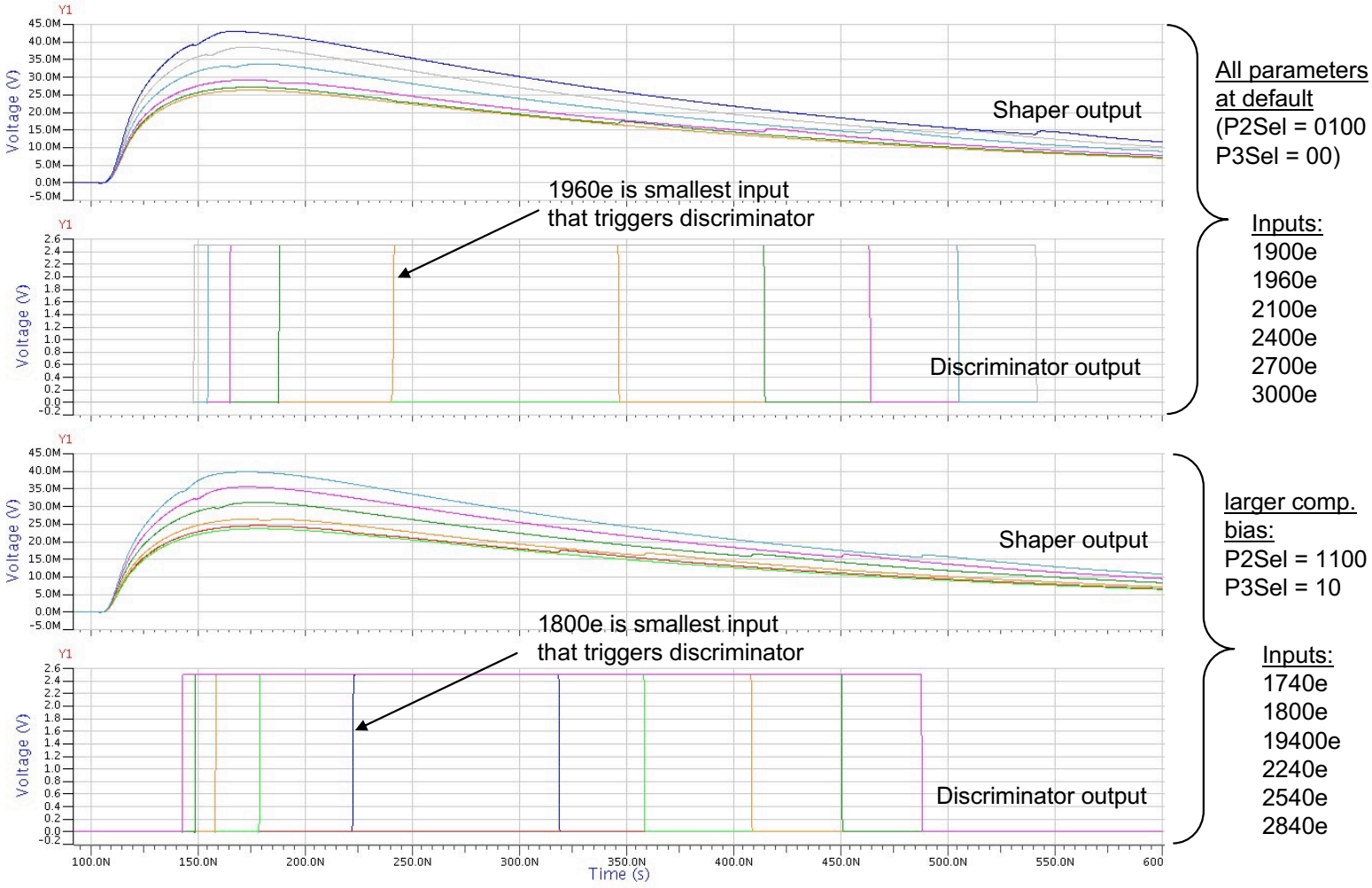


Figure 23.

The difference between a “noiseless” and a “noisy” simulation is illustrated in Figure 24. With all parameters at default, the input charge is adjusted to be the minimum that results in triggering the comparator in the noiseless simulation. Then with the same conditions, 25 runs with transient noise included are performed. Note that approximately half of these runs trigger the comparator and half do not. Due to noise, there is a significant variation in the time that the comparator is triggered. This is the worst case situation – this variation will be reduced for higher amplitude pulses.

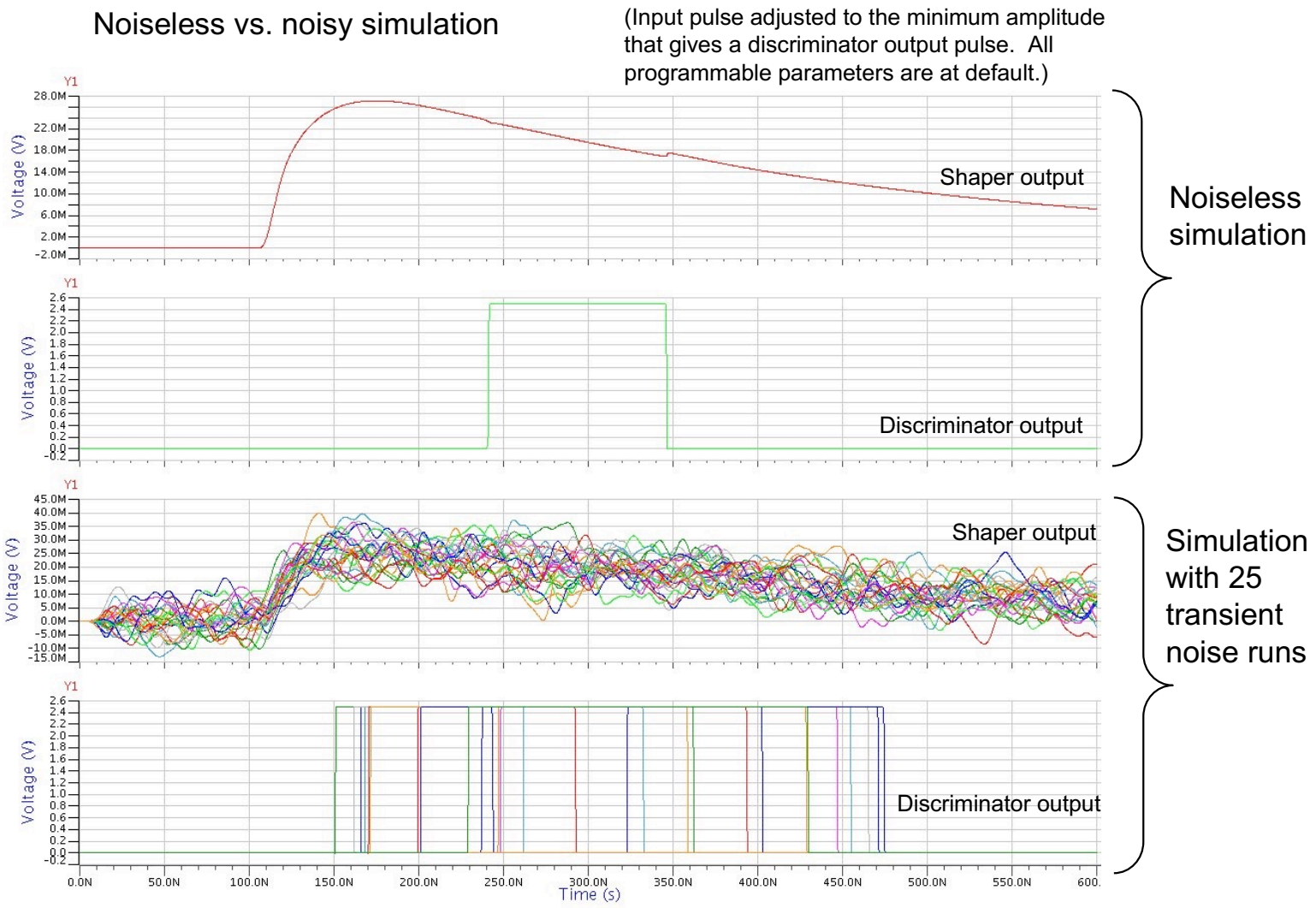
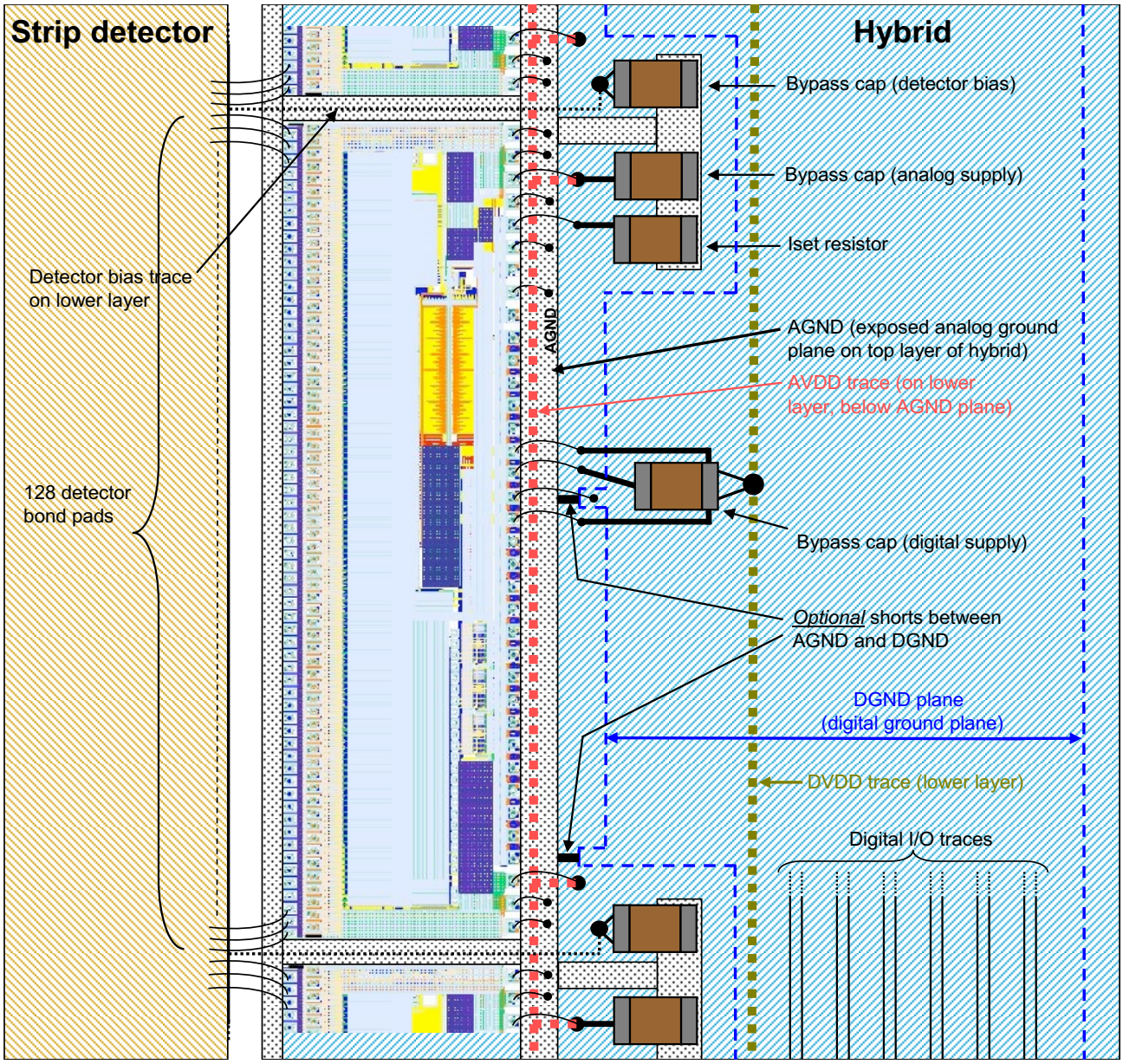


Figure 24.

Figure 25 is a drawing suggesting the layout approach for the High Density Interconnect (HDI) hybrid, which holds multiple FPHX chips. Bare chips will be mounted directly on a top-level analog ground plane, which extends underneath all the chips. All analog ground pins on the chip can then be directly down-bonded to this plane. A detector bias bypass cap and an analog supply bypass cap should be mounted close to the chip and returned directly to the analog ground plane. A digital ground plane should be formed covering the rest of the hybrid, probably on a lower layer. All digital I/O signals can then run on the top layer directly over the digital ground plane. Provision should be made to optionally short the digital ground plane to the analog ground plane at least in one place per chip, and probably two. These two planes should definitely be joined in one place at the edge of the hybrid where the analog and digital power regulators are located. There should be a digital power bypass cap per chip, which is connected in a somewhat non-standard fashion. The two digital power pins connect to the far terminal of the capacitor, with one trace on each side. The near terminal of the capacitor should then connect not to the digital ground plane, but only to one digital ground pin of the chip. The other digital ground pin of the chip should then be connected to the digital ground plane. This should help to keep digital bypass transient currents on the chip itself, and limit circulating currents in the plane.

Figure 25.

FPhx hybrid layout strategy



Front-end related chip pins

- **Analog Ground**
  - ground return dedicated to input transistor source only. Bond directly to hybrid ground plane.
- **Analog Ground 2**
  - ground for integrators, shapers, and analog portion of comparators. Bond directly to hybrid ground plane.
- **Analog Power**
  - 2.5V analog supply for integrators, shapers, and analog portion of comparators. Bypass directly to ground.
- **Substrate**
  - separate net for substrate connection only. Bond directly to hybrid ground plane.
- **Iset**
  - Master current reference to set Front-end bias levels. Nominally set to 30 uA (40K resistor to ground).
- **injInputAnalog**
  - A positive going transition from an analog level V1 to level V2 will inject a charge of (V2 – V1)(25 fF) into all channels that are enabled for charge injection. Level V1 should be close or equal to ground.
- **injInputDigital**
  - a negative going digital transition on this input will inject a programmed charge into all channels that are enabled for charge injection. The magnitude of the charge is set with a DAC.
- **Reset, Resetb**
  - This is a Back-end LVDS input, whose internal CMOS version is also used by the Front-end to reset all the comparators.
- **Digital Ground and Digital Power**
  - These are Back-end supply pads which are also used by the Front-end to power the digital portion of the comparators

Bondable Test Pads

A series of bondable test pads in the interior of the chip are provided on the prototype chip, in order to be able to view buffered versions of certain analog and digital signals. Some or none of these may be included on the production version of the chip. See Figure 26 for the location and function of these pads.

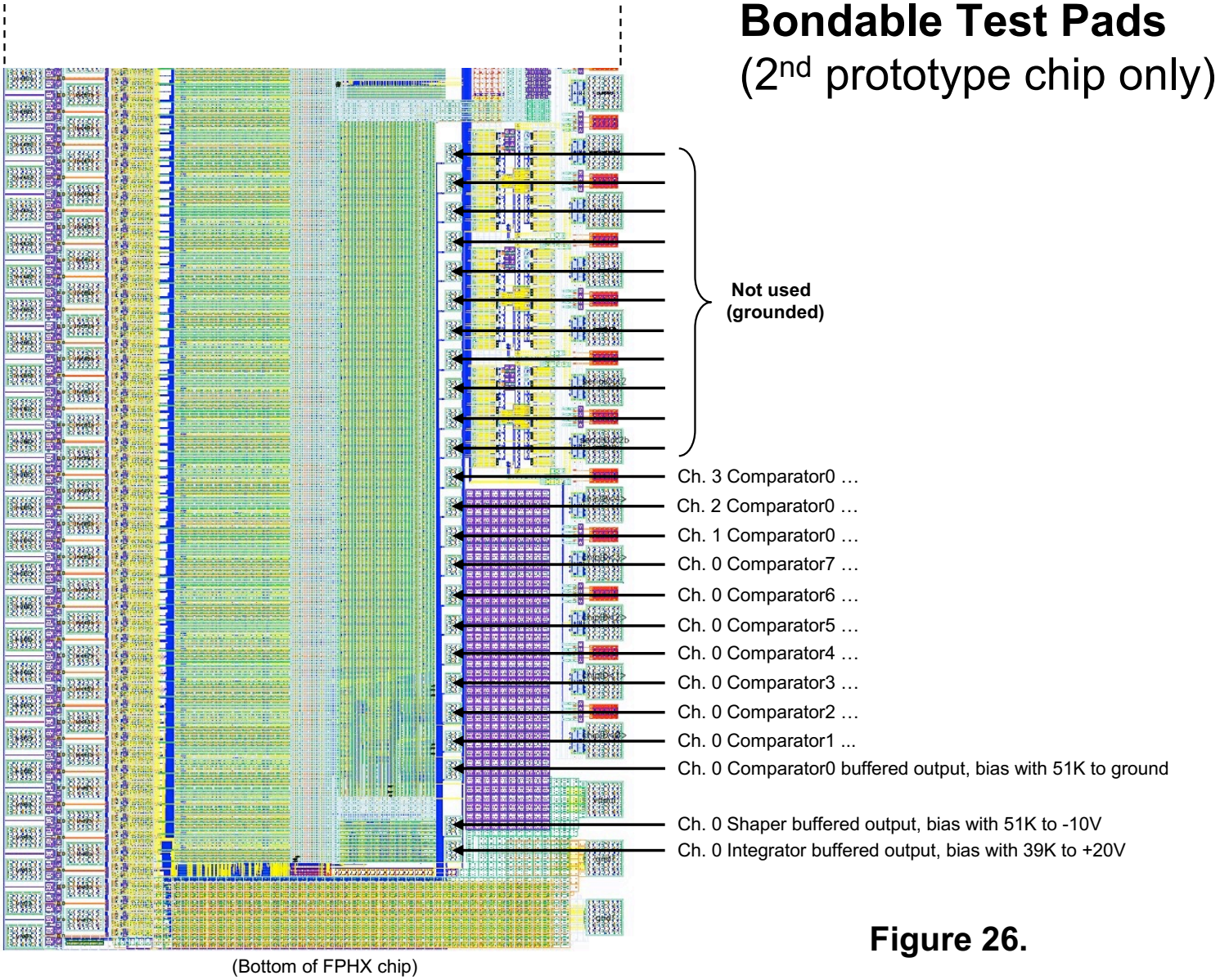


Figure 26.

The FPHX back-end

Chip Organization

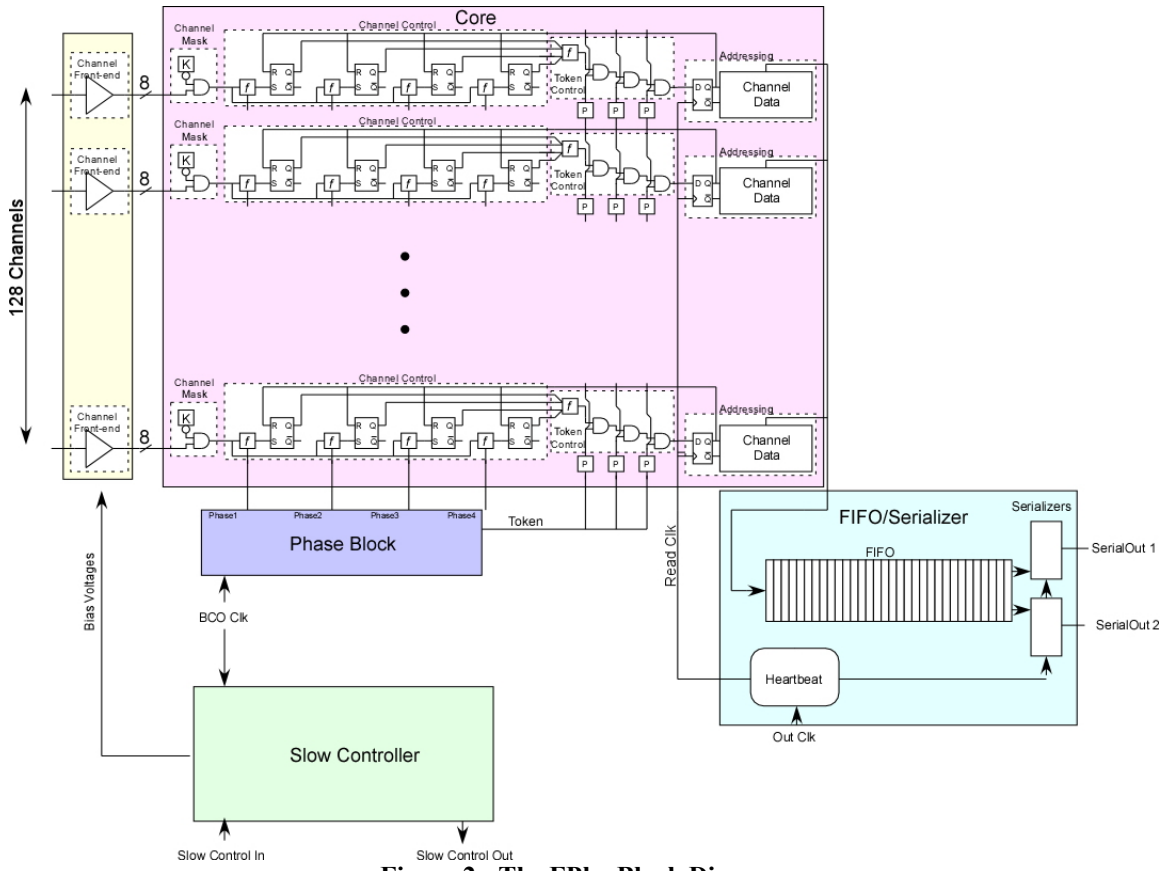


Figure 2 - The FPhx Block Diagram

The back-end block diagram is shown in Figure 2. Charge signals enter the analog front end (yellow in Figure 2) through 128 independent inputs. In the analog front end, each strip's signal is amplified and discriminated against threshold voltages that are provided with other bias voltages by the Slow Controller (green in Figure 2). Each Channel Front End provides eight bits of digital information (one bit of hit discriminator and seven bits of thermometer-coded magnitude information) to the Core (pink in Figure 2). If the channel is masked in the Core (K=1), then the front end signals are blocked by the Channel Mask and nothing more happens. If the channel is unmasked (K=0) then the signals are passed unchanged to the Channel Control circuitry. The Channel Control circuitry breaks the signals up by phase depending on the state of the Phase Block (purple in Figure 2). Tokens are then passed by phase into the Token Control circuitry which selects one channel at a time to output its data (address and ADC value) into the FIFO/Serializer (blue in Figure 2). The Read Clock for the Channel Data output to the

FIFO is provided by a Heartbeat generator inside the FIFO/Serializer. Once in the FIFO/Serializer the data is serialized and output on one of two serial lines. If desired all data can be output on Serial Out 1. However, this slows processing down. Figure 3 shows the FPhx Layout (without metals 3, 4, and 5)

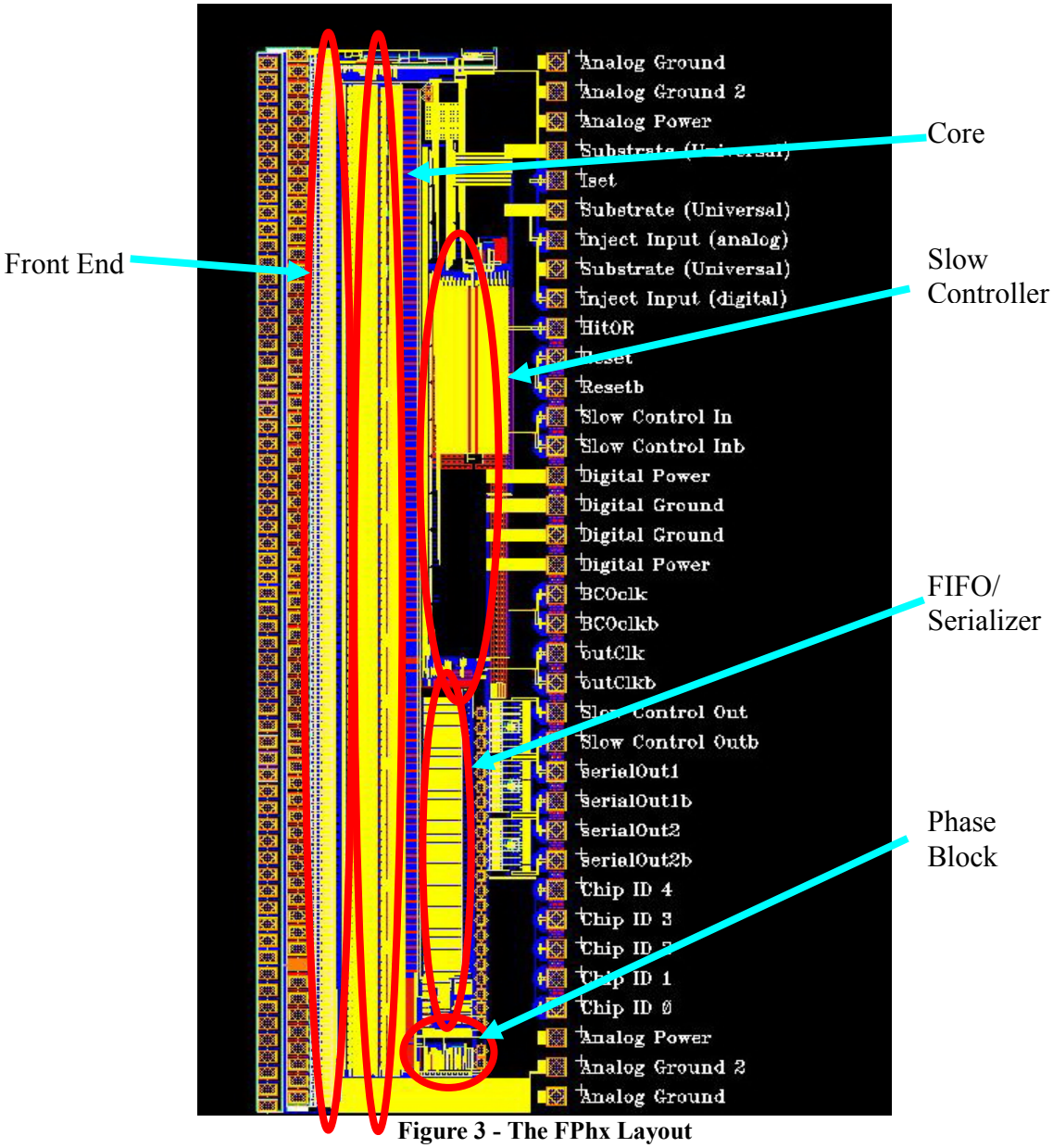


Figure 3 - The FPhx Layout

Figure 4 shows how the chip breaks down logically inside the Cadence Database.

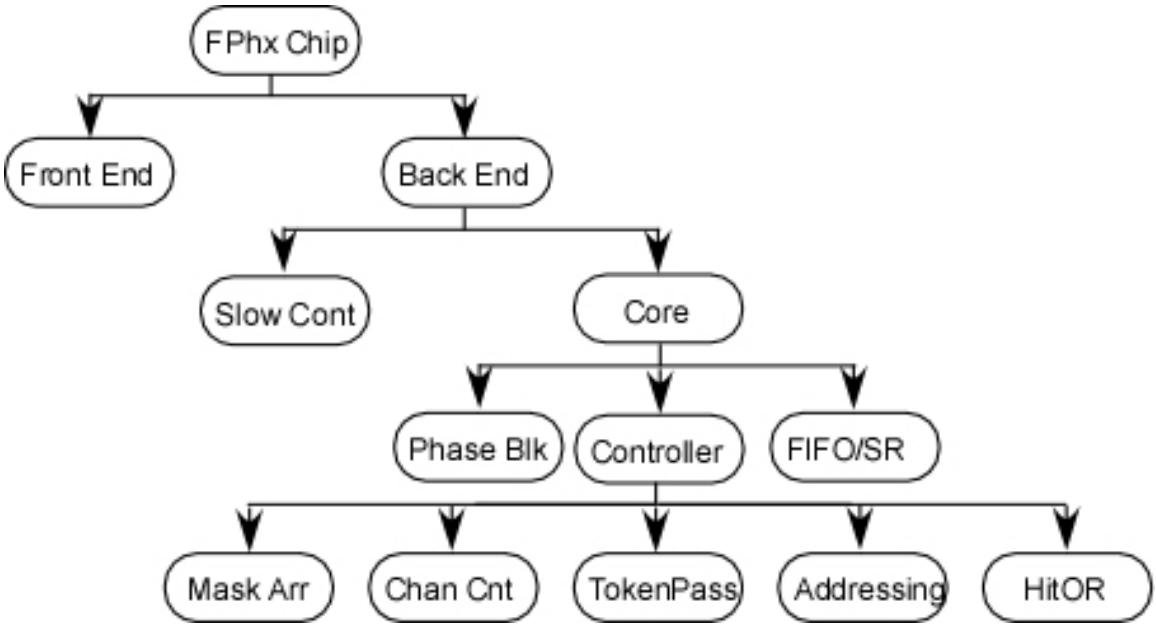


Figure 4 - A Rough Breakdown of the FPhx Sub-circuits

Serial Words

Each of the FPhx's two serial output words will drive a continuous stream of 20-bit words. Each word will be 1 sync bit followed by 19 data bits. If all 19 data bits are zeros, then that 20-bit word is a Sync Word placed in the data stream to enable the data acquisition hardware to align itself to the data flow from the FPhx chip. If the 19 data bits are NOT all zeroes then that 20-bit word is a Data Word and it contains information on a hit strip. The data is organized as follows:

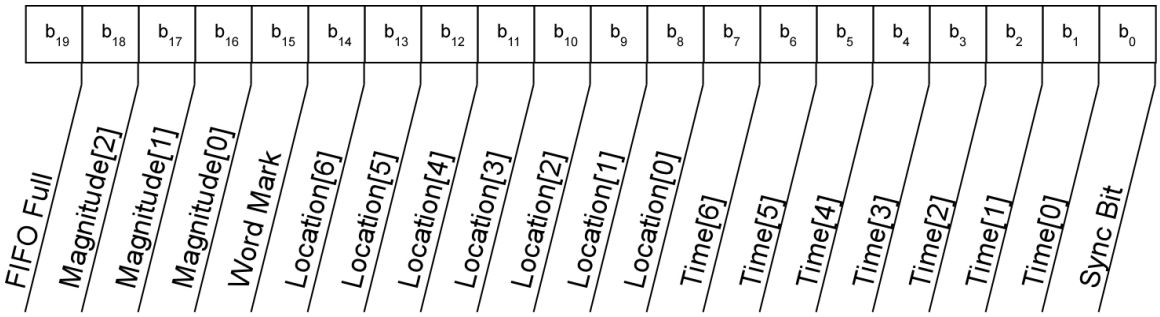


Figure 5 - The Serial Output Data Word Structure

Note that in time order, the Sync bit comes out first and the FIFO full bit comes out last. Some examples follow.

Output Word	Location	Time	Mag.	Comments
00000000000000000001	X	X	X	Sync Word

00011000000100000011	1	1	1	FIFO not full
00111110000001010011	96	41	3	FIFO not full
10001000000000000001	0	0	0	FIFO full

Back End

Slow Controller

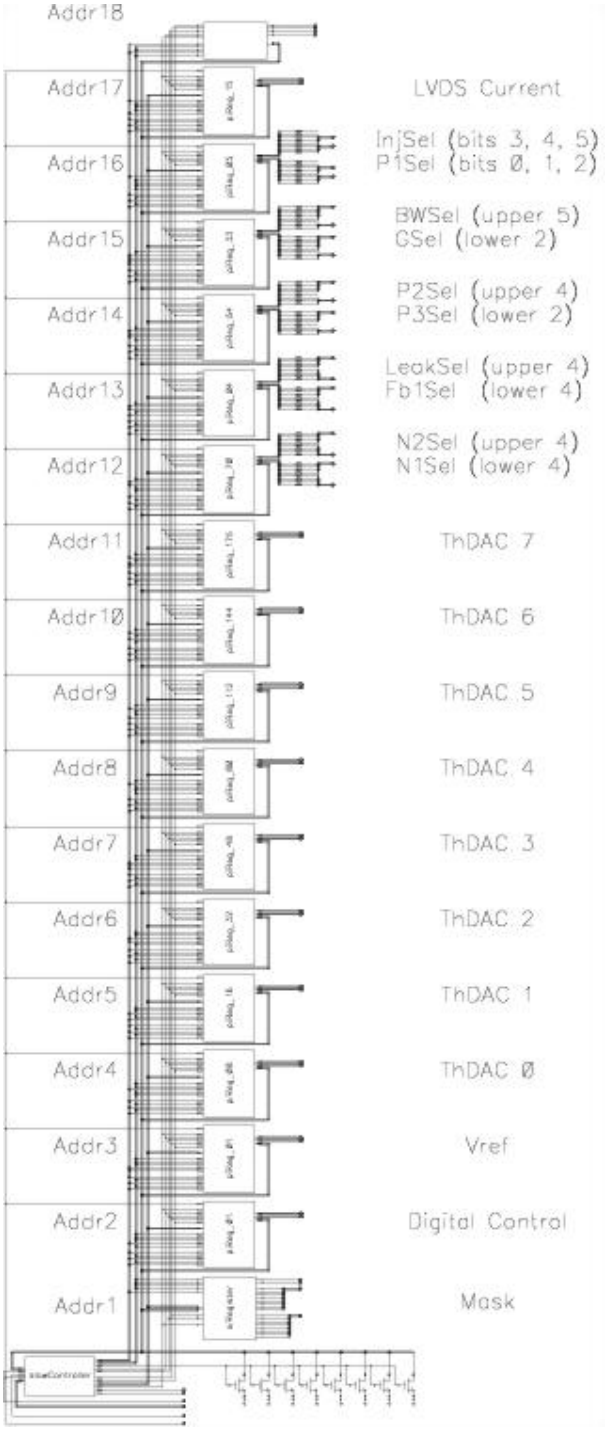


Figure 6 - Phenix Slow Controller Schematic

The Slow Controller provides a serial interface between the user and the chip. Most notably, this allows the user to adjust the bias values for the front end. The Phenix Slow Controller uses a so-called "three-wire" interface, meaning that communication is

accomplished through three lines. – Slow Control In, Slow Control Out, and Slow Control Clock.

Slow Control In is an input to the Slow Controller that will communicate the Slow Control Words from the user to the chip. The Slow Control In line can be shared among several chips or connected point-to-point.

Slow Control Out is an output line from the chip that will provide the user with read-back information from the chip. The FPhx Chip assumes that the Slow Control Out line is bussed with several other chips. Therefore, the Slow Control Out line is tri-stated unless there is data to be output from the chip.

Slow Control Clock is an input to the chip that provides a clock for data processing. Slow Control In data is assumed to change on the rising edge of the Slow Control Clock and will be latched on the falling edge of the Slow Control Clock. Slow Control Output will be changed on the rising edge of the Slow Control Clock. In the FPhx chip, the BCO clock is used as the Slow Control Clock to reduce pad count.

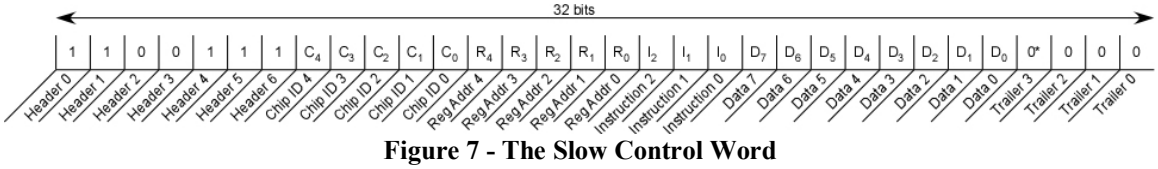


Figure 7 - The Slow Control Word

The Slow Control Word is a 32-bit word that is input to the Slow Controller through the Slow Control In line. It is shown in Figure 7. [In time, the word is shifted from left to right.] The Slow Control Word consists of a 7-bit header (always 1100111), a 5-bit Chip ID, a 5-bit Register Address, a 3-bit Instruction, an 8-bit Data Word, and a 4-bit Trailer (always 0000).

In order for action to occur in a particular chip,

1. The first seven bits of the word must be 1100111
2. The 5-bit Chip ID in the Slow Control Word must match either
    a. The Chip ID as set by external pads on the FPhx Chip (see エラー! 参照元が見つかりません。) or
    b. The Wild Chip ID (10101) which will cause action simultaneously on all chips connected to the same Slow Control In line.
3. The 5-bit Register Address in the Slow Control Word must match either
    a. A particular register's address in the register array.
    b. The Wild Register Address (10101) which will cause action simultaneously in (nearly) all registers.
4. Any 3-bit Instruction will cause the current contents of the addressed register to be output on the Shift Control Out line provided that:
    a. The address is not a Wild Register Address
    b. The address is a valid register address
    c. The Chip ID is not a Wild Chip ID
    d. The Chip ID is the exact address of this chip as set by external pads on the FPhx Chip (see エラー! 参照元が見つかりません。)
5. A valid instruction (Set, Reset, Default, Write) will cause the appropriate action in the addressed register(s) in addition to outputting the current

contents of the addressed register(s). [If a Wild Register Address is used, the chip will output all zeros.]

6. Any eight bits in the Data Bits are acceptable. Whether or not they are appropriate is left entirely to the user.
7. The last four bits must be 0000.

The instructions are as follows:

Table 1 - Slow Controller Instruction Set

I_2	I_1	I_0	Instruction	Effect
0	0	0	No effect	
0	0	1	**Write**	The eight data bits (D_7-D_0) are written into the register
0	1	0	**Set**	All the register bits are set to 1 [D_7-D_0 are ignored]
0	1	1	No effect	
1	0	0	No effect	
1	0	1	**Reset**	All the register bits are reset to 0 [D_7-D_0 are ignored]
1	1	0	**Default**	The register bits are sent to a preset, hardwired value [D_7-D_0 are ignored]
1	1	1	No effect	

The registers, their addresses and defaults are shown in .

Table 2 - Register Addresses and Defaults

Address	Name	Bits	Default	Notes
1	Mask	N/A	N/A	Set Command = Mask Channel Reset Command = Unmask Channel Data bits D6-D0 = Channel Address Data bit D7 = Global Command (i.e. Mask all channels or Unmask all channels)
2	Digital Control	7:0	1	Bit 0 = Active Serial Lines (1=Two, 0=One) Bit 1 = Accept (1=Accept Hits, 0=Reject) Bit 2 = Global Inject Enable Bit 3 = Serial Output Order
3	Vref	1:0	1	
4	Threshold DAC 0	7:0	8	
5	Threshold DAC 1	7:0	16	
6	Threshold DAC 2	7:0	32	
7	Threshold DAC 3	7:0	48	
8	Threshold DAC 4	7:0	80	
9	Threshold	7:0	112	

10	DAC 5 Threshold	7:0	144	
11	DAC 6 Threshold	7:0	176	
	DAC 7			
12	N1Sel	3:0	6	
12	N2Sel	7:4	4	
13	FB1Sel	3:0	4	
13	LeakSel	7:4	0	
14	P3Sel	1:0	0	
14	P2Sel	7:4	4	
15	GSel	2:0	2	Gain Select
15	BWSel	7:3	4	Bandwidth Select
16	P1Sel	2:0	5	
16	InjSel	5:3	0	
17	LVDS Current	7:0	16	1mA LVDS drive current for each active bit
18	Programmable Resets	N/A	N/A	Reset 1 - Set Command – core Reset Reset 2 - Reset Command – time Reset Reset 3 – Default Command – unused Reset 4 – Write Command - unused

Two addresses of note are Address 18, the Programmable Resets, and Address 1, the Mask Register. Neither "register" actually stores any value. Rather, they process Slow Control Words for a particular effect.

The Programmable Reset register provides resets to the FPhx Chip in response to a particular command. If a Set Command is given to address 18, then Reset 1 will be activated during the Trailer 3 bit period (see Figure 7). Reset 1 is a Core Reset, meaning that all hits contained in the FPhx Core will be reset. However, the BCO time stamp located in the Phase Block will NOT be affected. If a Reset Command is given to address 18, then Reset 2 will be activated during the Trailer 3 bit period (see Figure 7). Reset 2 is the Time Reset which will reset the BCO time stamp located in the Phase Block, but will not erase any hits currently residing in the Core. Resets 3 and 4 are activated as the result of a Default and Write command (respectively). As of the first prototype, these Resets are unused in the FPhx Chip.

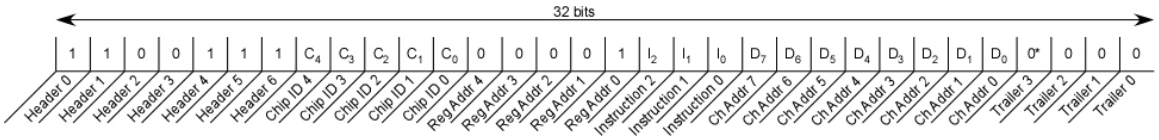


Figure 8 - The Slow Controller Word when using the Mask Register

The Mask Register (Address 1) serves as an interface between the user and the Mask Array (see Figure 4) which blocks transmission of channel activity from back-end processing. As shown in Figure 8, when using the Mask Register, the eight data bits become eight Channel Address bits. The lower 7 bits are the actual channel ID (from 0 to 127). The most significant bit is a global activator bit. If it is a zero, then the command

is intended for the single channel addressed in the lower 7 bits. If it is a one, then the command is intended for all channels simultaneously. If the command is a set, then the channel will be masked off. If the command is a reset, then the channel will be released for data processing.

The following figures show Slow Controller operation:

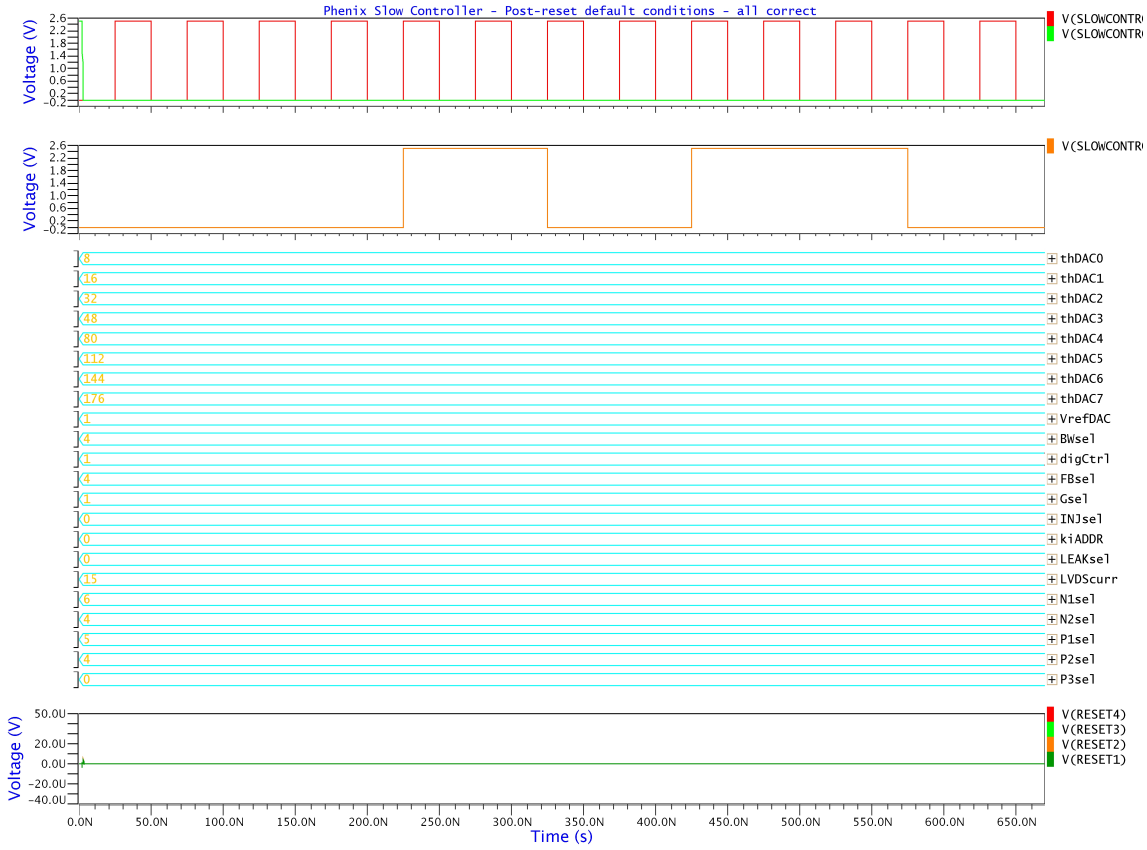


Figure 9 - Slow Controller Register States after a Reset

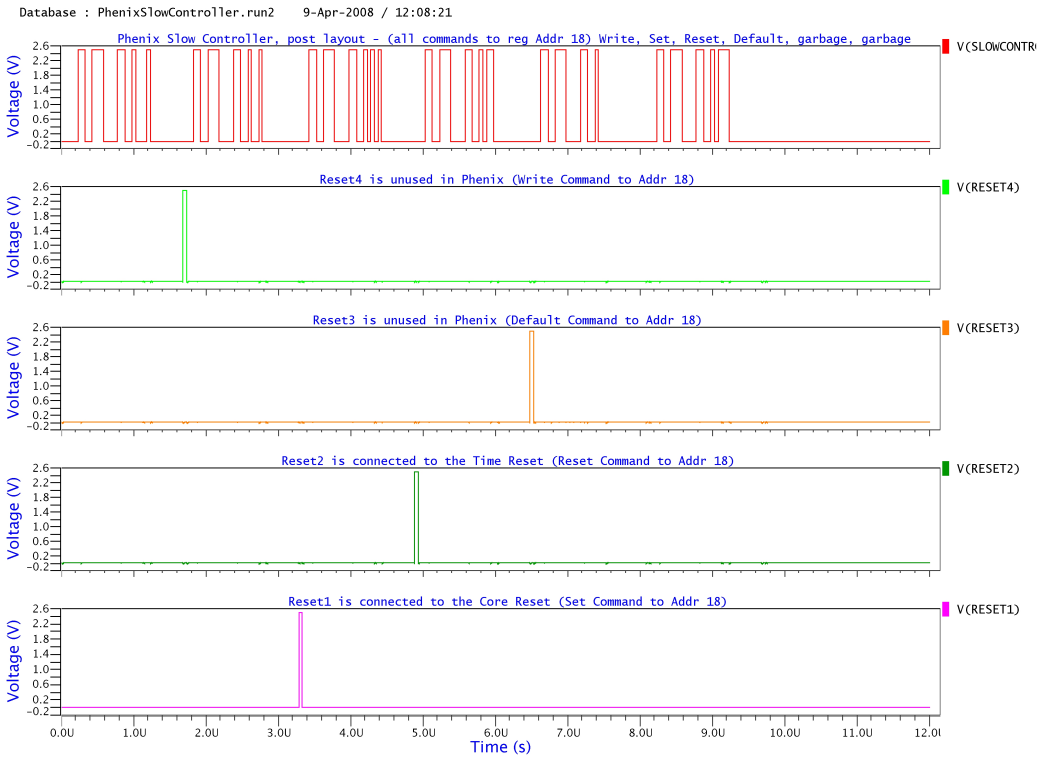


Figure 10 - Slow Controller Resets as a result of a series of commands

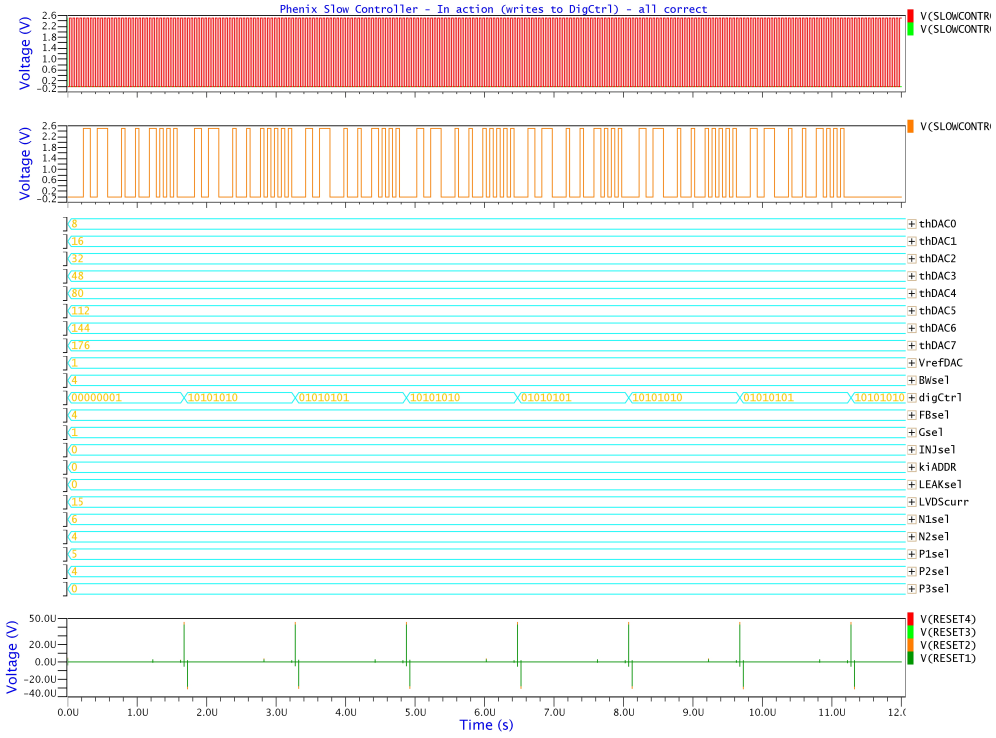


Figure 11 - Slow Controller activity on the Digital Control Register

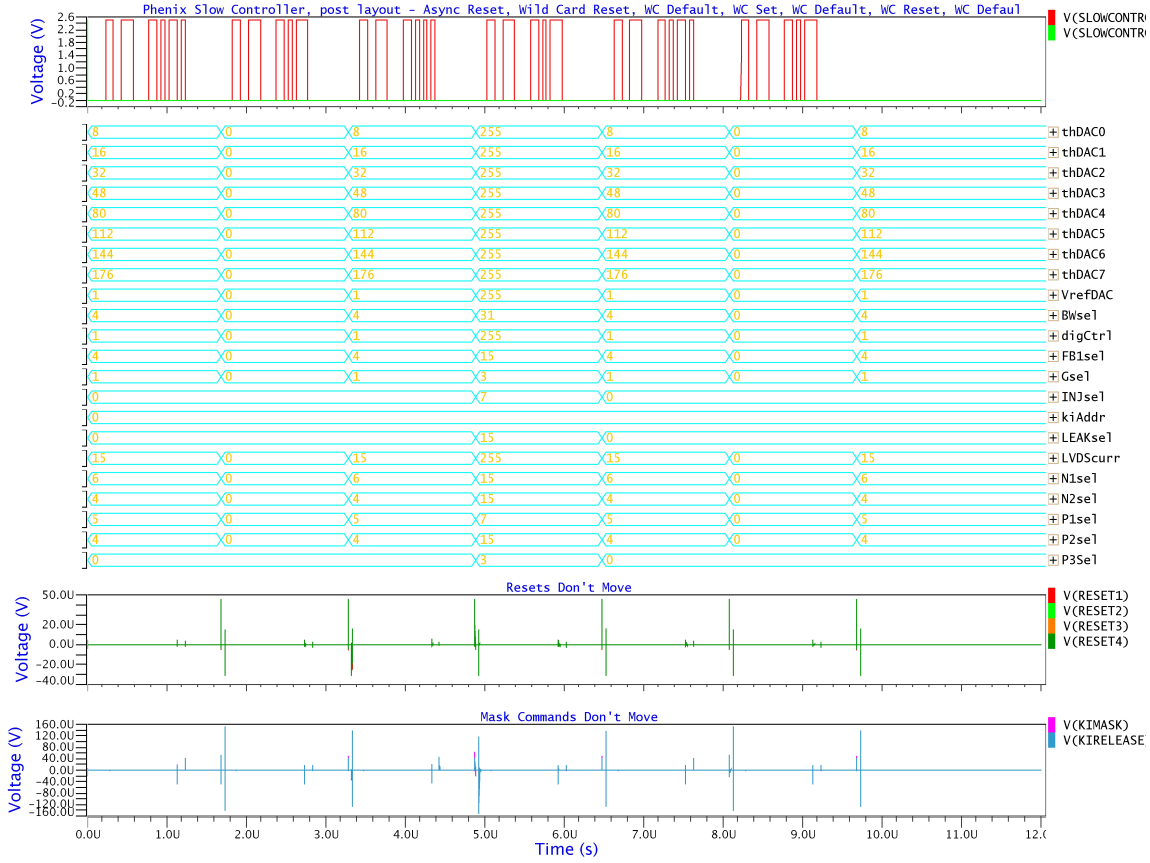


Figure 12 - Slow Control showing "Wild Register Address" commands. Note that the programmable resets and mask registers are unaffected by the wildcard.

Core

Phase Block

A primary requirement of the FPhx architecture is that it be able to read out within four beam cross-over periods an event that contained four hit strips. In other words, regardless of activity level, long latency cannot be tolerated. Hits must be sensed, amplified, discriminated, captured, sorted, serialized and read out of the chip. Moreover, the requirements do not allow for dead time, so if an FPhx chip receives an event in beam cross-over period "N", it must be able to deal with an event in beam cross over period "N+1" and in beam cross-over period "N+2", etc.

These twin requirements of low-latency and zero dead time give rise to the notion of phase architecture. During any given phase, amplification, discrimination, acquisition, sorting, serialization and output must happen. However, amplification, discrimination and acquisition are happening for hits that are occurring in *this* phase. Sorting is happening for hits that occurred in the last phase. Serialization and output are happening for hits that occurred at least two phases ago.

The requirement of "four-hits-in-four-beam-crossings" suggests that a cyclic progression of four phases (Phase 1 -> Phase 2 -> Phase 3 -> Phase 4 -> Phase 1 -> etc) could be used as the cornerstone of an architecture built to work for Phenix. During Phase 1, some circuitry would deal with amplification, discrimination and acquisition of hits that occur in Phase 1. Other circuitry would deal with the sorting of hits acquired in Phase 4. Still other circuitry would deal with the serialization and output of hits that were acquired in Phase 3 and earlier. Similarly, during Phase 2, some circuitry would deal with amplification, discrimination and acquisition of hits that occur in Phase 2. Other circuitry would deal with the sorting of hits acquired in Phase 1. Still other circuitry would deal with the serialization and output of hits that were acquired in Phase 4 and earlier.

This approach would require redundant circuitry. For example, there would be Phase 1 acquisition circuitry and Phase 2 acquisition circuitry and Phase 3 acquisition circuitry and Phase 4 acquisition circuitry. Moreover, there would have to be additional circuitry to manage the flow of data through these different phases. However, this approach would enable the job to be done without requiring excessive speed and the consequent power that would require.

One problem with this approach, however, is how to make it deal with variable event sizes. For example, even though "four-hits-in-four-beam-crossings" is the measuring stick for Phenix, events of only one hit will be very common and events with more than four hits are actually desirable.

Fortunately, this architecture can deal with events of less-than four hits easily. The only consequence is a slight inefficiency in the readout bandwidth. However, since synchronization words will be output whenever there is no data to be output, this slight inefficiency should allow the data acquisition system to remain in sync with the FPhx chips.

Larger hits pose a bigger problem. Of necessity, amplification and discrimination must occur in parallel for all 128 channels. Therefore, larger events will have no impact on this activity. Acquisition, even though it has four phases, also occurs

in parallel for all 128 channels. Sorting, serialization and output, however, will be impacted by the size of the event. Larger events will take longer to sort, longer to serialize and longer to output. Serialization and output overloading can be mitigated by FIFOs which can be filled by large events and drained during empty or low occupancy beam cross-over periods. This could have an impact on the "four-hits-in-four-beam-crossings" requirement. For example, if there are several large events in a row, the first large event might output four hits in four beam-crossings, but, because of the FIFOs, the second and subsequent hits might not get their first four hits out in four beam-crossings. This inefficiency should be very low, however, given the expected beam profile in Phenix.

Unfortunately, very little can be done about sorting (also known as zero suppression). Larger events will take longer to sort. Therefore, some mechanism must be in place to halt the advancing of the phase in the event that sorting cannot be completed in the required time. This is the principle responsibility of the Phase Block.

The Phase Block maintains the eight-bit beam cross-over counter or BCO Counter. This is the time stamp. It advanced on the rising edge of each BCO clock. The Phase Block also maintains the phase state machine which advances on the rising edge of each BCO clock *provided* that the sorting has been completed. If the sorting has not been completed, the phase does not advance, and the acceptance of further hits by the FPhx chip is suppressed until the phase can finally advance. Finally, the Phase Block operates as indirect addressing logic relating a particular phase to a particular time stamp so that when data is output, the hits are associated with the correct time stamp regardless of how many times the phase advance was blocked.

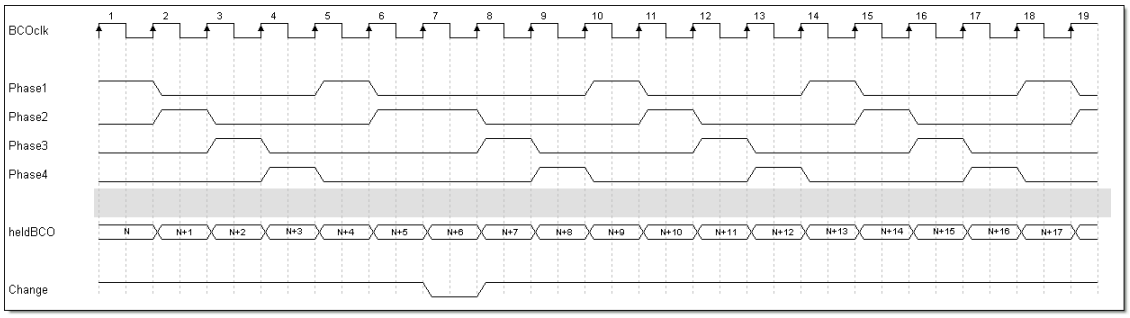


Figure 13 - Phase Block Operation

Finally, the functions of amplification, discrimination, acquisition, sorting, serialization and output should be related to physical blocks in the FPhx chip. Amplification and discrimination are the jobs of the analog front end. An injection of charge goes in one end; it is amplified (amplification); that amplified signal is compared to threshold values (discrimination) and a binary signal indicating the presence or absence of a hit as well as the magnitude of that hit is driven to the back end. Acquisition is accomplished by the Mask Array and Channel Control Array blocks. The binary output of the front end is latched by the Channel Control Array provided the Mask Array is not blocking that channel. Any unblocked binary signals are stored in the Channel Control Array (acquisition). It is at this point that the phase splitting occurs. Depending on the state of the Phase Block, incoming binary signals are stored as Phase 1, Phase 2, Phase 3, or Phase 4 hits. Sorting is accomplished through the Token Control and

Addressing arrays which, governed by the state of the Phase Block, take the appropriate Phase Hits and select one at a time (sorting) to release its address onto the bus. Serialization and output are controlled by the FIFO/Serializer.

FIFO/Serializer

The FIFO/Serializer consists of a circular FIFO, a pair of 20-bit serializers, and the Count_40 Heartbeat which provides to the rest of the chip clocks that are derived from the output clock.

The circular FIFO is very straightforward. It is twenty bits wide and 32 words deep. The input pointer points at the next available FIFO stage. If the input pointer plus two is equal to the output pointer, then the FIFO is full. If there is a valid word, then it is loaded into the stage pointed at by the input pointer and the input pointer advances by one to the next stage unless the FIFO is full. If the FIFO remains full, new words read into the FIFO overwrite old words. The output pointer points at the top two words of the FIFO. When a word is popped off the top of the FIFO and serialized, the output pointer advances by two to point at the newest top of the FIFO. If the output pointer equals the input pointer, the FIFO is empty.

The Count_40 Heartbeat uses the output clock (a.k.a. Serial Clk) and the programmable oneActiveLine/twoAcitveLines signal (see Table 2) to generate the readClk, grabClk, newSerialWord, newParallelWord, and resetParallelWord (see Figure 14 for the derived signals for twoActiveLines).

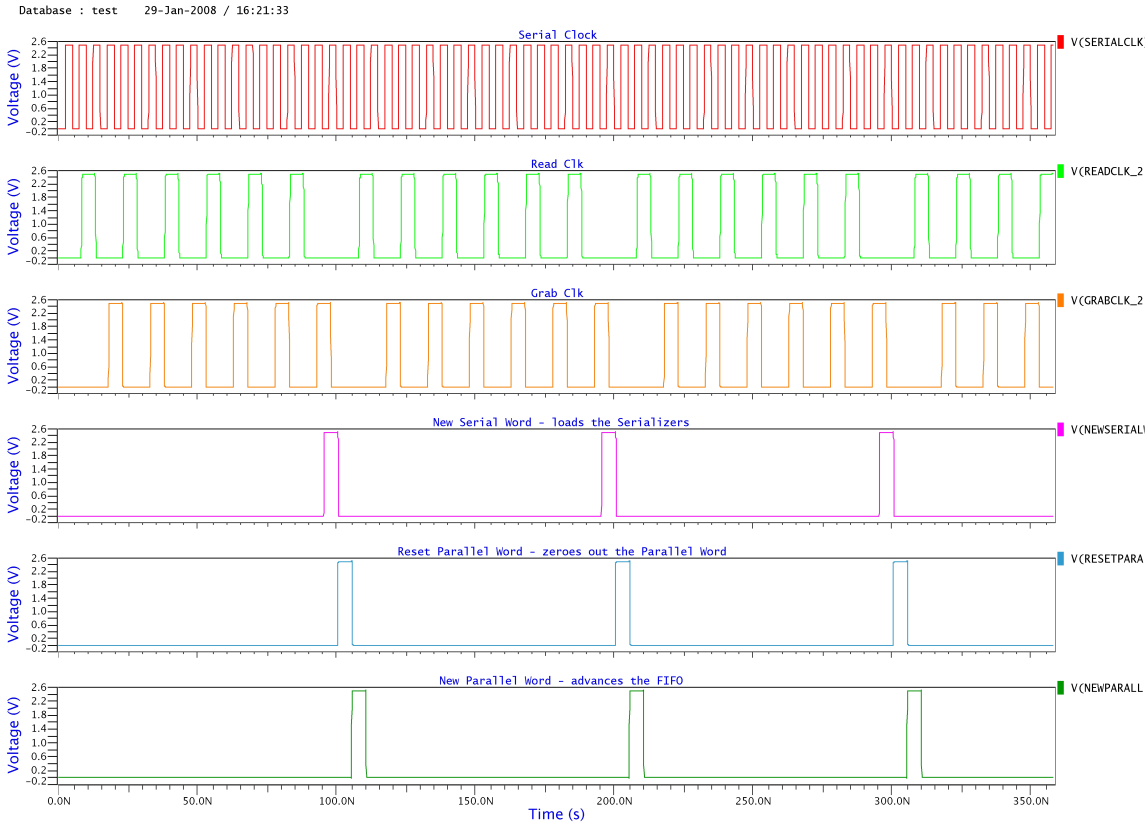


Figure 14 - Count_40 Heartbeat in action.

1. **newSerialWord** – loads the data at the top of the FIFO into the Serializers.
2. **resetParallelWord** – resets the data at the top of the FIFO. FIFO data words reset to a sync word.
3. **newParallelWord** – advances the FIFO output pointers to the next two output words.
4. **readClk** – This is the clock used by the Core to sort hit channels and transfer them to the FIFO serializer.
5. **grabClk** – This is the clock used by the FIFO/serializer to grab data being transferred from the Core to the FIFO. Note that the grabClk is a slightly delayed version of the readClk. The time delay from readClk to grabClk defines the speed with which signals must be driven from the Core.

Note that it is assumed that the serial clock is 20 times the beam cross-over clock. Since there is no Phase Locked Loop in the chip, the chip assumes that the serial clock and beam cross-over clocks provided to it are already phase locked to one another. Therefore, even though the beam cross-over clock is not involved in the Count_40 heartbeat circuit, the circuit is designed under the assumption that 20 serial clock periods equals one beam cross-over clock periods.

The only real difference between one active line and two active lines is the number of readClk/grabClk signals per cycle. See Figure 15.

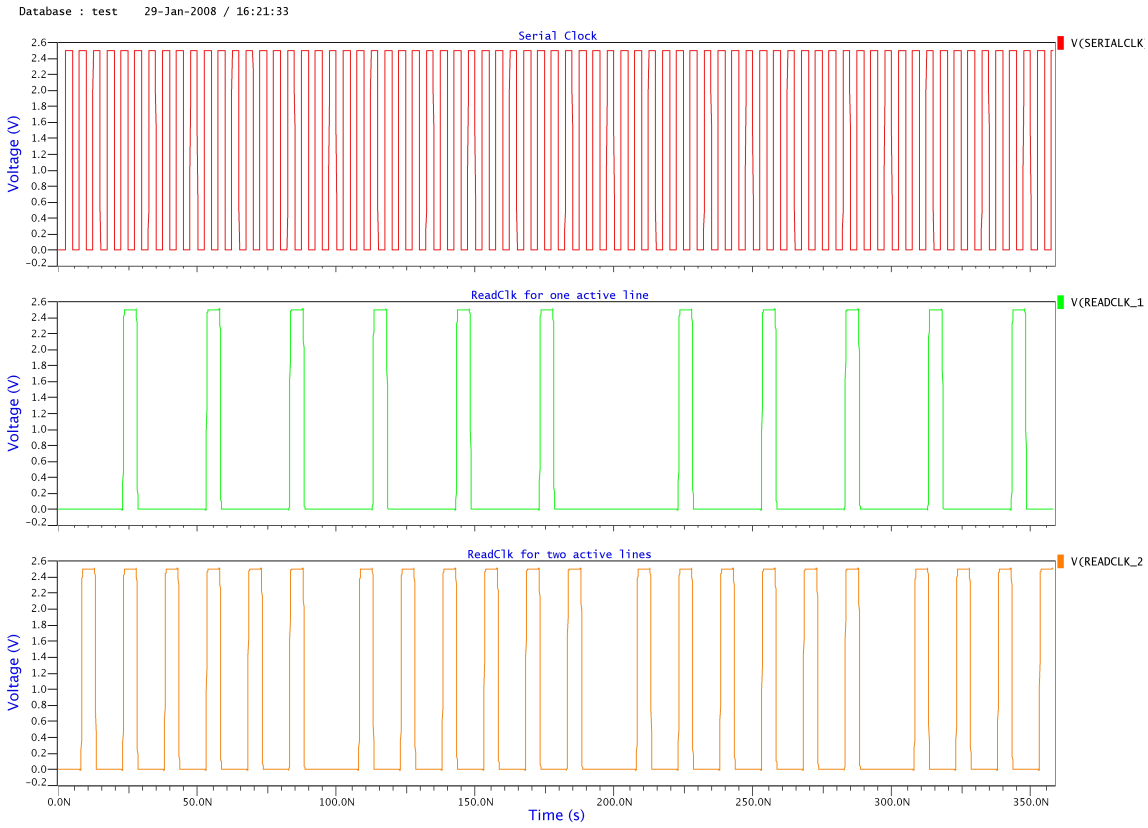


Figure 15 - A comparison of Read Clock frequencies under one and two active lines

When there are two active lines, the Count_40 heartbeat counts to 20 (one count per serial clock period) and creates 6 read clock pulses and 6 grab clock pulses in that time. It then issues the newSerialWord, resetParallelWord and newParallelWord signals (see Figure 14). When we have one active line, the Count_40 heartbeat counts to 40 (two full beam cross over periods) and creates same 6 read clock pulses and 6 grab clock pulses in that longer time. It then issues the newSerialWord, resetParallelWord and newParallelWord signals (see Figure 14). Obviously, the periods between read clock and grab clock pulses is longer when using one active line than when using two active lines. Equally obvious should be that newSerialWord, resetParallelWord and newParallelWord are issued every other beam cross-over period.

The serializers are very straightforward and they are shown in Figure 16. At any given time, the top word in the FIFO is connected to the first serializer and the second-to-top word in the FIFO is connected to the second serializer. When the Count_40 heartbeat issues a newSerialWord signal, those top two words are loaded into the two serializers. When the Count_40 heartbeat issues a resetParallelWord signal, the top two words in the FIFO are reset back to sync words. When the Count_40 heartbeat issues a newParallelWord signal, the output pointers advance to the next two top words. If the FIFO is empty, the output pointers do not advance, and the next time the Count_40 heartbeat issues a newSerialWord signal, the serializers will load sync words.

Under two active lines, the user is latching both serial outs, so over the course of 20 serial clock periods, two entire 20-bit words will be read out Under one active line, the user is ignoring SerialOut2, and over the course of 40 serial clock periods, two entire 20-bit words will be read out of SerialOut1.

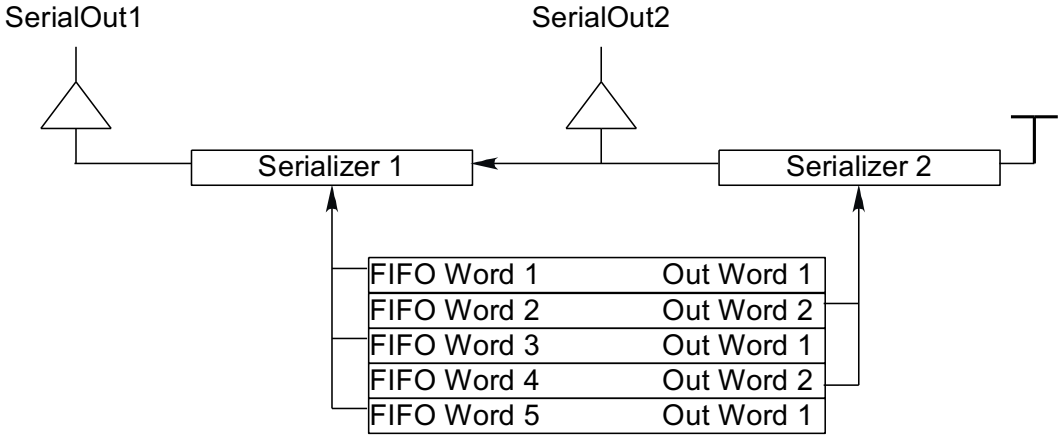


Figure 16 - The Serializers

Mask Array

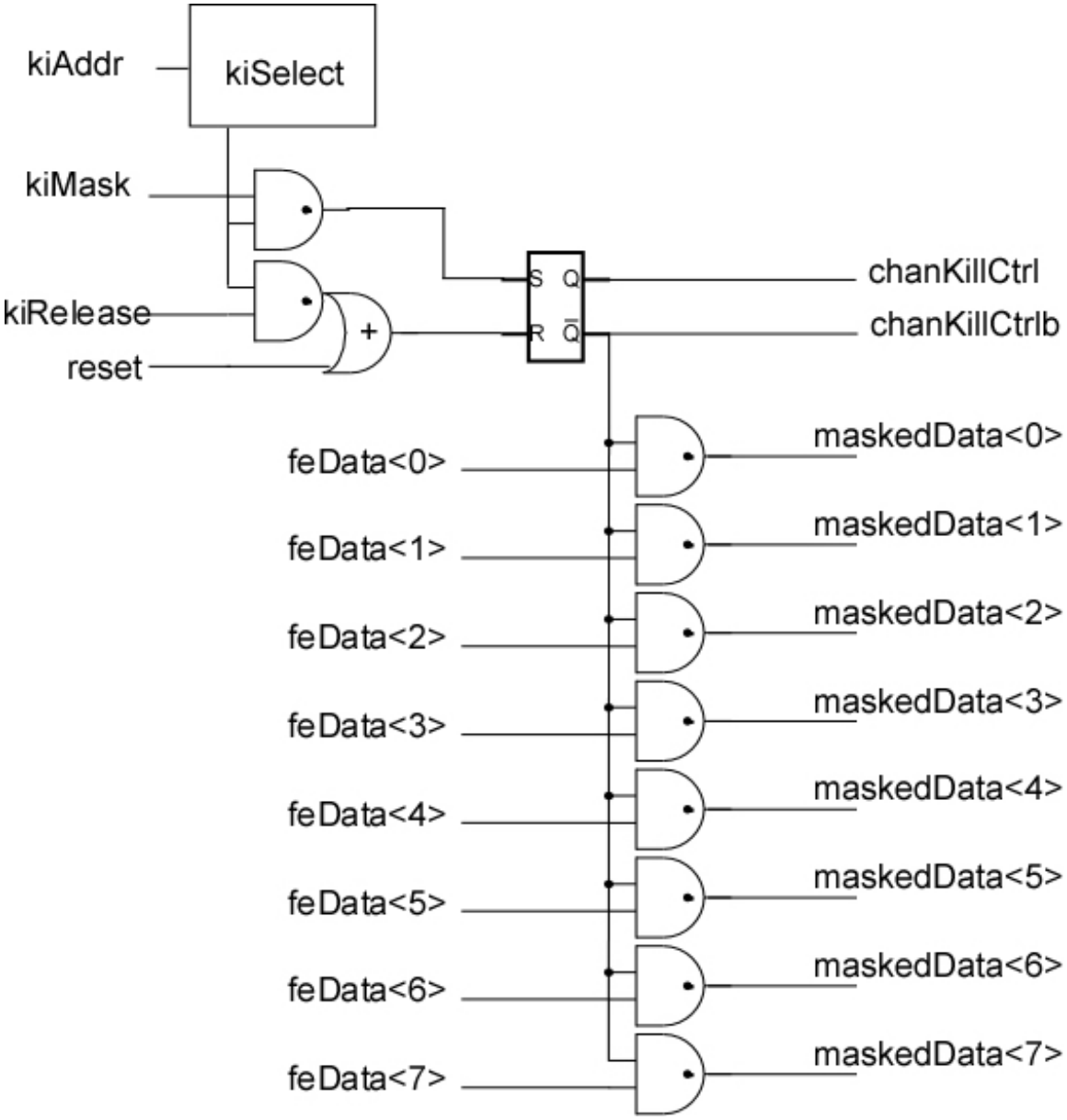


Figure 17: The Mask Control Sub-circuit

The Mask Array is an array of 128 identical masking sub-circuits, one per channel (see Figure 2). The purpose of this array is to enable the user to mask off individual channels as desired. Control of the Mask Array and of which channels are masked and which are free is accomplished through the Slow Controller.

At the heart of each sub-circuit is an SR-flip-flop that stores the channel's current status (masked or unmasked). If the flip-flop is set, then this channel is being masked and a bank of eight and gates blocks the front-end data (feData<0-7>) from being passed on to the rest of the back end circuitry. If a channel is unmasked, then the front-end data is passed into the back end circuitry unchanged.

The SR-flip-flop is principally controlled by three signals – kiAddr, kiMask and kiRelease. kiAddr is an eight bit wide word that is the address of a particular channel (e.g. X0000001 for channel 1, X0000010 for channel 2, etc. The unique significance of the most significant bit will be explained shortly.) It comes from the Slow Controller and ultimately from the Slow Controller Command Word itself. The eight bit data portion of the Slow Controller Command Word is the address that is passed to the Mask Array as kiAddr. Signals kiMask and kiRelease are activated by the Slow Controller as well, depending on whether the Slow Control Command Word was a Set command (kiMask) or a Reset command (kiRelease). If a particular channels address is given in the Slow Controller Command Word and that command word is a Set, then that channel will be masked off. Conversely, if a particular channels address is given in the Slow Controller Command Word and that command word is a Reset, then that channel will have its mask released.

The most significant bit of kiAddr is a global command bit. If the most significant bit of kiAddr is a one, then, regardless of the address presented in the remaining 7 bits of kiAddr, all channels obey the kiMask or kiRelease command. This allows the user to mask off or release all channels at once with one command.

Control Array

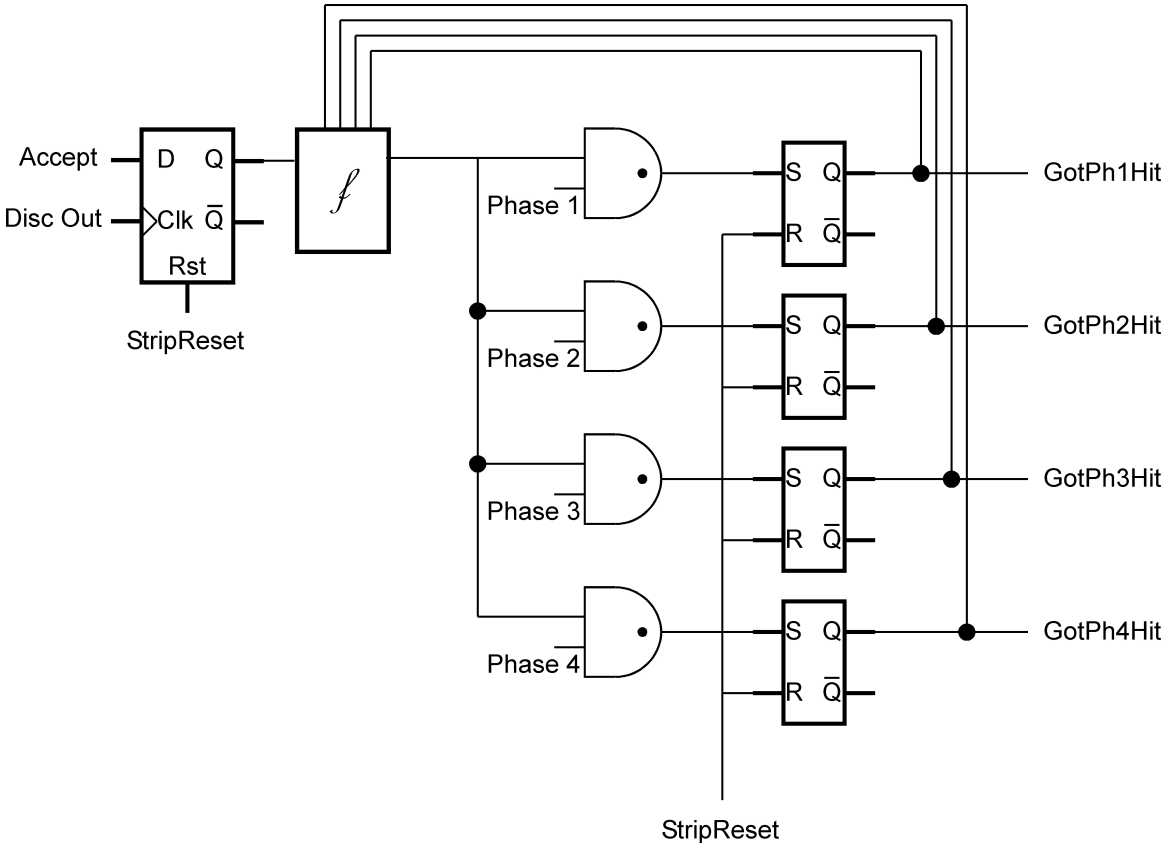


Figure 18 - How the Control Circuit Acquires Hits

The Control Array is an array of 128 identical sub-circuits (Channel Control Circuits), one per channel (see Figure 2). This array has a number of important functions, namely:

- Each hit must be captured once and only once.
- Hits must block the acquisition of future hits until the original hit is read out (i.e. a channel cannot be allowed to record and store more than one hit at a time).
- The analog section has no knowledge of time or of phases. The digital section is required to assign each hit to an appropriate phase. Simultaneously, this must assign each hit to the appropriate time stamp.
- The user must have the ability to ignore hits if the user chooses to do so.
- Strip Controllers must ignore hits if the phase does not advance.

The Channel Control Circuits accomplish these functions as follows:

- To capture a hit once and only once: acquire on the rising edge of the discriminator output using a positive edged dff.
- To block subsequent hits once a hit is stored: block the output of the acquisition dff whenever a hit is stored
- To assign hits to phases, demultiplex the signal with the phase signals
- To ignore hits as the user demands, there must be a programmable signal in the slow controller (globalAccept)
- To ignore hits when the phase does not advance, the "No Change" signal must be logically combined with the globalAccept signal to create the Accept signal. In order for Accept to be a 1, then the user must activate globalAccept through the slow controller and the "No Change" signal from the Phase Block must be inactive. If Accept is a 1, then on the rising edge of the front end's discriminator output, the Channel Control Circuit will latch a hit into the appropriate phase storage cell. If Accept is a 0, then the Channel Control Circuit does not register a hit.

Figure 18 shows how the FPhx acquires hits in the Four Phase Architecture. The signals Phase1, Phase2, Phase3, and Phase4 are broadcast in parallel to all 128 Channel Control Circuits in the Control Array. Accept is also broadcast in parallel to all 128 Channel Control Circuits after it is created in the periphery of the Core by combining a digital control signal from the Slow Controller (Register Address 2, bit 1) with the "No Change" signal from the Phase Block. Strip Reset is created in each Channel Control Circuit from the global reset signal and from the local getBus signal which activates when this particular channel acquires the bus to output its data. Therefore, as a channel is read out, it is simultaneously reset, removing the hit from the Core.

Several other functions are also accomplished in each Channel Control Circuit of the Control Array but are not shown in Figure 18. First, each Channel Control Circuit contains seven edge-triggered d-flip-flops which capture the peak state of the thermometer-encoded hit magnitude from the front-end analog-to-digital converters. Each flip-flop is latched on the rising edge of its respective hit magnitude signal. This guarantees that no matter how long it takes for a discriminator to fire, the peak magnitude will be recorded. These flip-flops are reset after the channel has been read out.

Also included in the Channel Control Circuitry is a thermometer-code to binary-code encoder. This converts the 7-bit thermometer code into a 3-bit binary code. Finally,

each of the four GotHitPhX signals are ANDed with their own PhaseX signal (e.g. GotHitPh1 is ANDed with Phase1) to produce four new GotHit signals that endure for only one Phase. By the time the Phase Block returns to that particular phase, that particular channel will have been read out and reset. These "temporary" GotHit signals are ORed together to form a single phase-less GotHit signal that is used by the hitOR array to create the hitOR signal. This signal is used as a diagnostic signal during testing.

HitOR Array

The hitOR Array is really just a gigantic, distributed OR gate that collects all the GotHit signals in the whole chip and combines them into one signal that gets driven off the chip to be used as a diagnostic signal.

This signal *could* be used by the Phase Block to latch the current state of the time stamp. The existing architecture does not require this, so to avoid possible time-walk issues related to the delay in processing the hitOR signal, the hitOR signal is only used as a diagnostic. Depending on the success of the first prototype chip, this strategy may be changed.

Token Logic

The Token Logic divides the chip into individual channels, blocks of eight channels and banks of 32 channels. It performs this division with three token tiers. The Tier 3 token selects which bank of 32 has access to the bus. The Tier 2 token selects which block of 8 has access to the bus. Finally, the Tier 1 token selects which individual channel has the bus. A channel can only have the bus if it has all three tokens AND a hit to output.

The Tier 1 token is sent into the bottom channel of each block of 8 simultaneously. The Tier 1 Token Logic is show in Figure 19

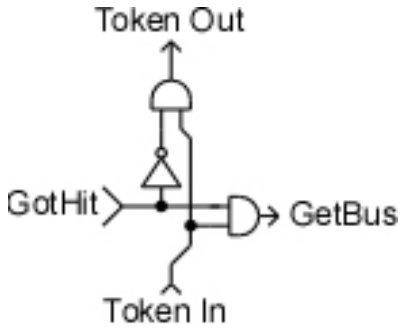


Figure 19 - Tier 1 Token Passing Logic

Obviously, if a channel has a hit, then the Tier 1 Token Out from that channel is a zero. If a channel has no hit, then Token Out equals Token In. Therefore, in each block of 8 simultaneously, the Tier 1 token will rise up the block until it reaches the first channel with a hit, and then it stops and remains there until the GotHit is cleared by the reading out the channel. If there are no (more) hits in a block of 8, then the Tier 1 Token will pop out of the Token Out signal from the top channel in the block of 8. When that happens, readout for this block of eight is done. Tier 1 tokens do not pass from one block of eight to another.

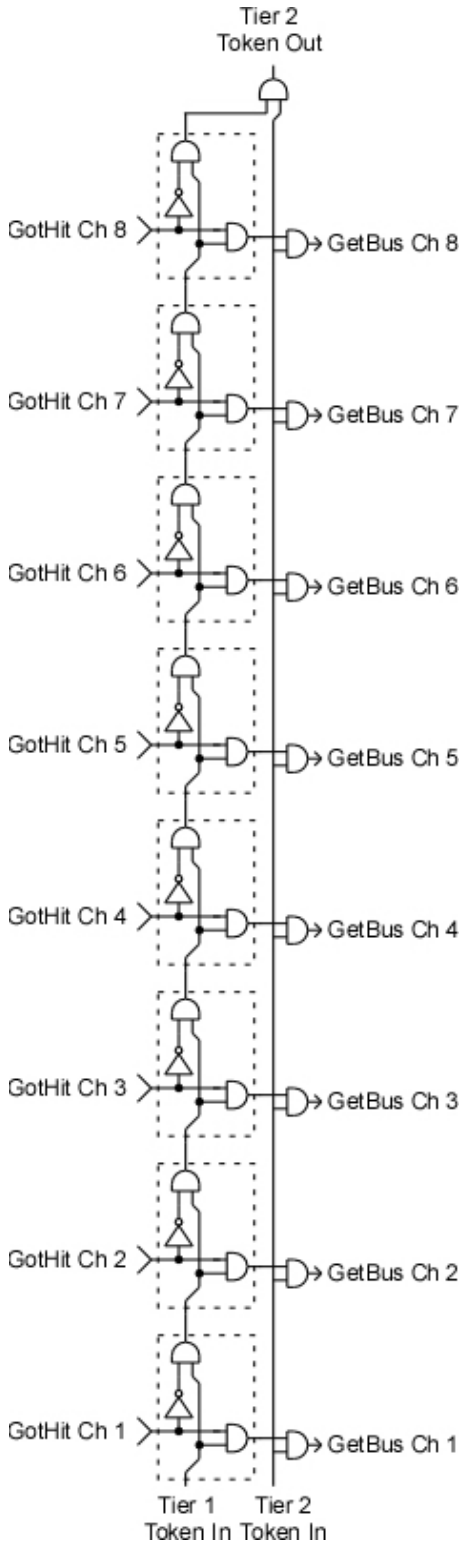


Figure 20 - Tier 2 Token Passing in a Block of 8

The Tier 2 token is sent into the bottom block of 8 in each bank of 32 simultaneously. If the Tier 2 Token In is a one for a particular block of 8, then all eight

channels in that block of eight are eligible to get the bus provided they also have a Tier 1 token. (See Figure 20.) If the Tier 2 Token is a zero for a particular block of 8, then none of the channels are eligible to get the bus, even if they have a hit. When the Tier 1 token reaches the top of a block of 8, then there are no more channels to be read out from that block of 8. This releases the Tier 2 token to the next block of 8. When the Tier 2 token pops out of the top of the fourth block of eight in a bank of 32, then readout from that bank of 32 is complete.

The Tier 3 token is sent to the bottom of the bottom bank of 32 only – i.e. to the bottom of the whole chip. If the Tier 3 Token is a one for a particular bank of 32, then all 32 channels in the bank of 32 are eligible to get the bus provided their block of 8 has the Tier 2 token and their channel has the Tier 1 token and they have a hit. If the Tier 3 Token is a zero for a particular bank of 32, then none of the 32 channels in that bank of 32 are eligible to get the bus even if their block of 8 has the Tier 2 Token and their channel has the Tier 1 Token and they have a hit. (See Figure 21.)

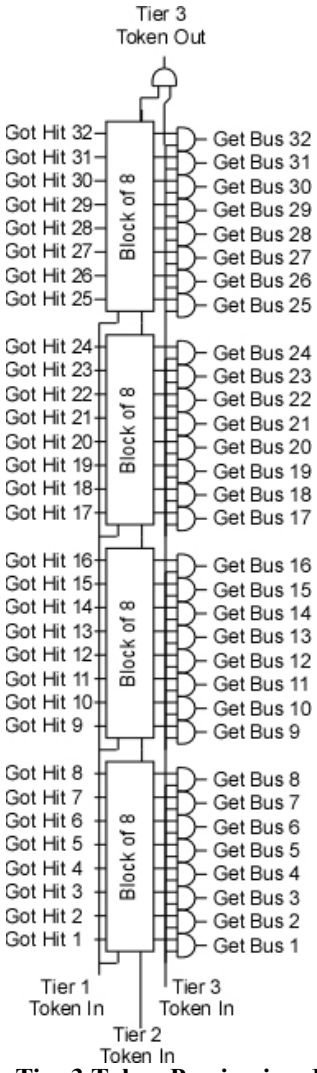


Figure 21 - Tier 3 Token Passing in a Bank of 32

When the Tier 2 Token pops out of the top of the top block of 8 in a bank of 32, then readout from that bank of 32 is complete and the Tier 3 token is passed on to the next bank of 32. Tier 2 Tokens are not passed between banks of 32.

When the Tier 3 Token pops out of the top of the top bank of 32, then the whole readout is done.

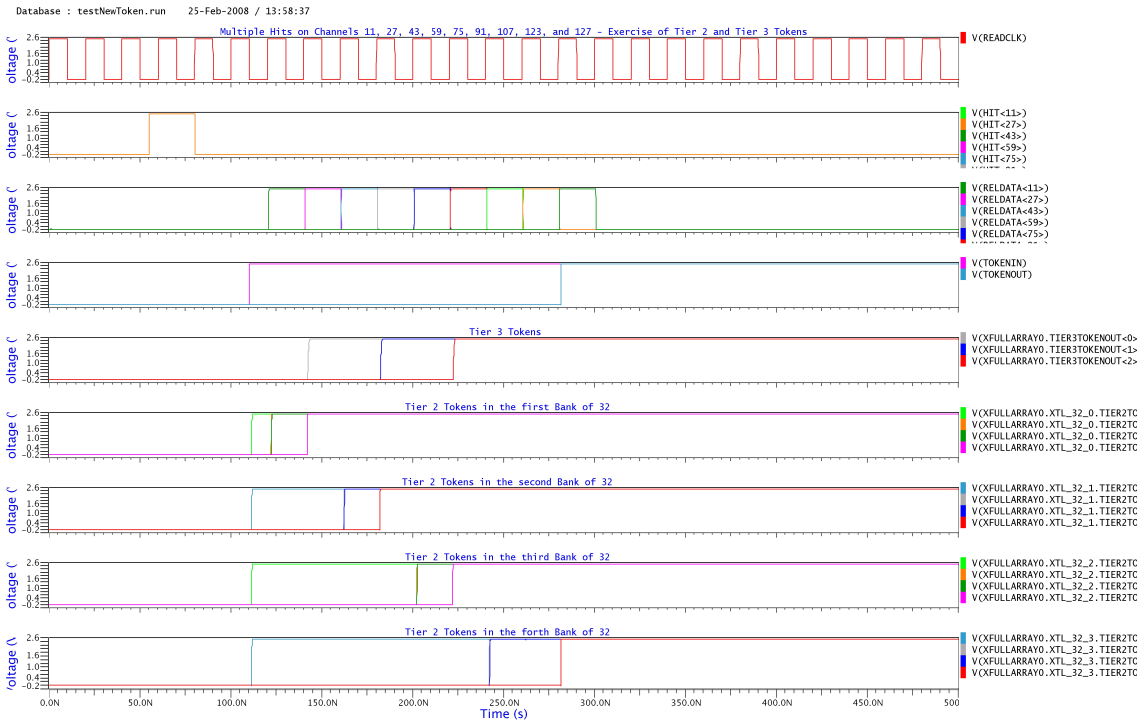


Figure 22 - An example of the propagation of tokens up the chip

When a readout **starts**, simultaneously, the Tier 1 token is presented to the bottom channel in each block of 8, the Tier 2 token is presented to the bottom block of 8 in each bank of 32, and the Tier 3 token is presented to the bottom bank of 32 in the chip. Tier 1 Tokens begin rising up their respective blocks of 8. Tier 2 tokens can also begin to move within their respective banks of 32 as Tier 1 tokens pass through empty blocks of 8. Since Tier 1 Tokens stop progressing when they reach a channel with a hit, Tier 2 Tokens will also stop rising when they reach that block because Tier 1 Tokens popping out of the top of a block of 8 release the Tier 2 tokens to move on to the next block of 8. Tier 3 Tokens will also be stopped from rising to the next bank of 32 because the Tier 2 token was stopped from rising. At any given time, using this purely combinatorial logic, only one channel can possibly have a hit and a Tier 1 Token and a Tier 2 Token and a Tier 3 Token. That channel is the lowest channel in the whole chip that has a hit and that channel will get the bus on the next rising edge of the readClk. Acquiring the bus will cause the channel to dump its data on the bus (Location and ADC) and it will also erase the hit from the channel, releasing the Tier 1 Token to continue up its block of eight. Consequently, this (potentially) releases the Tier 2 Token to rise up to the next block of eight (if there are no more hits in this block of 8). This, in turn, (potentially) releases the Tier 3 Token to rise up to the next bank of 32 (if there are no more hits in this bank of 32).

As stated in the previous paragraph, when a readout starts, the Tier 1 token is presented to the bottom channel in each block of 8, the Tier 2 token is presented to the bottom block of 8 in each bank of 32, and the Tier 3 token is presented to the bottom bank of 32 in the chip. This means that at the bottom of the chip, the three tokens are the same signal. From the four phase architecture we know that we must acquire hits in one phase and begin to read them out in the next. That means that the token at the bottom of the chip is really just one of the phase signals. It also means that there must be four of these Token passing architectures in the chip, one for each phase. For hits acquired in Phase 1, they are read out using a token derived from Phase 2 (so they are read out in the Phase after they are acquired). For hits acquired in Phase 2, they are read out using a token derived from Phase 3, etc.

For each channel, the four signals GotHit_ChX_Phase1, GotHit_ChX_Phase2, GotHit_ChX_Phase3, and GotHit_ChX_Phase4, are ORed together to create a single signal, WantBus_ChX. This signal is passed on to the Address Array.

Address Array

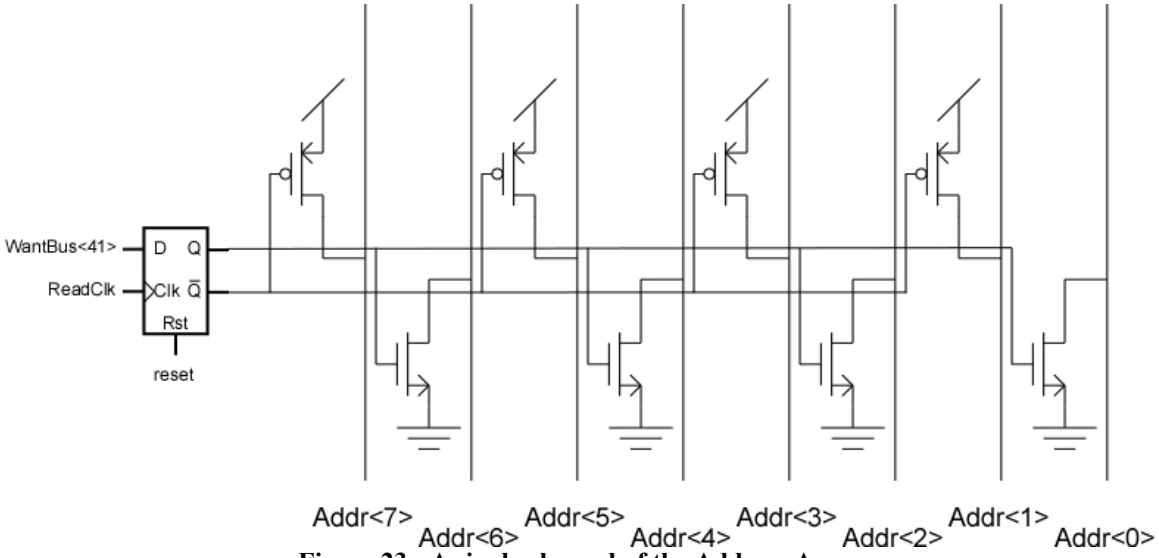


Figure 23 - A single channel of the Address Array

The token logic has already assured that one and only one channel will be allowed to request the bus at any given clock period. The Address Array converts a "WantBus" request for a particular channel into the appropriate channel address and keeps that address active on the coreLocation bus for one ReadClk period. The conversion simply requires an array of 128 edge triggered flip-flop that trip on the rising edge of the ReadClk (one flip-flop per channel) and a set of 128 banks of eight pull-up and pull-down transistors (one unique bank per channel). Figure 23 shows the configuration for channel 41. Note that the most significant bit is always a pull-up transistor. This bit becomes the word mark bit (bit 14 in the Serial Output word) that helps ensure that the Sync word is the only possible combination of 19 zeros and 1 one.

Expected Operation

The need to RESET

It is assumed that upon power-up the chip will be reset. There is no power-on-reset built into the chip and if the chip is not reset, the state of the chip cannot be predicted. Once the reset is given, all hits will be erased, all bias voltages will be sent to their default values, the time stamp will be reset to zero, the chip will be operating with two active lines and the chip will be rejecting any hits presented to it.

Initial programming

If the chip will be exercised through its inputs (i.e. the strips), all that is needed is for the chip to be told that it can accept hits. This is accomplished by setting bit 1 of Register 2 (Digital Control) to a 1.

If the chip will be exercised through its inject Inputs (Analog or Digital) then the following sequence is assumed:

1. Bit 2 of Register 2 (Digital Control) is set to a 1. This activates the Global Inject Enable.
2. Bit 1 of Register 2 (Digital Control) is set to a 1. The enables the chip to respond to hits.

It is not recommended (at this time) that these instructions be executed simultaneously even though the Slow Controller could handle this easily. From simulations, it appears that turning the Global Inject Enable on injects charge into channels. By doing the above programming in two steps, you prevent the chips from reading out an excessively long number of spuriously hit channels.

If you want to mask off channels, the following sequence is assumed:

1. Reject incoming hits by setting Bit 1 of Register 2 (Digital Control) to a 0. If you just hit reset, it is already at a zero.
2. Download the kill pattern using Register 17 (Mask).
3. Accept future incoming hits by setting Bit 1 of Register 2 (Digital Control) to a 1.