Logic & Top-Level Simulations

LOGIC Simulations

Dialogue

This is a preliminary version for circulation before the IDR. The outstanding items listed below should be completed for the IDR where a new version of this document will be available.

This is a long document, intending to cover all aspects of the digital control circuits on the ASIC1 test structure, starting with individual logic gates and including mixed-mode simulations incorporating Verilog stimulus, full row-controller custom logic, SRAM memory banks and analog pixels.

Dialog is included after results where appropriate – the contents of this document will be discussed in more detail in the IDR.

Items outstanding…

- Top level schematic
  - Pad selection
  - Global buffering
  - Power pad duplication estimates
- High-speed Config shift reg not yet implemented
- Buffering of digital signals between arrays
- Power-up sequencing – (disabled monostables, for example?)
- Top level sim with presample pixel

Logic simulation: Logic Cell Drive Strength

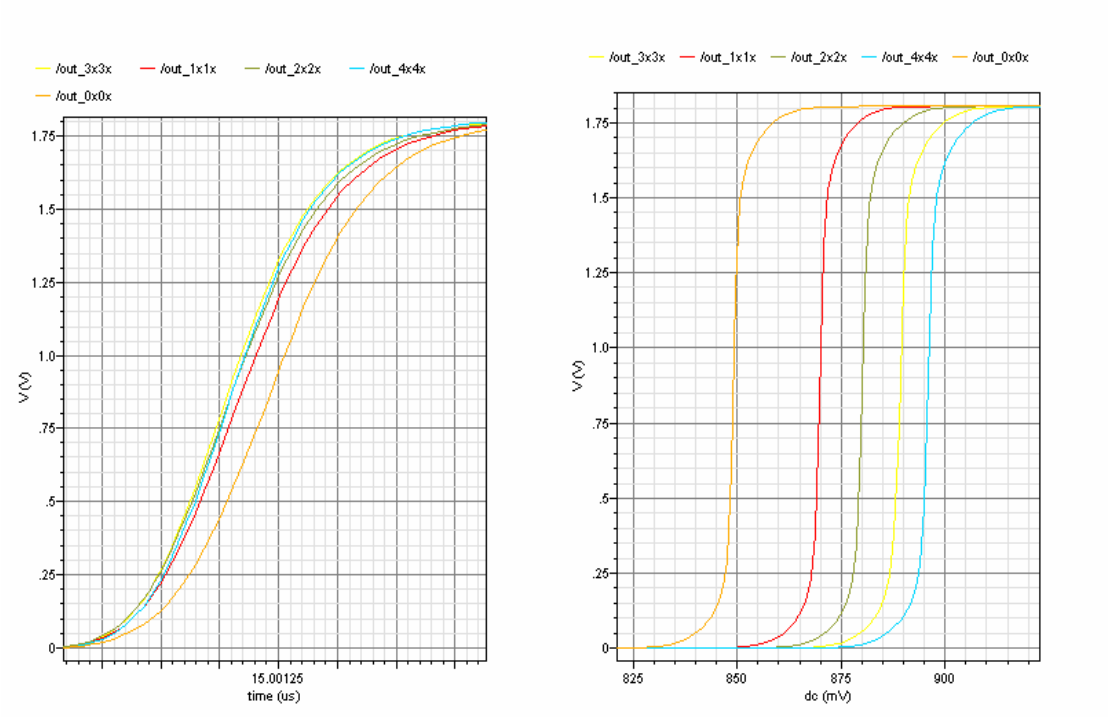
Simulation top-level = “sim_inverter”

Logic inverters were designed with different drive strengths. Functional logic blocks (2 and 3 input NAND and NOR gates) were designed to the minimum drive strength only.All transistors used are 1.8v parts to reduce power consumption.

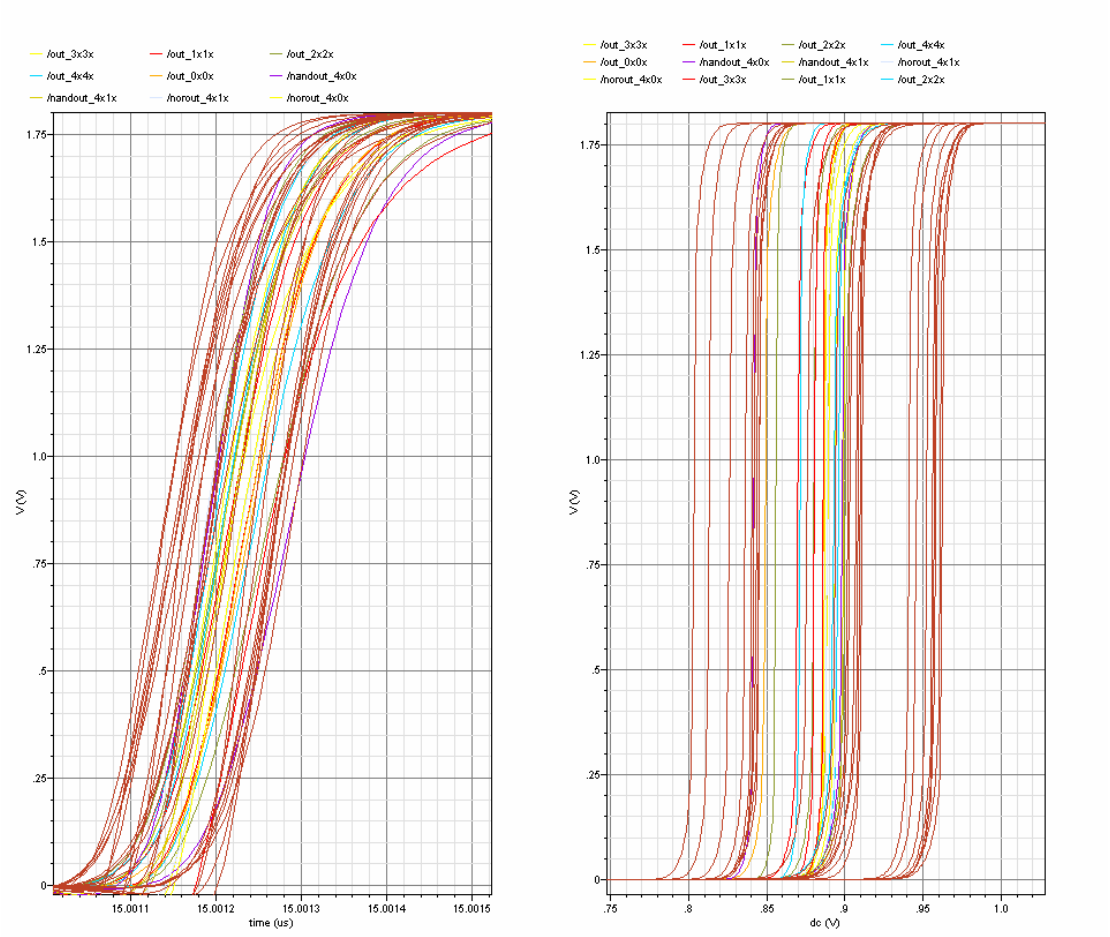
Cell variant	**x0**	**x1**	**x2**	**x3**	**x4**
Test load	10fF	30fF	55fF	90fF	125fF
Rise time	132p	129p	133p	125p	125p
Fall time	113p	120p	127p	129p	133p
Pmos width	0.8	2.0	3.5	6.0	8.4
Nmos width	0.3	0.8	1.4	2.2	2.9

Above: Tabulated results for typical process corner. Rise and fall time (for these tests) are recorded as the transition time between 15% and 85% of the full-scale 1.8v signal.

Results waveforms



Above: Inverters of each strength variant are simulated together with their corresponding test load. Transient waveforms are shown on the left (50ps per division); DC sweep to determine switching point is shown on the right.



Above: Inverters and logic gates of each strength variant are simulated together with their corresponding test load in all process corners. Transient waveforms are shown on the left (100ps per division); DC sweep to determine switching point is shown on the right: spread approx 200mV.

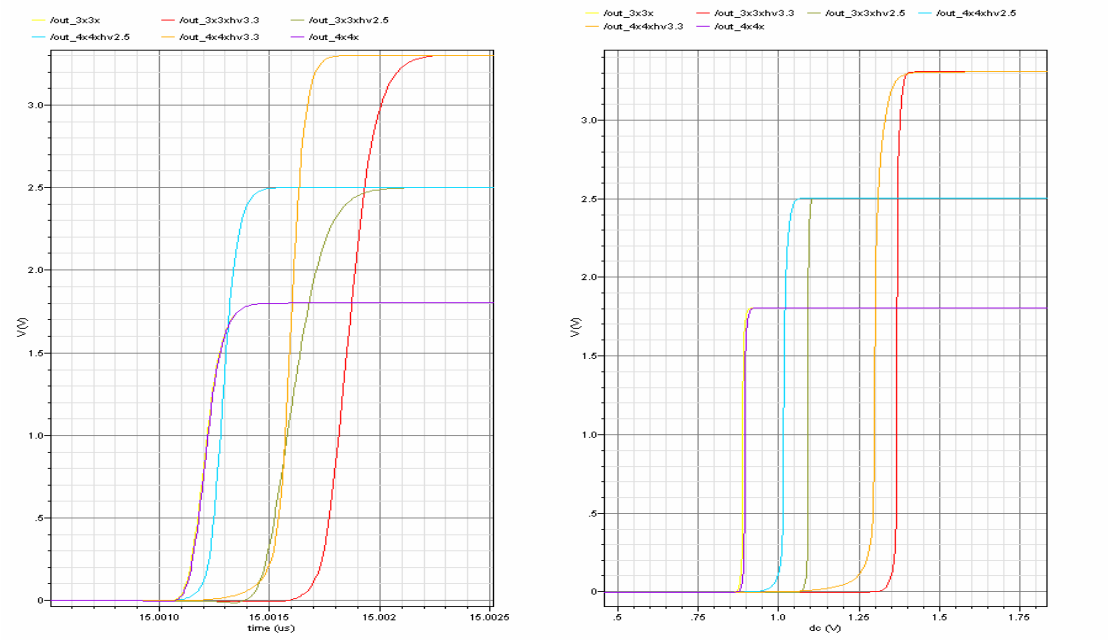
Logic simulation: HV Logic Cells

Simulation top-level = "sim_inverter"

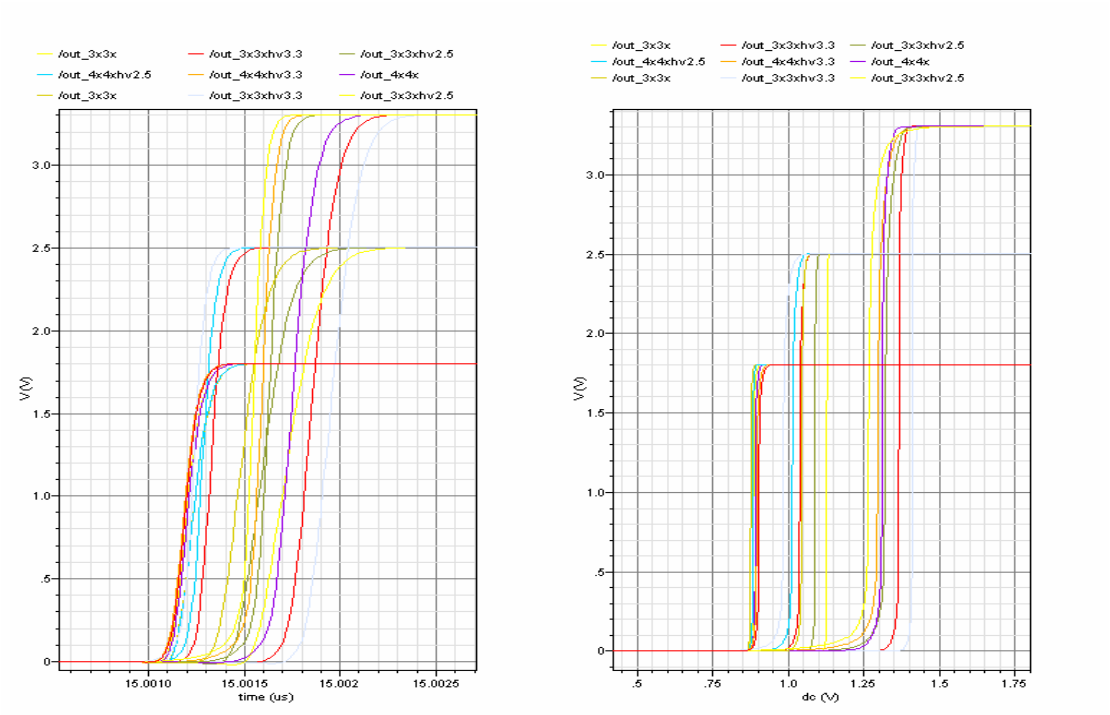
Specifically and only for driving the write transistor of the SRAM cells, inverters for high voltages (>1.8v) were created: The x3_hv and x4_hv strength inverters are copies of their low-voltage equivalents, with the nmos transistor enlarged to the same dimensions of the pmos. The equal transistor sizing lowers the switching point such that these cells could safely be used with a 1.8v logic input (and guarantees that '1' and '0' are both correctly identified, although the noise margin on a '1' will be smaller). Increasing the size of the nmos device means the rise- and fall-times of these devices are compromised (unequal).

Cell variant	**x3_hv**	**x3_hv**	**X4_hv**	**X4_hv**
Test load	90fF	90fF	125fF	125fF
VDD	2.5	3.3	2.5	3.3
Rise time	235p	200p	110p	111p
Fall time	131p	131p	82p	103p
Pmos width	6.0	6.0	8.4	8.4
Nmos width	6.0	6.0	8.4	8.4

Above: Tabulated data for typical process corner



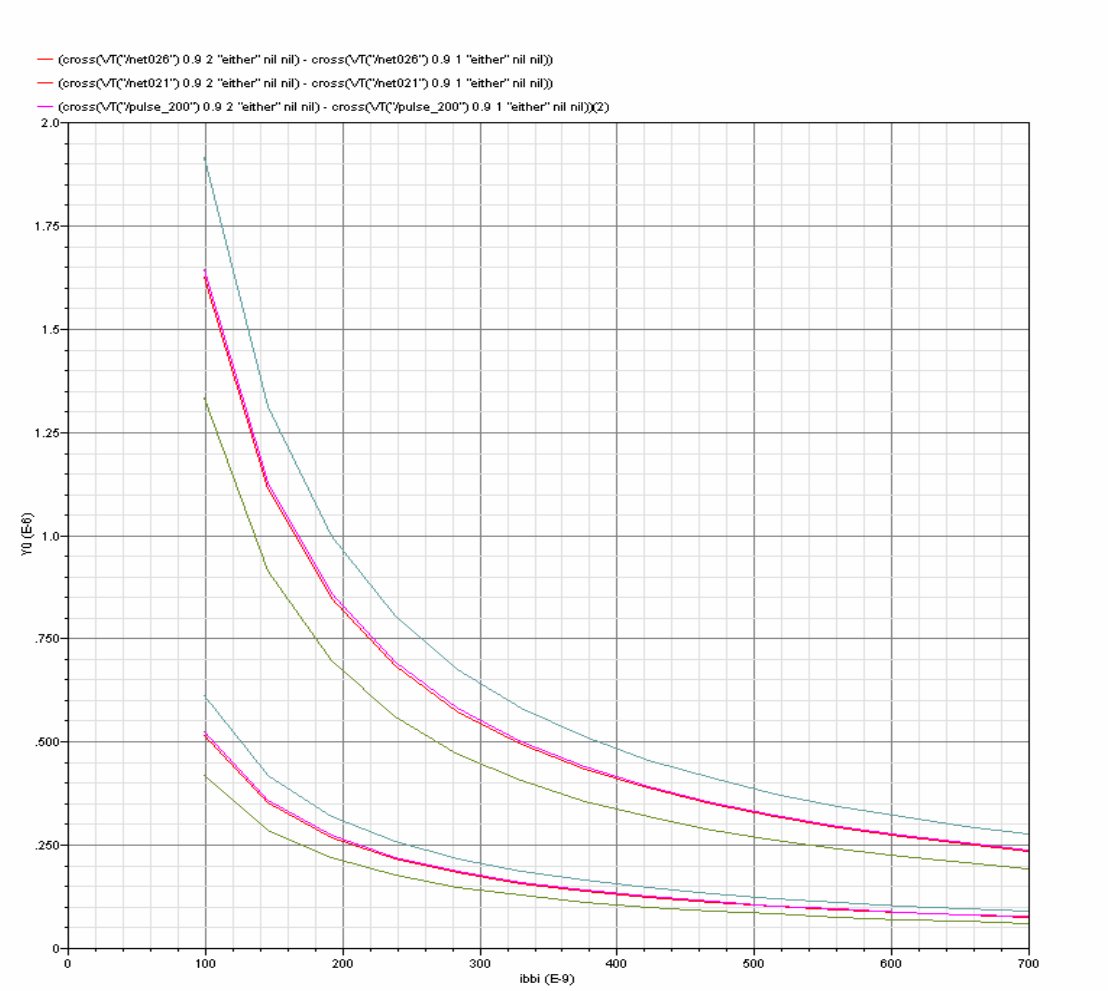
Above: Standard and high-voltage x3 and x4 inverters are simulated with their respective load. 2.5v and 3.3v supplies for the hv parts are simulated to check the switching point is a safe margin from the 1.8v supply. Transient results are shown on the left (500ps per major division), DC sweep shown on the right.



Above: Standard and high-voltage x3 and x4 inverters are simulated with their respective loads at 1.8v, 2.5v and 3.3v supplies in three process corners.

Logic simulation: Monostables

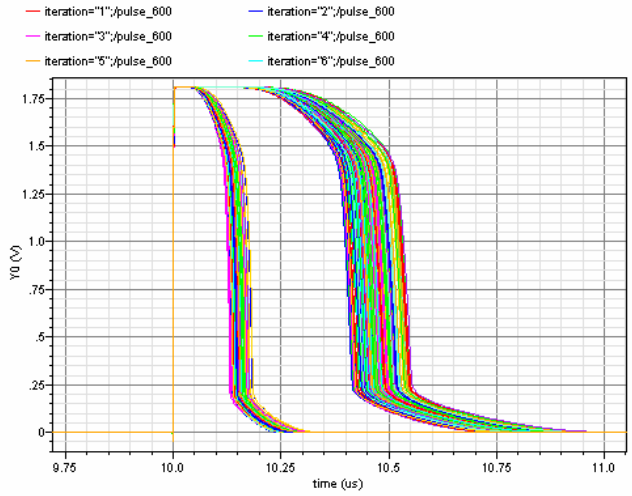
Two monostables were designed with delays in ratio 1:3. These can provide the reset timing for the pre-sample pixel, and also the correct length hit pulse for the preshape pixel. The graph and table below allow the desired timing to be selected by setting the appropriate bias current. The monostables can be triggered with a short or long pulse, the output will be high for the desired time.



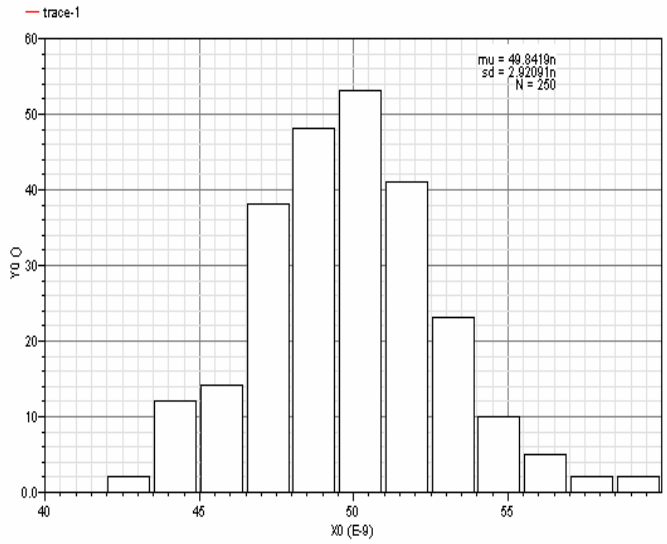
Above: Capacitor corners are checked since these will have the dominant effect on the timing. The timing ratio remains consistent, so the external control would allow the circuits to be trimmed to the desired time delays. These circuits can be evaluated on ASIC1 to determine how localalised the control of these needs to be.

Bias current	**“200ns” monostable output**	**“600ns” monostable output**
215nA	~250nS pulse	~750nS pulse
265nA	~200nS pulse	~600nS pulse
350nA	~150nS pulse	~450nS pulse

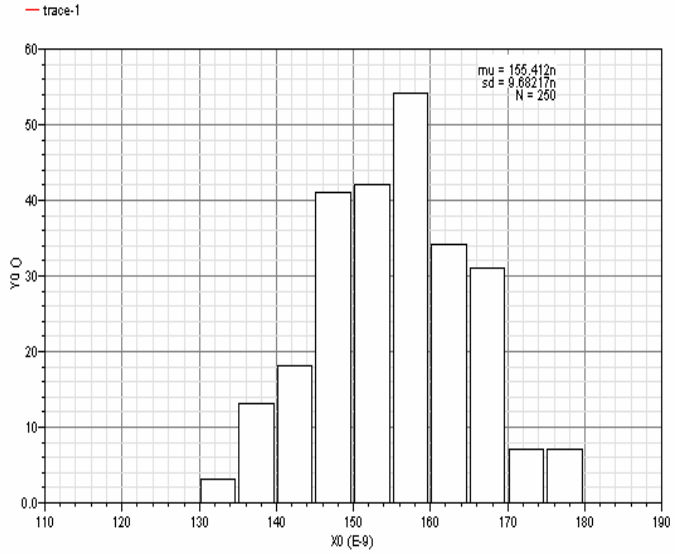
A monte-carlo simulation is run to check the mismatch effects on the monostable circuits.



An accidental scaling of the formula used in the monte-carlo setup means these results must be multiplied by 3 to yield correct results:



Mu = 149.523ns
Sd = 8.763ns



Mu = 466.236ns
Sd = 29.047ns

Early simulation results with original OPIC sense amplifier, since superseded (see following section)
Some relevance remains hence this section is retained with this note.

Logic simulation: 1 bit SRAM registers: Data Sense (readout)

Simulation top level = “sim_sram_reg_parasitics”

Circuit stimulus/scenario

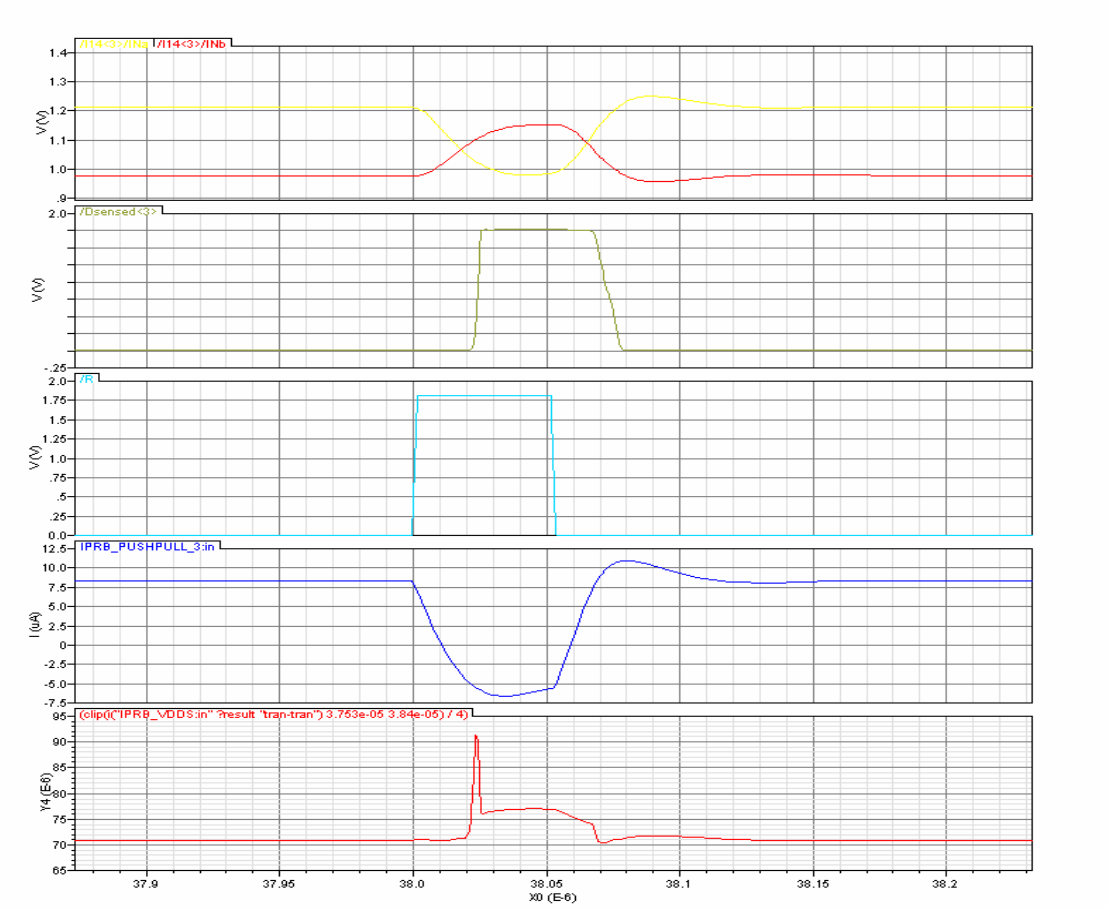
SRAM operation is evaluated using a bank of 19 1-bit registers at various points along a full-length readout column (ie near sense amplifier, mid way along and furthest from sense amplifier).

Signal line model consists of series resistance of 8 ohms and capacitance to ground for each pixel (50um). Capacitance is estimated at 10fF for parasitics; a further 30pF is added to account for the 18 other SRAM cells that are disconnected from the line (Drain of 0.5um wide inactive NMOS device).

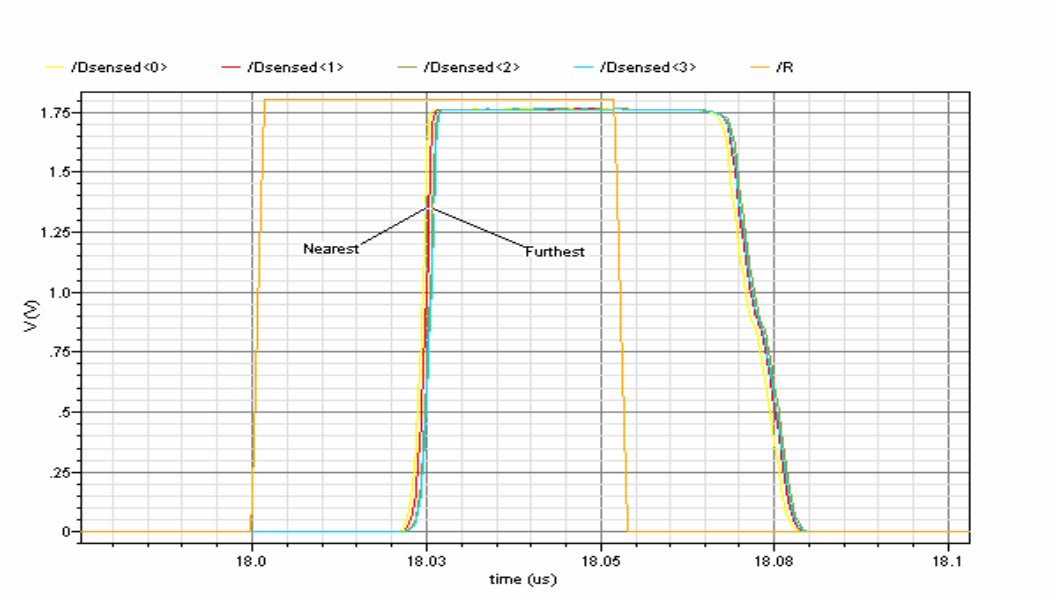
Simulation variables

icompbias1 = 6u	unitR = 8
icompbias2 = 16u	unitC = 40f

Results waveforms



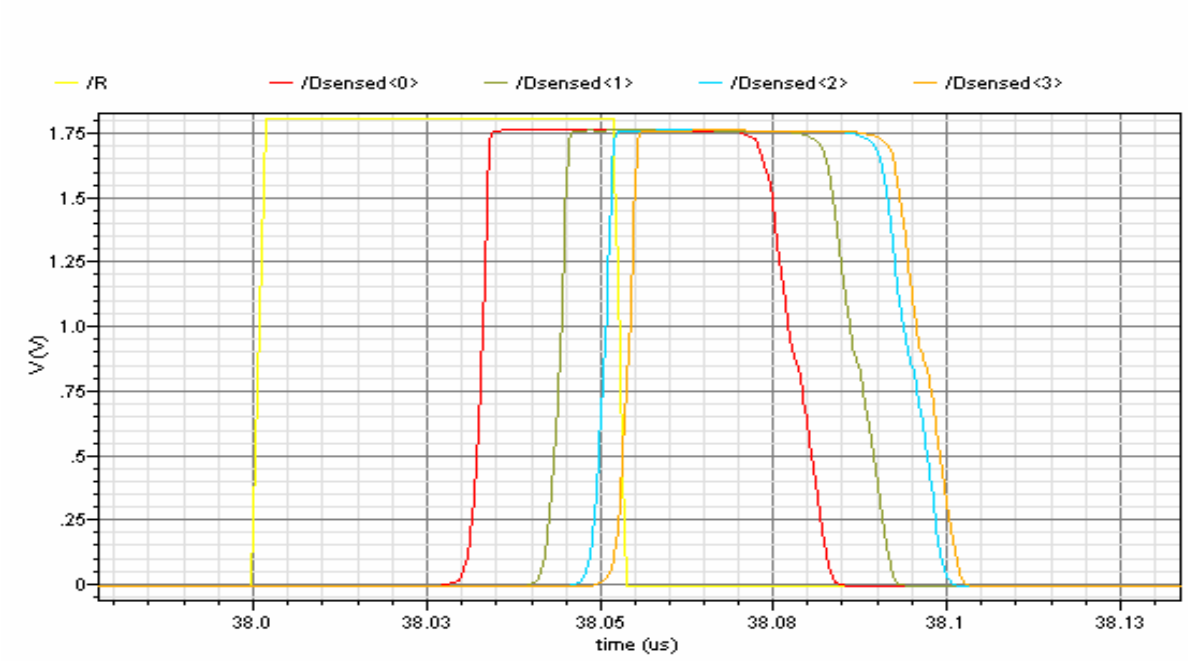
Above: Data sense amplifier internal signals: Current consumption ~75uA per bit.



Above: Signals from near/far/mid point along typical readout column.

Furthest cell data is sensed through 96 R-C load elements.
Interim cell data is sensed through 64 and 32 R-C load elements.
Nearest cell data is sensed directly at input to column amplifiers.

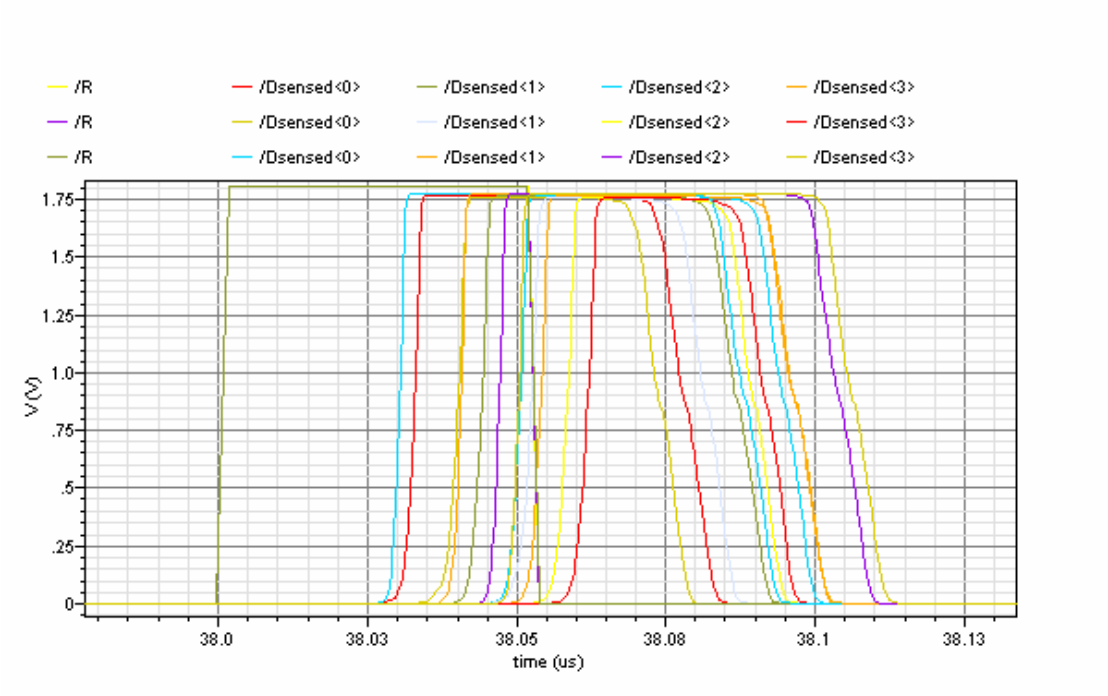
96 pixels = 4.8mm → one of the four pixel test structures on ASIC1



Above: Signals from near/far/mid point along very long readout column

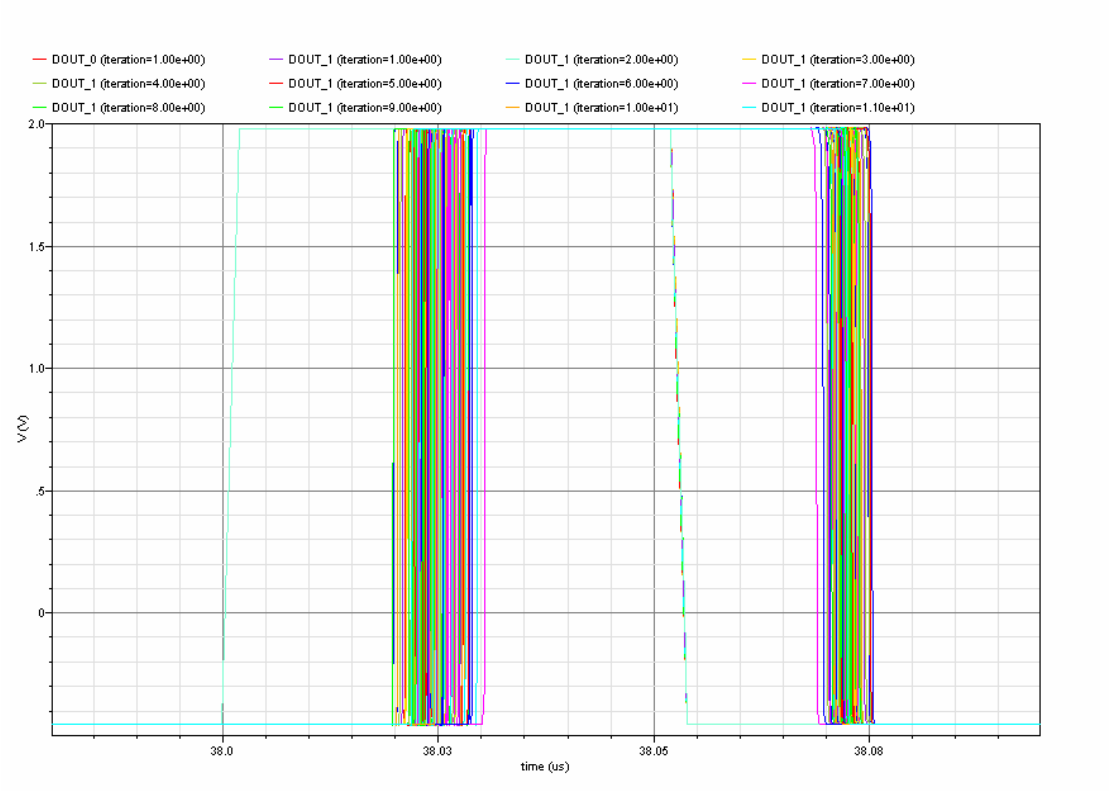
Furthest cell data is sensed through 384 R-C load elements.
Interim cell data is sensed through 256 and 128 R-C load elements.
Nearest cell data is sensed directly at input to column amplifiers.

385 pixels = 19.2mm → longer lines are possible at slower speeds



Above: Signals from near/far/mid point along very long readout column: Corners.

Data delay is between 30ns and 70ns.



Above: Sensed data output for mismatch monte-carlo statistical analysis
~15Mhz maximum readout rate is shown to be possible, (10Mhz is simulated)

SRAM overdrive strength

Basic SRAM cell is a pair of cross-coupled inverters that must be overpowered to write new data. Therefore it is important to understand the strength required to overpower a SRAM cell – if too many cells were written from the same data (ie the timestamp signal) they might become corrupted.

Process corner	**Driving cell (SRAM Write signal)**	
	x3_hv (2.5v)	**x3 (1.8v)**
SS	9	0
SF	8	0
TT	10	11
FS	10	10
FF	11	12

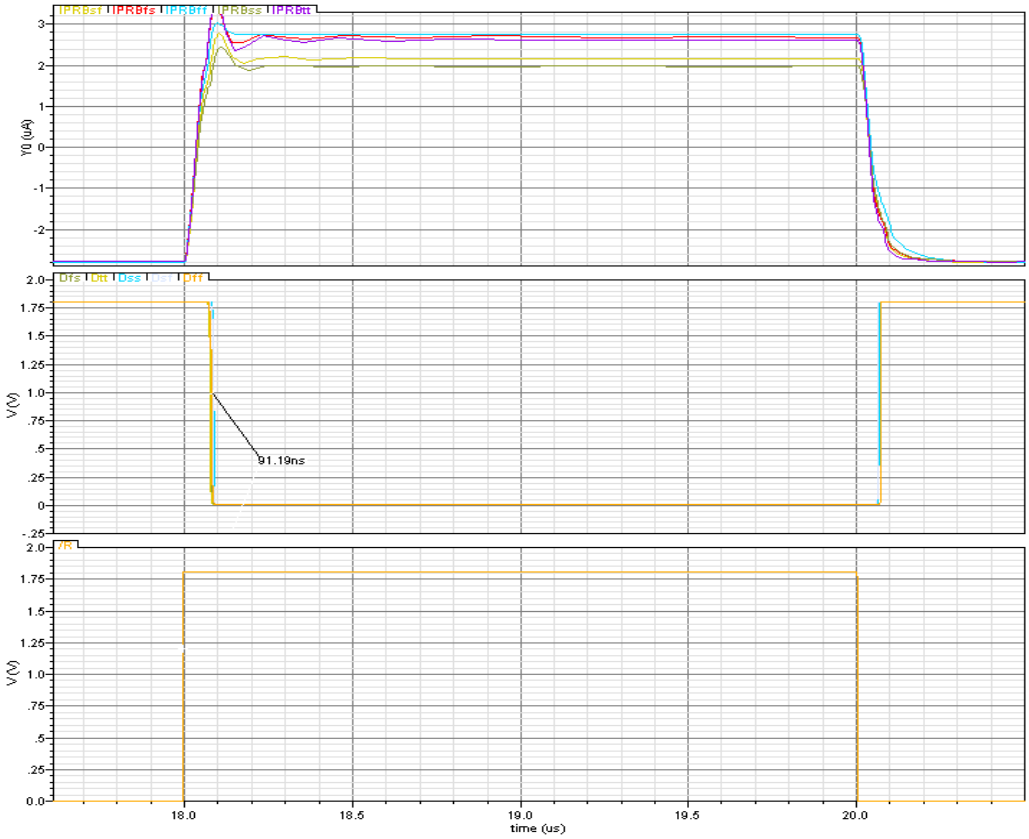
Above: Number of active SRAM cells that can be overpowered (successfully written). This data indicates the need to overdrive the write transistor for reliable SRAM operation in all process corners. This information is also important to ensure timecode buffers are not expected to drive too many rows, maximum 8. Monte-Carlo simulations also show that a data driver cell of weaker strength is insufficient and results in corrupted data.

Logic Simulation: Data Sense (reduced current)

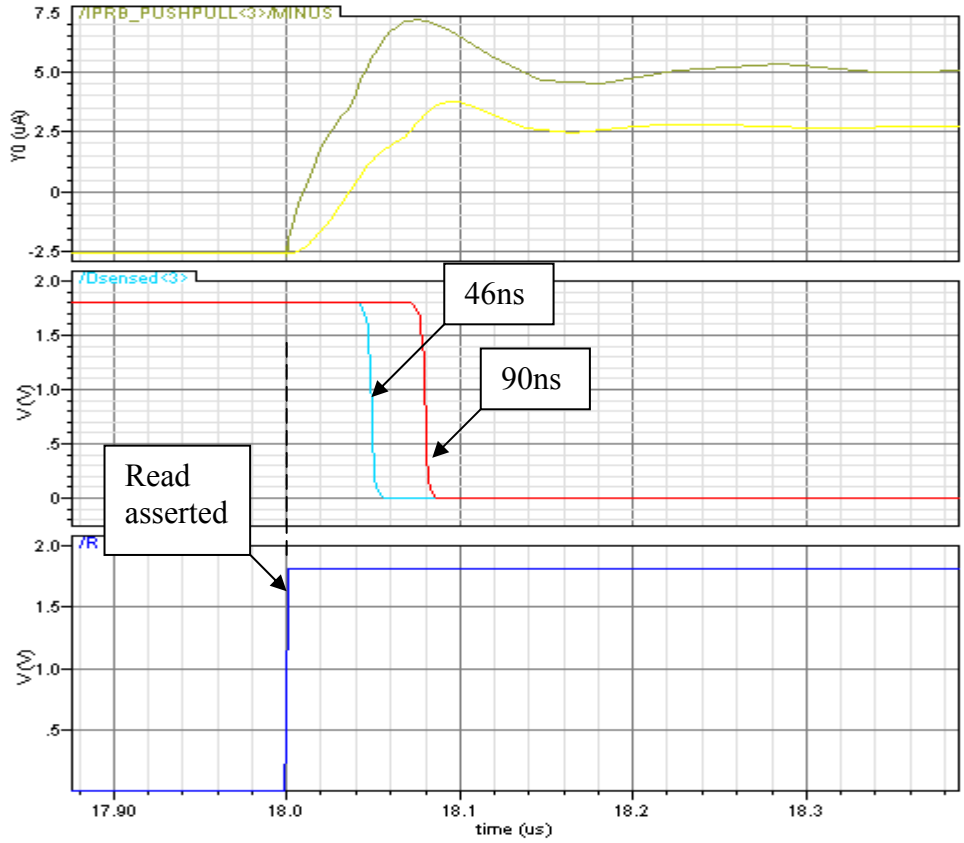
The circuit is operated in reduced current mode as detailed in the table below:

Data Column	Static current bias	10uA
	Mirrored 4:1 onto line as	2.5uA
SRAM cell	Current sink for ‘zero’	5uA
	Current sink for ‘one’	0
Data Sense	3.3v bias ref	13uA
	Total in sense amplifier	10.4uA

Results



Above: Process corner variations have negligible effect on the operation of this circuit. The bias circuit (which was originally designed for the OPIC project) includes stacked transistors to try to maintain consistent switching performance in different process corners; this seems to be working in this process also, and has not been modified.



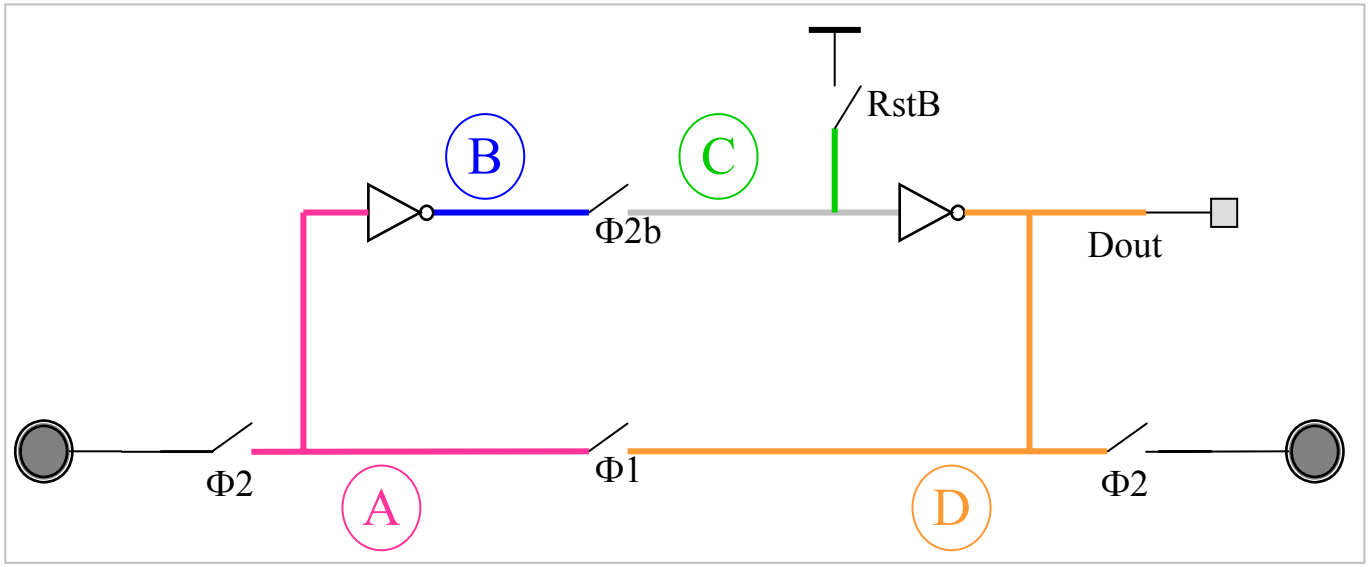
Above: Zoomed operation for cell accesses at the near and far ends of a 25mm data column. These simulations indicate a 5Mhz SRAM readout rate will possible from the full-sized ASIC2.

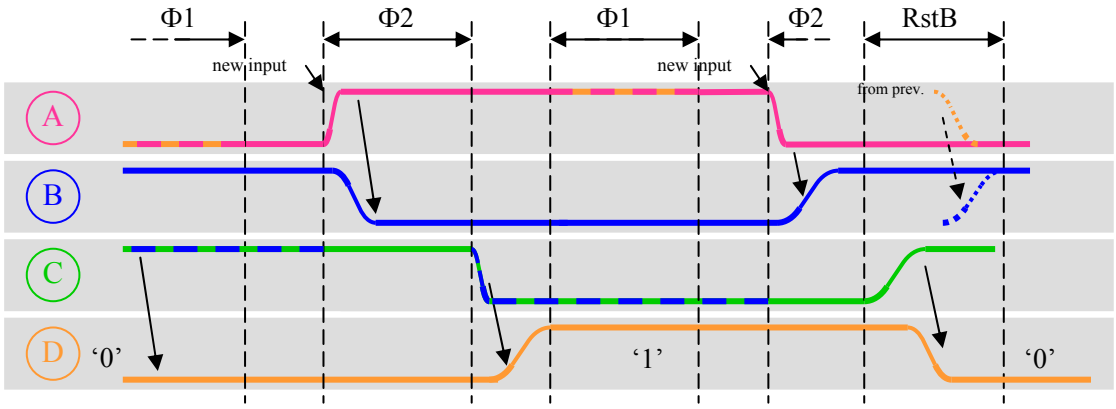
Monte carlo simulations for the furthest cell report the following delay statistics from assertion of read-enable to valid data appearing at the column base:

mu	82	ns
sd	17	ns
N	70	

Logic Simulation: SRAM shift register cell

The SRAM shift register cell and waveform timings are illustrated below.





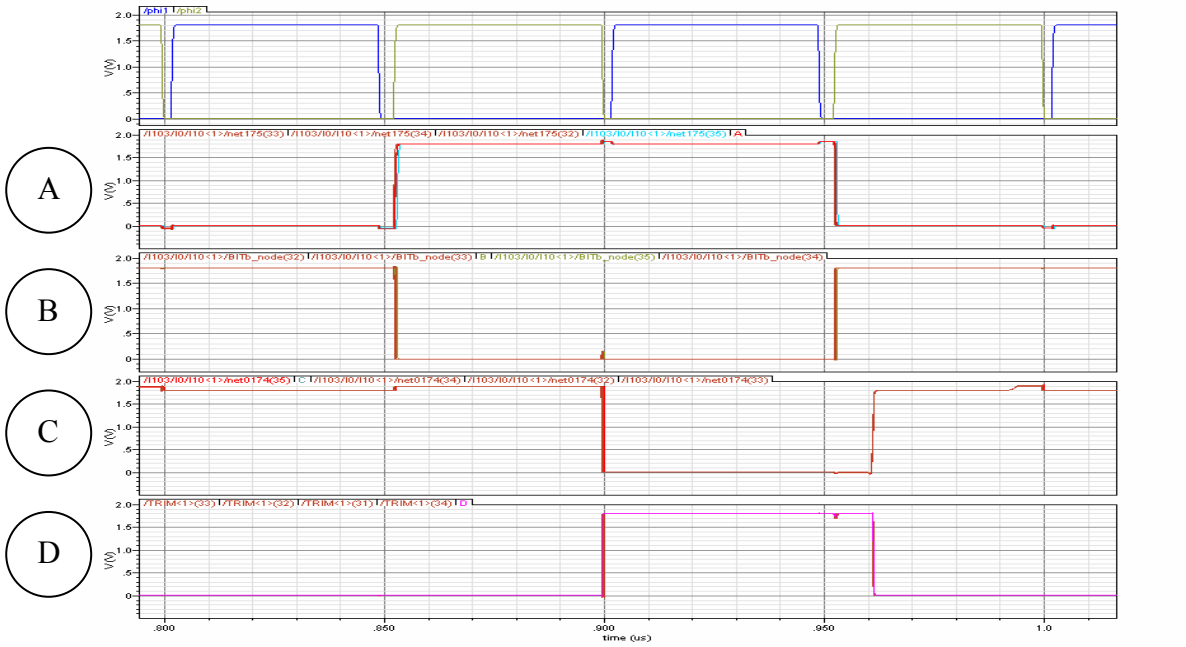
- During phi1 the SRAM cell holds its own state with two cross-coupled inverters.

- As phi2 switches on, node A is updated with the new input state (connected from previous SR cell through a phi2 switch). Node B updates accordingly. At the same time switch phi2b switches off, thus allowing the cell to drive its current state at the output without corruption from the new input.
    - RACE CONDITION
    - Node B is updated with the new value at node A. Node A must not be allowed to propagate through the inverter and phi2b switch to the node C (where it would corrupt the cell’s previous data).
    - The undesirable signal path includes an extra inverter with respect to the desirable signal sequence, therefore the race condition will result in the desired outcome.

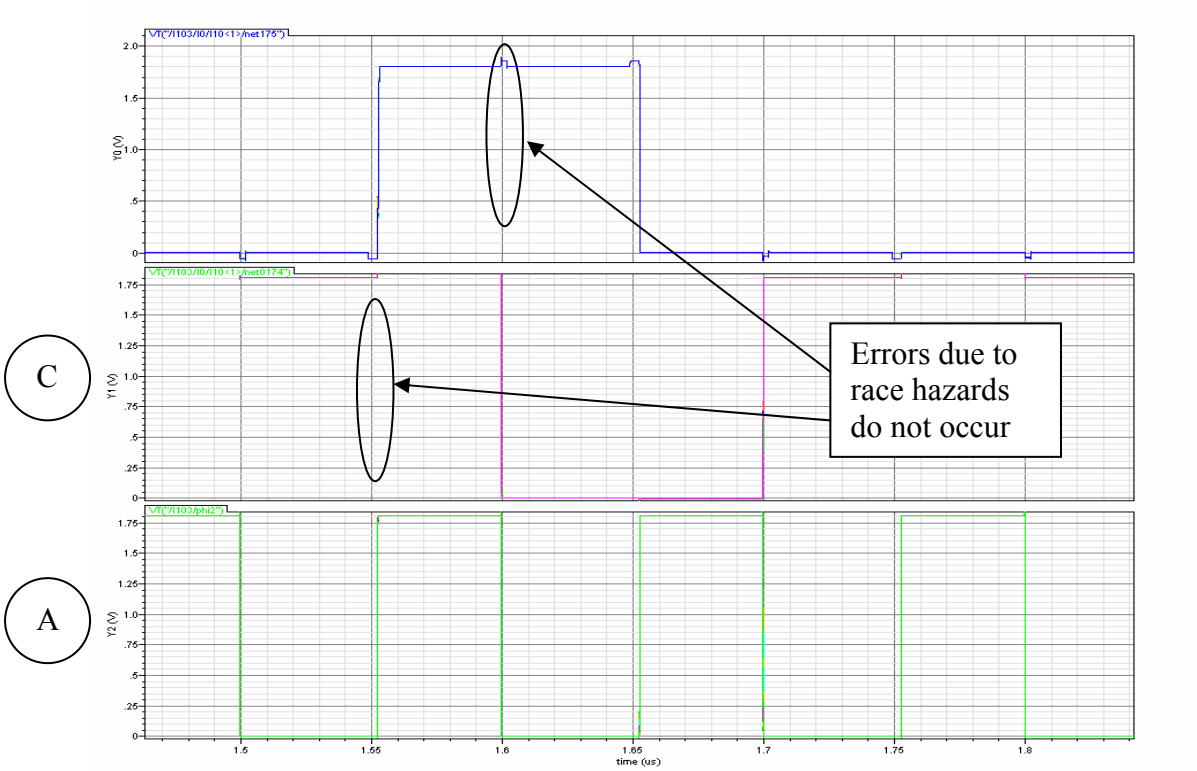
- During phi2 the SRAM cell drives its stored state to the next cell (connected to node D through a phi2 switch). The stored state is held on node C, comprising parasitic and gate capacitances of the inverter.

- As phi2 switches off the cell becomes isolated from its neighbour, and switch phi2b closes to update node C with the new state.
    - RACE CONDITION
    - Node C is updated with the new value at node B. Node C must not be allowed to propagate through the inverter and phi2 switch to the output node and into the next cell (where it would corrupt data at node A).
    - The undesirable signal path includes an extra inverter and phi2 switch with respect to the desirable signal sequence, therefore the race condition will result in the desired outcome.
    - An intermediate value may be sampled on the parasitic capacitance between two SR cells – this would not affect the correct operation of the SR cell.

- As phi1 switches on the cell completes the internal feedback loop and the new state is stored indefinitely (whilst powered).



Above: Corner simulations showing the internal nodes in the SR cell. The race conditions at both edges of phi2 behave correctly.



Above: Mismatch Monte Carlo (100 runs) to check performance is robust in race conditions.

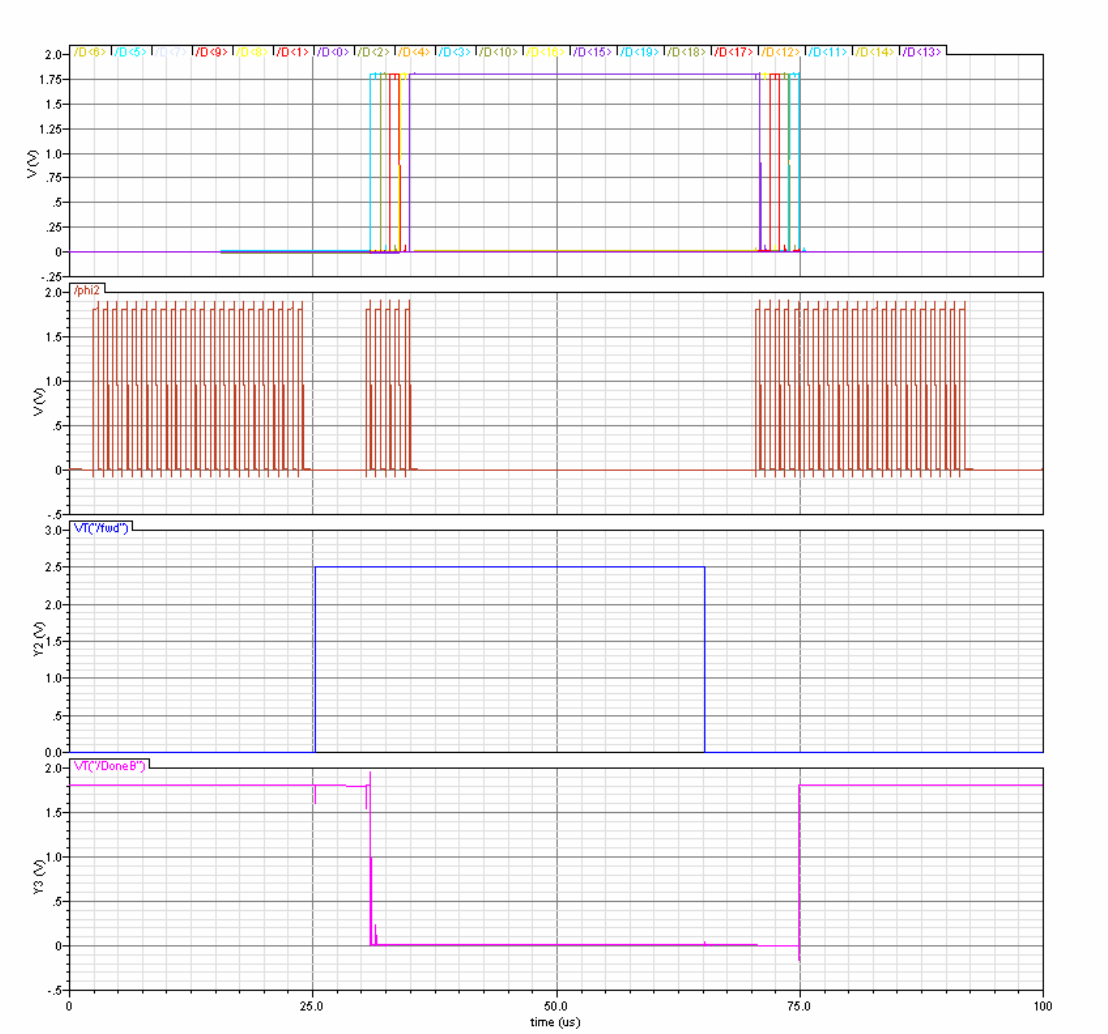
Logic simulation: Bi-Directional SRAM shift register

Simulation top-level = “sim_sram_shift20”

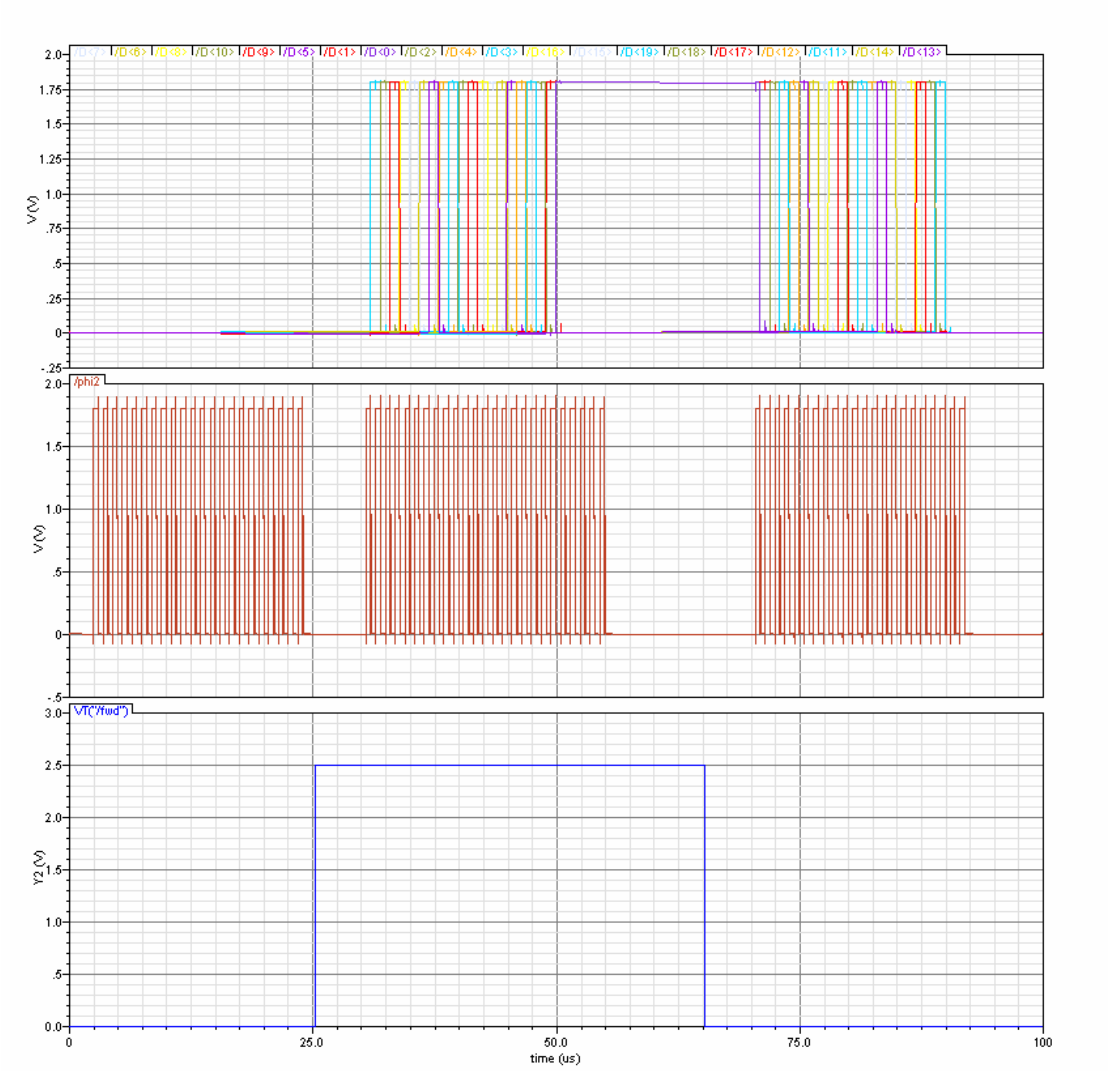
The bidirectional shift register generates the read and write pointers for the SRAM memory bank. After power-on it could be in an unknown state, so the initialisation routine requires the shift register to be clocked in the backwards direction for 20 clocks to ensure all elements are programmed to zero.

In normal operation the register is clocked once in the forward direction to initialise, and then once again for each register that must be written. During readout mode the register is clocked once in backward direction to initialise, then combinational logic enables each row to access each of its valid registers sequentially until the column read is complete.

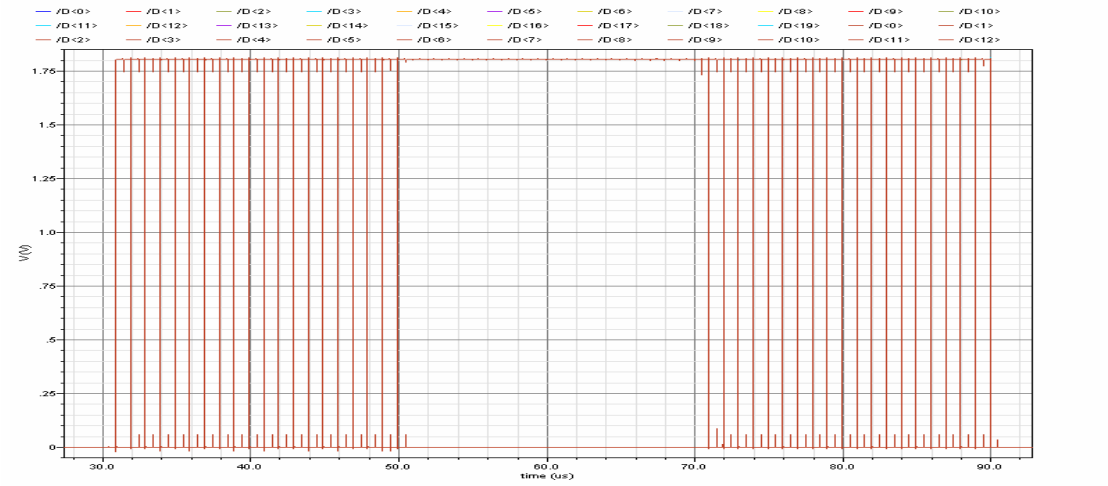
If >19 forward (“write”) clocks are received the register outputs a “full” wired-or signal, and will correctly read back the first 19 registers that were written.



Above: Typical operation, showing 5 clocks in the forward direction, then 25 clocks in the reverse direction, of which the first 5 act upon the shift register and replay those registers that were written-to. The shift register thereafter ignores any further phi2 clocks and sets the “Done” signal (combinational NOR of all 20 bits).



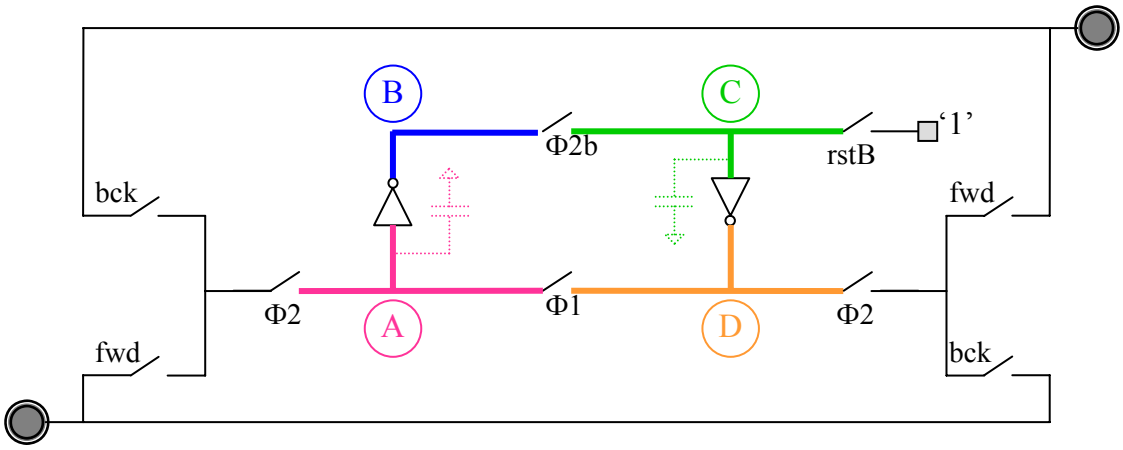
Above: Overflow condition: 25 hits are simulated, note that reg 0 remains active beyond the 20th hit – this is used to generate the overflow signal, and ensures that when clocked backwards the shift register has not lost its token so that readout of the other 19 registers is achieved (as illustrated).



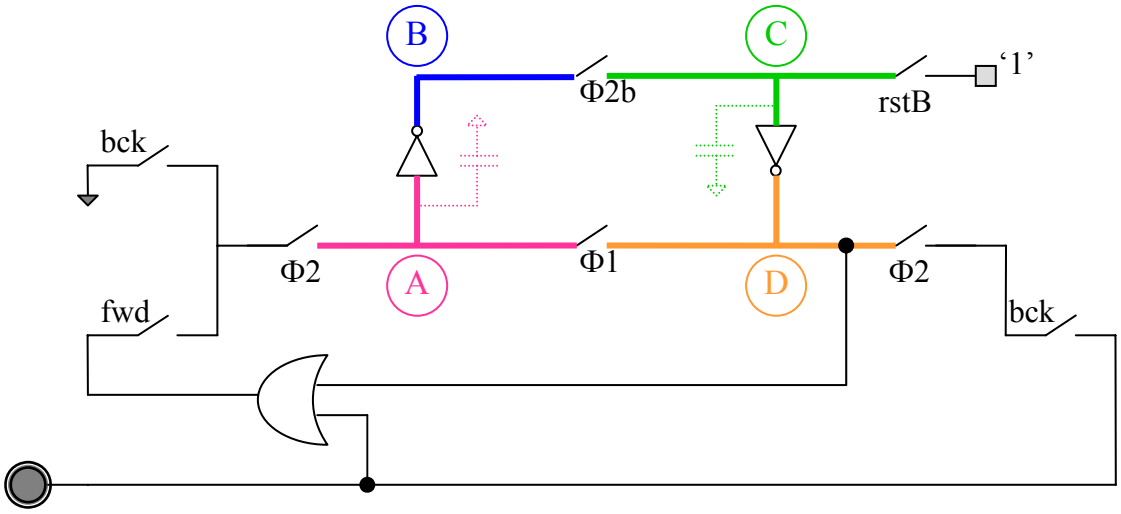
Above: Correct operation verified in process corners SS,TT,FF.

Logic simulation: Bi-Directional SRAM cells

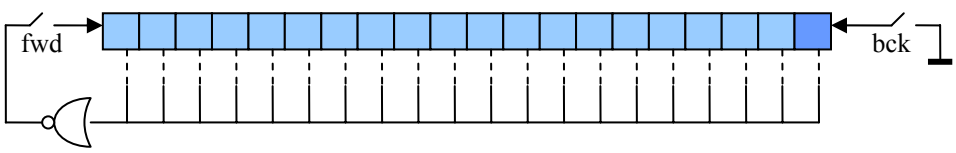
The standard SRAM shift register cell is adapted to make it bi-directional as illustrated below.



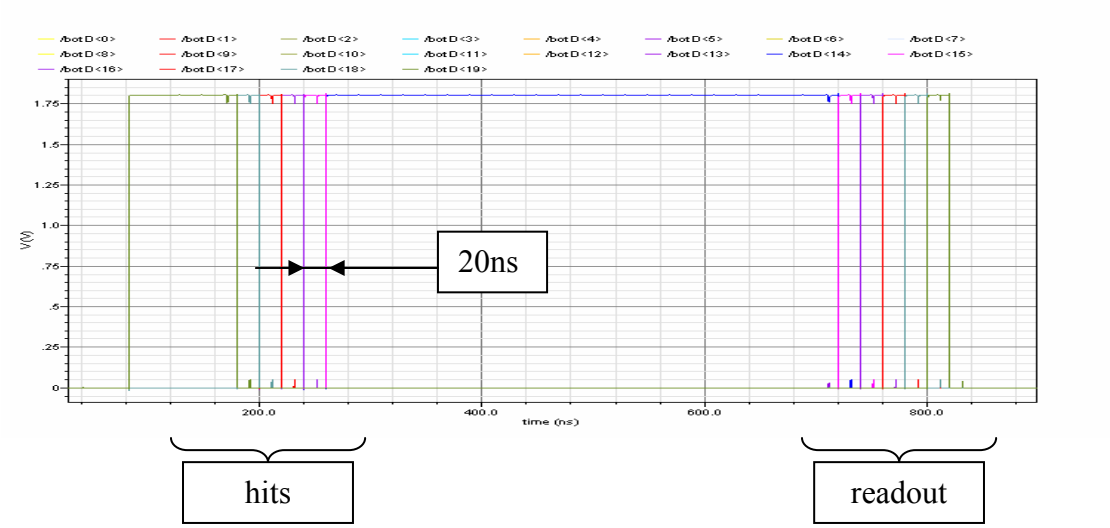
The last cell must be modified such that it will hold the token rather than lose it if clocked again (robust operation in overflow conditions) and also ensure that a '0' is driven as the cell input when in backwards mode.



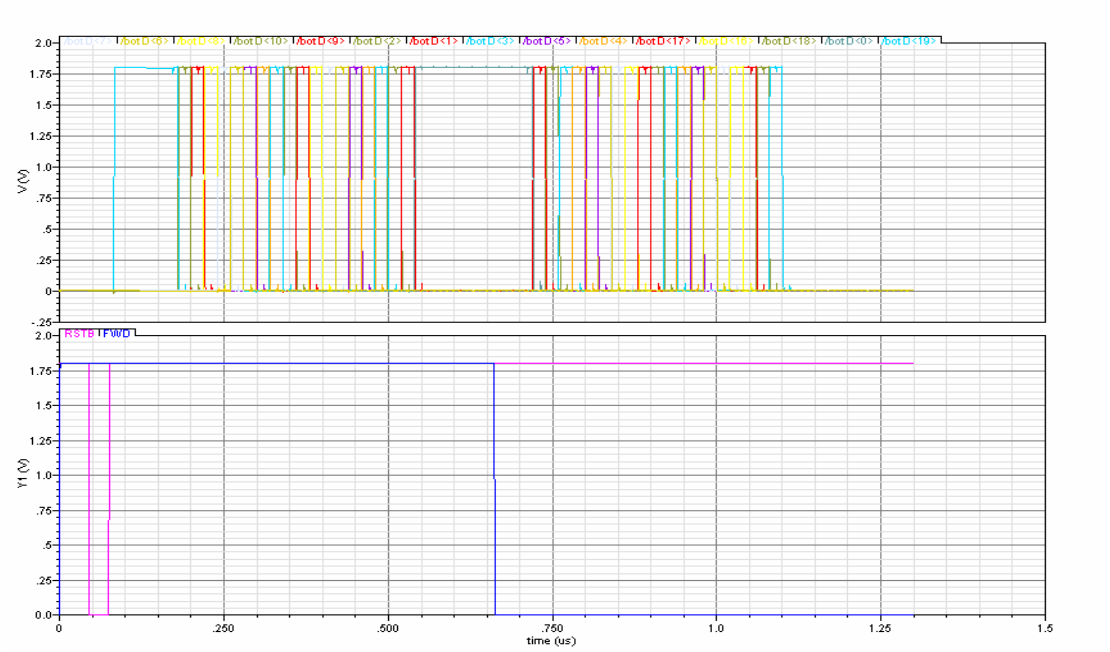
The "bi-directional" and "endstop" shift register cells are arranged to form a 20-element shift register. A logical NOR generates the SR input in the fwd mode such that either after reset or one forward clock cycle when empty, the register is reset to the default condition 10000000000000000000.



Results Waveforms



Above: Typical operation involving 5 consecutive hits (ie all in the same 42-pixel section). These simulations are run at ~50Mhz to check operation is possible at the target 150ns bunch-crossing rate.



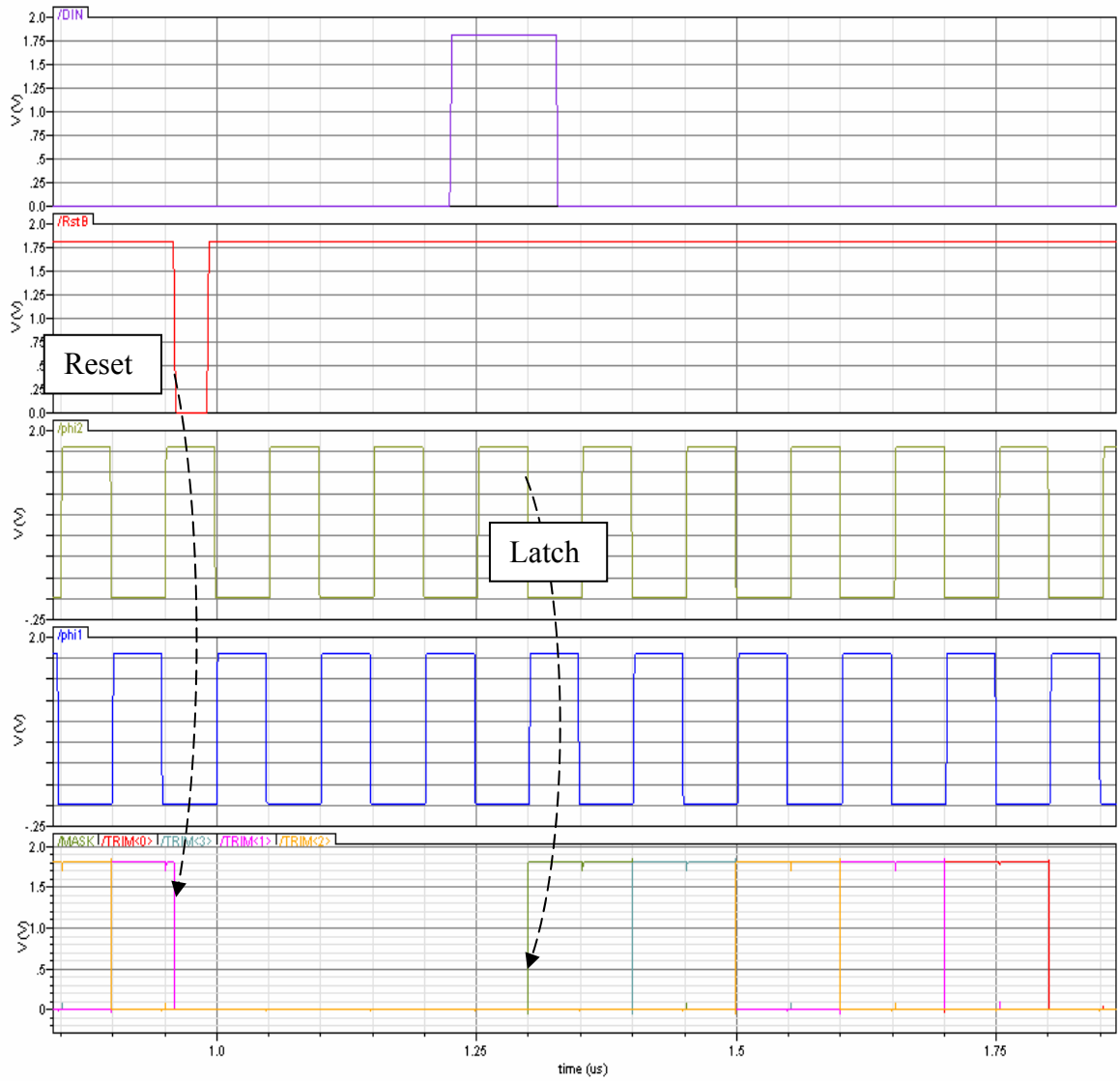
Above: The overflow condition is verified – the register is clocked for 25 hits. The circuit can be seen to hold the token at position 0 and then assert each enable line for all the registers back to 19.

Logic simulation: Latch-Hold circuits

Logic simulation: Mask & Config programming

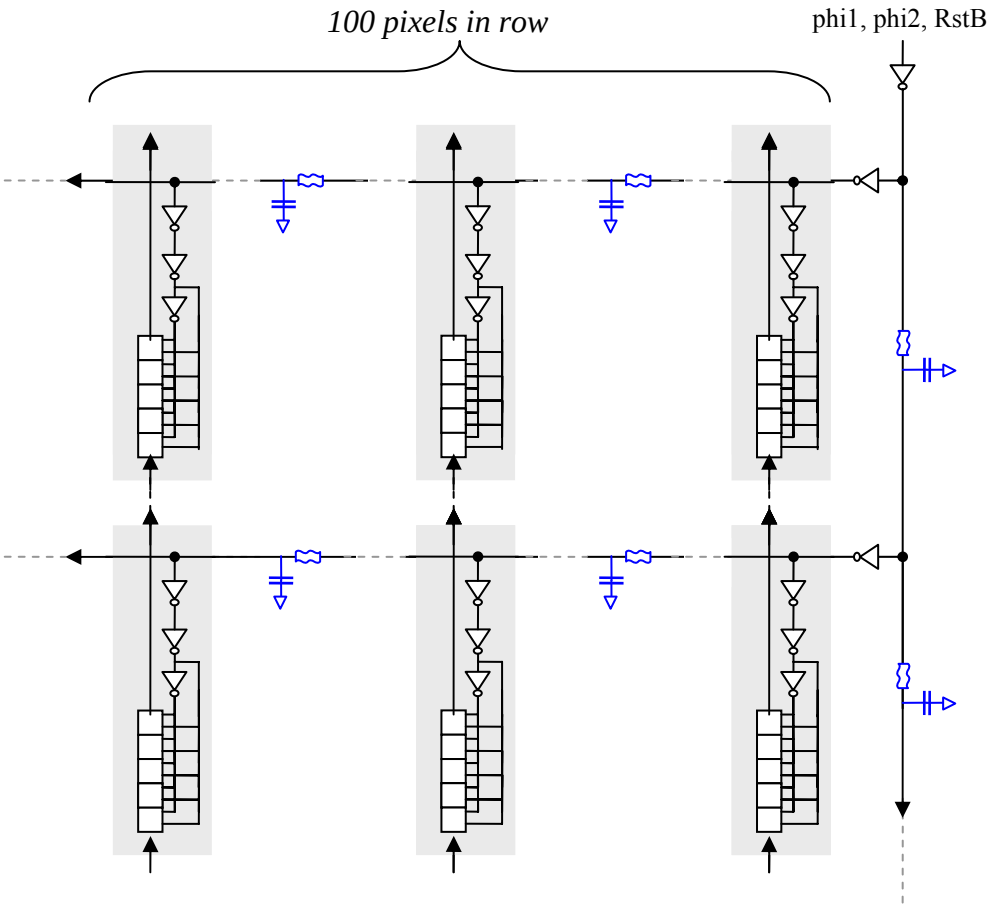
Each pixel contains a 5-bit SRAM shift register, arranged such that each column forms a long shift register. Global clocks are distributed along each row. Pixels buffer and generate the compliment clocks internally to ensure clean clock waveforms for the SRAM shift register cells.

Pixel operation



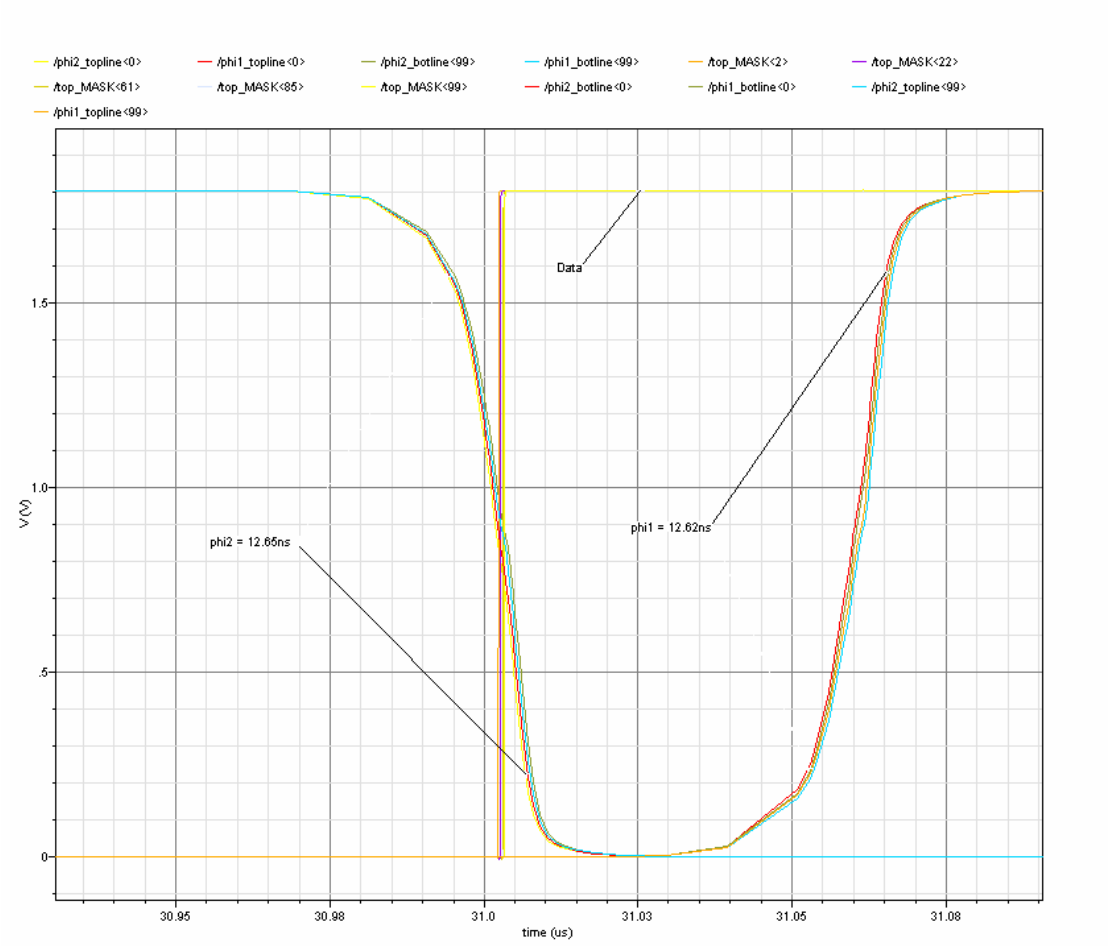
Reset occurs during phi2. Input data is latched on the falling edge of phi2. In the example above a single 'one' is shifted through the SR. In the real system, data is shifted in as TRIM-LSB first with the MASK as the final bit. The full string of 5-bit codes in this order are shifted through the column.

System operation

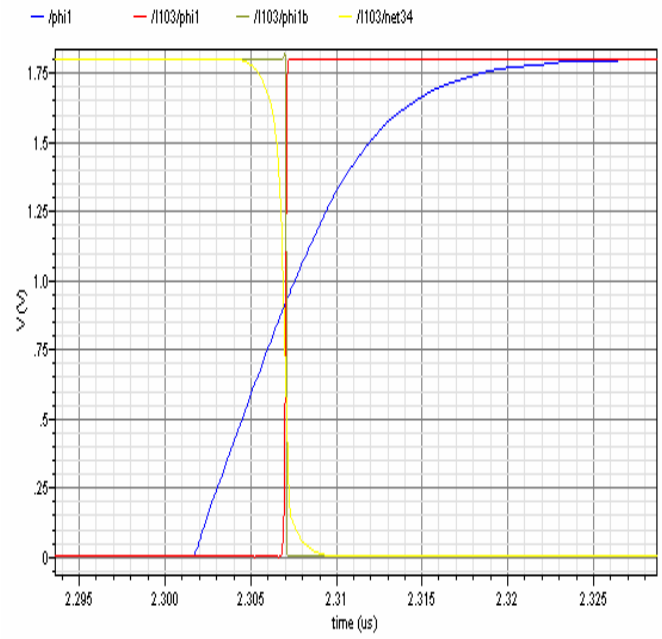


Above: Diagram representation of the phi1/phi2 non-overlapping clock distribution simulation for a section the mask shift register. Each pixel generates clk and clkb from the column clock to ensure edges are quick and a true-complement to ensure correct operation of the SR cells. Distributed RC models for the long signal lines are inserted to model a row of 100 pixels. Clock drivers of strength x4 are used to drive the long row and column lines.

The triple-inverter clock buffering scheme is used to ensure local edges are fast and true compliment. Three buffers per pixel (x2 for two clocks) will all switch in the same time window, thus the current drawn could be significant during mask programming. Care must be taken to ensure power tracking is wide enough to minimise power droop. Since these cells are not used during normal operation the power net can be shared with other circuits.

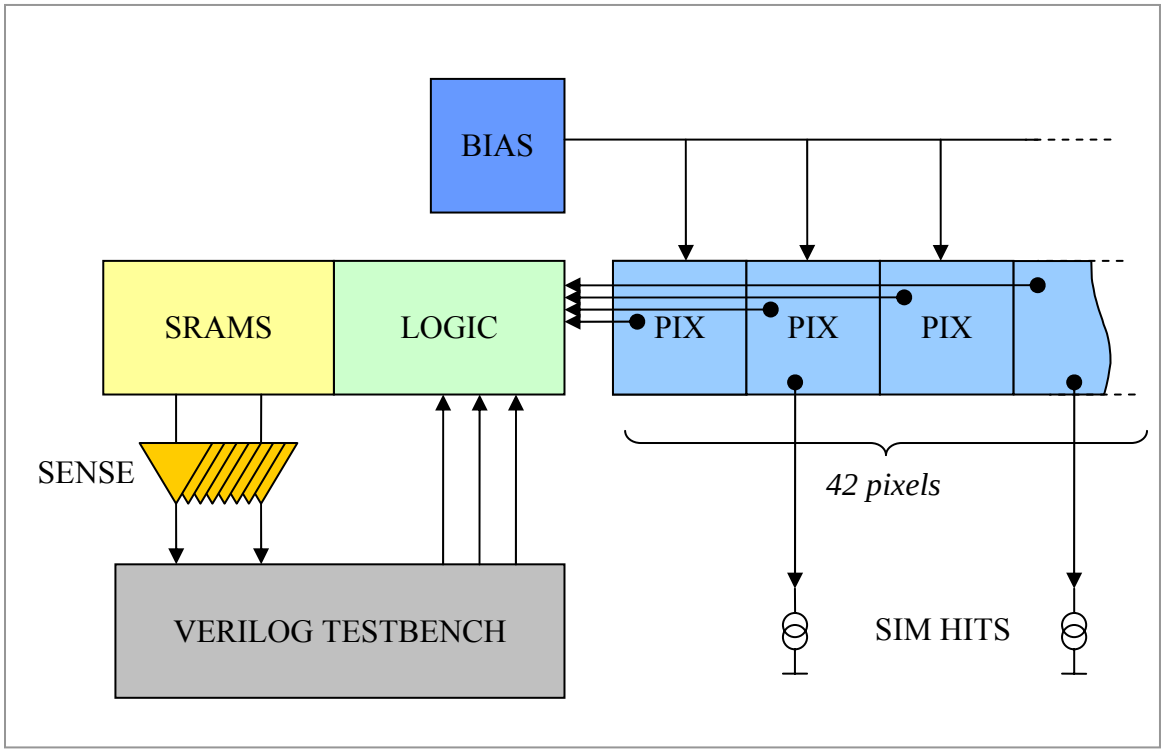


Above: Operation is demonstrated at 1Mhz for 100 parallel shift registers. 50ns separation of the non-overlapping clocks ensures correct operation. Edges of around 12ns are seen on the global clocks (these are buffered and the complement generated in-pixel to ensure good clock integrity).



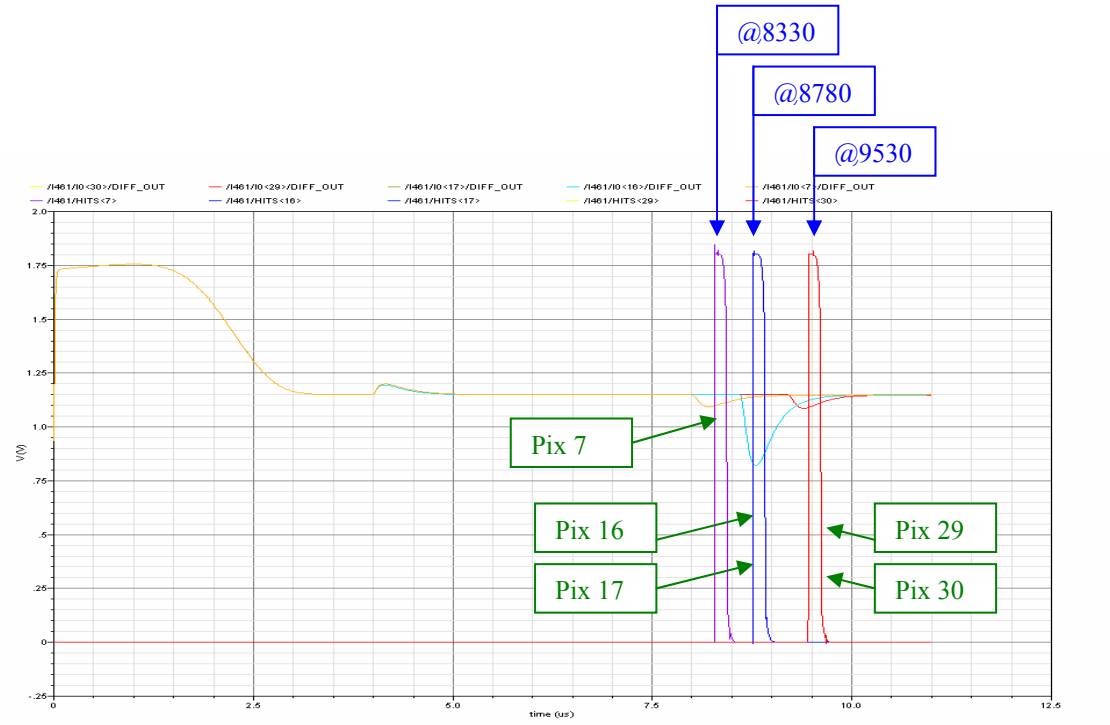
Left: Local clock buffering ensures the potentially slow row clock signals are fast and true compliment.

Logic simulation: Full pixel 'slice' simulation



The diagram above depicts the slice simulation including 42 pixels, the master row controller, 19 SRAM registers and data sense amplifiers. This simulation omits any mask or trim programming to reduce the simulation time (all channels are enabled).

The testbench initialises the logic and waits for the analog to settle/power-up. The logic is then sequenced as per bunch-train operation, and simulated hits are applied to different pixels at different times. The logic detects and stores the hit location and time, which is then read out and displayed in the simulation log after the bunch train is complete (16 crossings).




```
Mask register = 11111100000011111111111111111111111111111111 at initialisation
Hit pattern   = 00000000000000000000000000000000000000000000 at initialisation
Mask programming omitted.
Clear & Initialise the SR control registers
SR Clear complete
SR Initialisation complete
Latch current Hit Pattern...                7730
Latch current Hit Pattern...                7880
Latch current Hit Pattern...                8030
Latch current Hit Pattern...                8180
Latch current Hit Pattern...                8330
Latch current Hit Pattern...                8480
Latch current Hit Pattern...                8630
Latch current Hit Pattern...                8780
Latch current Hit Pattern...                8930
Latch current Hit Pattern...                9080
Latch current Hit Pattern...                9230
Latch current Hit Pattern...                9380
Latch current Hit Pattern...                9530
Latch current Hit Pattern...                9680
Latch current Hit Pattern...                9830
End of bunch train.
Readout phase:
SR Initialisation complete
     >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000001101
     >>           BANK = 010
     >>           PATT = 000001
     >>           DATA = 00000000011010100000001
     <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
     >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000001101
     >>           BANK = 000
     >>           PATT = 100000
     >>           DATA = 00000000011010001000000
     <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
     >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000001000
     >>           BANK = 101
     >>           PATT = 110000
     >>           DATA = 00000000010001011100000
     <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
     >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000000101
     >>           BANK = 100
     >>           PATT = 000010
     >>           DATA = 00000000001011000000010
     <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0A 0x0B 0x0C 0x0D 0x0E 0x0F

0x0D — Pix 30

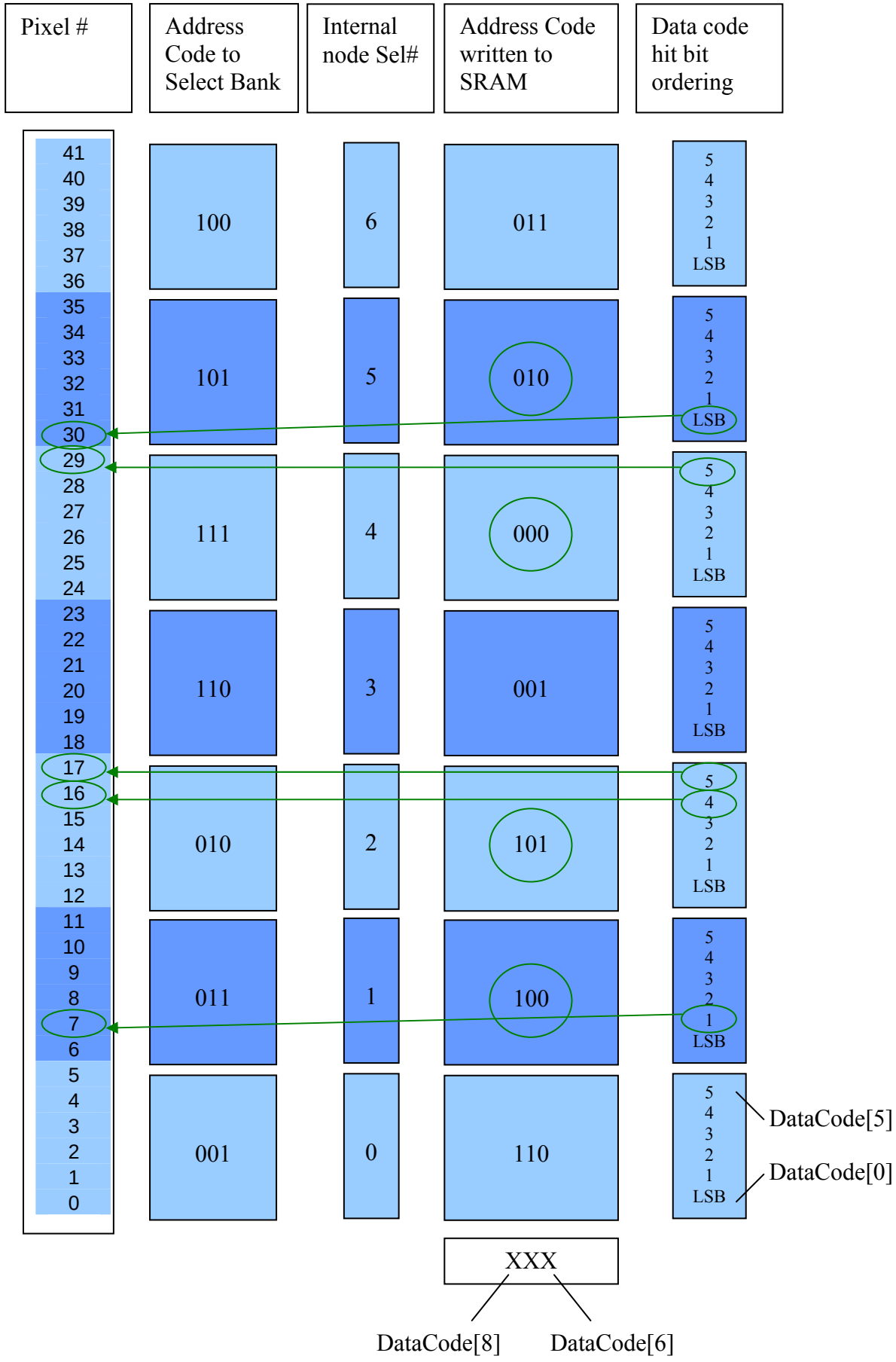
0x0D — Pix 29

0x08 — Pix 16

Pix 17

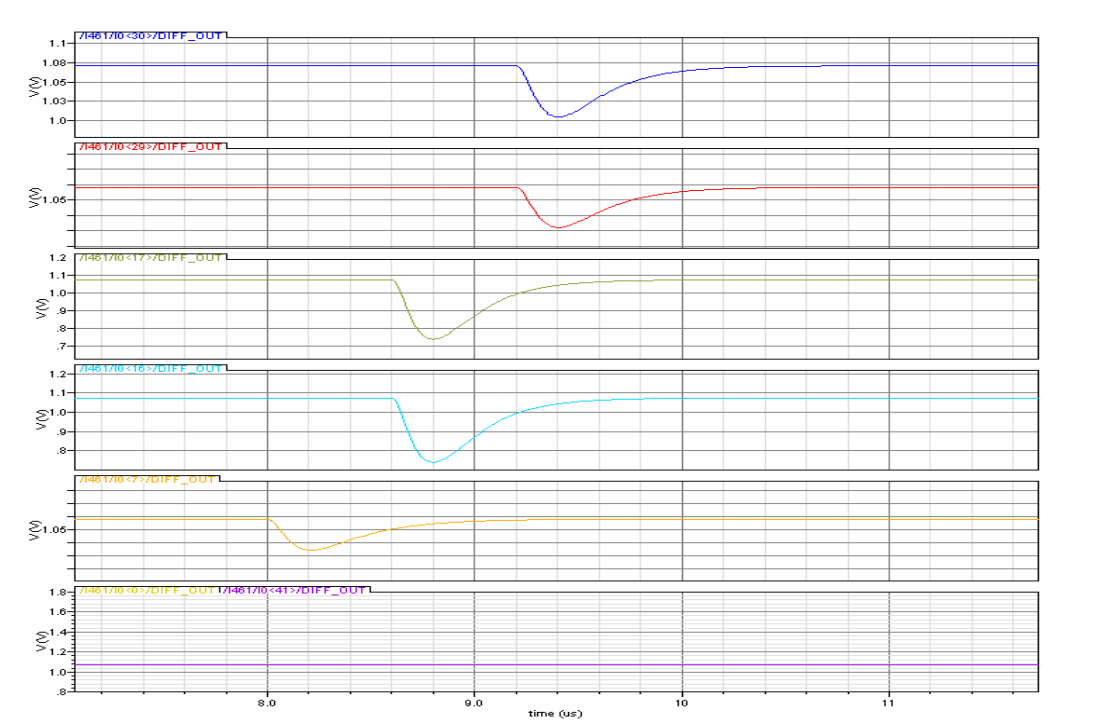
0x05 — Pix 7

TT ✓
SS ✓
FF ✓


Pixel #
Address Code to Select Bank
Internal node Sel#
Address Code written to SRAM
Data code hit bit ordering
41 40 39 38 37 36
35 34 33 32 31 30
29 28 27 26 25 24
23 22 21 20 19 18
17 16 15 14 13 12
11 10 9 8 7 6
5 4 3 2 1 0
100
101
111
110
010
011
001
6
5
4
3
2
1
0
011
010
000
001
101
100
110
5 4 3 2 1 LSB
DataCode[5]
DataCode[0]
XXX
DataCode[8]
DataCode[6]

Logic Simulation: Full Pixel Slice Including Masking

This is a subsequent revision of the simulation above to include the mask and trim programming control. Analog hits are generated in pixels 7, 16, 17, 29 and 30 as shown below. The mask register is set to deactivate pixels 6 → 11 inclusive.



Results Transcript

The corresponding hit data is seen in the excerpt from the simulation log below. The hit on pixel 7 has correctly been masked. Note also that the hit on pixel 30 appears twice in the transcript – this is a double hit (see below).

```
Readout phase:
SR Initialisation complete
     >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000001100
     >>           BANK = 010
     >>           PATT = 000001
     >>           DATA = 00000000011000010000001
     <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
     >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000001011
     >>           BANK = 010
     >>           PATT = 000001
     >>           DATA = 00000000010110100000001
     <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
     >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000001011
     >>           BANK = 000
     >>           PATT = 100000
     >>           DATA = 00000000010110000100000
     <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
     >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000000111
     >>           BANK = 101
     >>           PATT = 110000
     >>           DATA = 00000000000111101110000
     <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
  -
  -
  -
```

Pixel 30 at time 1100

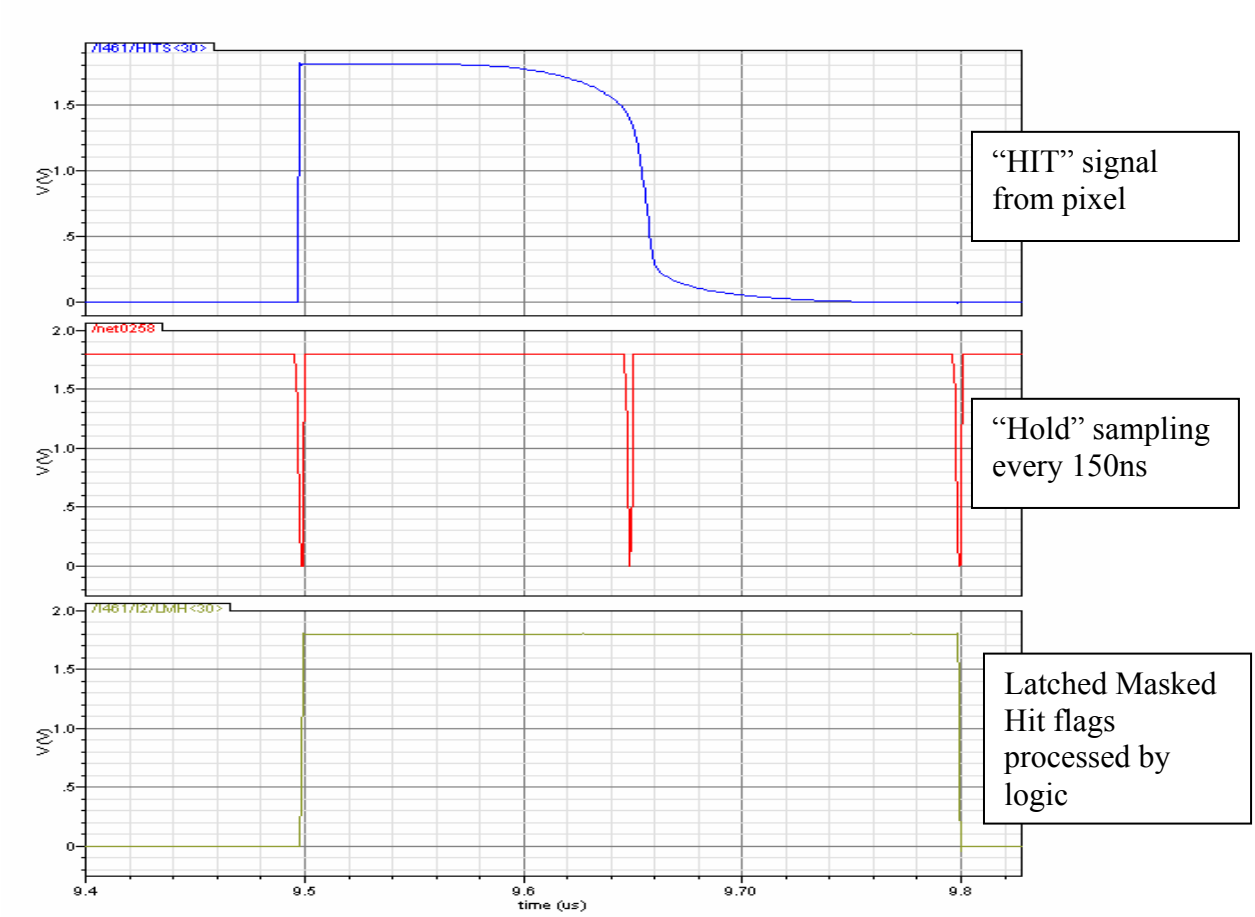
Pixel 30 at time 1011

Double Hit!!!

Pixel 29 at time 1011

Pixels 16&17 at time 1100

The double hit seen in the simulation deserves further investigation:



Looking at the internal signals for pixel 30 shows that the rising edge of the hit signal occurs at approximately the same time as the "hold" sampling. The duration of the hit signal is measured to be 157ns, which means it is sampled as active (high) on two consecutive occasions. This shows how a double-hit might occur.

Whilst it is interesting that this did occur in simulation, we consider the probability of this occurring to be low, since it depends on so many variables (some of which are artificially generated in the simulation). A full analysis would involve much analysis from many device and circuit simulations and is probably impractical! (Discussion welcome)

If double-hits are seen during testing, this might indicate that the biasing of the monostables is too high. In this case the "mso_bias" that controls the timing of the hit pulse can be adjusted until double-hits cease to appear. Note however that adjusting the bias too low would make the "HIT" signal pulse much less than 150ns, in which case hits could be dropped altogether if a similar set of timing circumstances occur.

```
End of bunch train.
Readout phase:
SR Initialisation complete
     >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000001100
     >>           BANK = 010
     >>           PATT = 000001
     >>           DATA = 00000000011000010000001
     <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
     >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000001100
     >>           BANK = 000
     >>           PATT = 100000
     >>           DATA = 00000000011000000100000
     <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
     >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000001011
     >>           BANK = 000
     >>           PATT = 100000
     >>           DATA = 00000000010110000100000
     <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
     >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000000111
     >>           BANK = 101
     >>           PATT = 110000
     >>           DATA = 00000000001111011100000
     <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
 -
 -
 -
 -
 -
 -
```

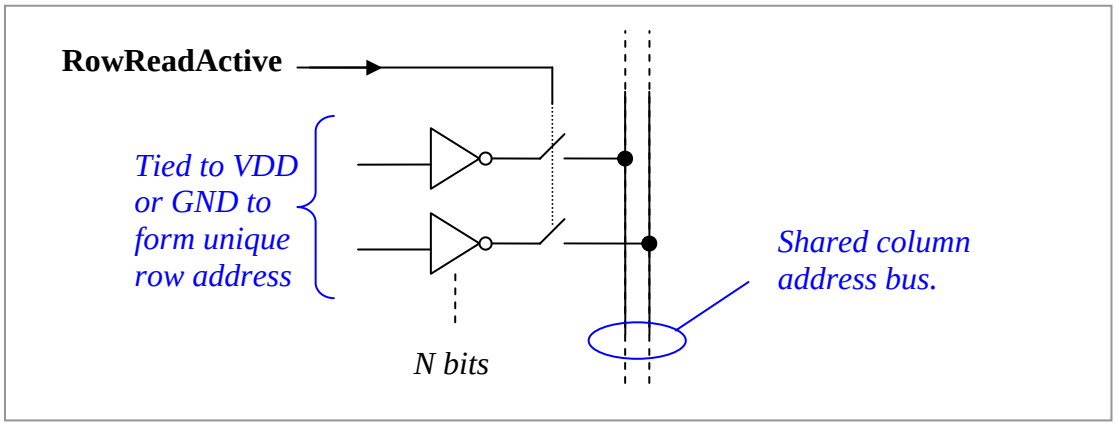
SS

```
Readout phase:
SR Initialisation complete
     >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000001100
     >>           BANK = 010
     >>           PATT = 000001
     >>           DATA = 00000000011000010000001
     <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
     >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000001011
     >>           BANK = 010
     >>           PATT = 000001
     >>           DATA = 00000000010110100000001
     <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
     >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000001011
     >>           BANK = 000
     >>           PATT = 100000
     >>           DATA = 00000000010110000100000
     <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
     >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000000111
     >>           BANK = 101
     >>           PATT = 110000
     >>           DATA = 00000000001111011100000
     <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
 -
 -
 -
 -
 -
 -
```

FF

Logic Simulation: Row Encoder

The basic structure of the row encoder is illustrated below. The function of this block is to report at the column base the row address of the cell that is currently being read. To minimise re-design effort for ASIC2 the full 9 bits are implemented, although only 7 are strictly required for the ASIC1 test structure. The row addresses are assigned in a GRAYCODE order.



The active cell drives the column bus with its own unique code. To avoid large current surges the inverter cells are current limited; this makes the column bus rise/fall time longer, but high-speed operation during the readout phase is not necessary.

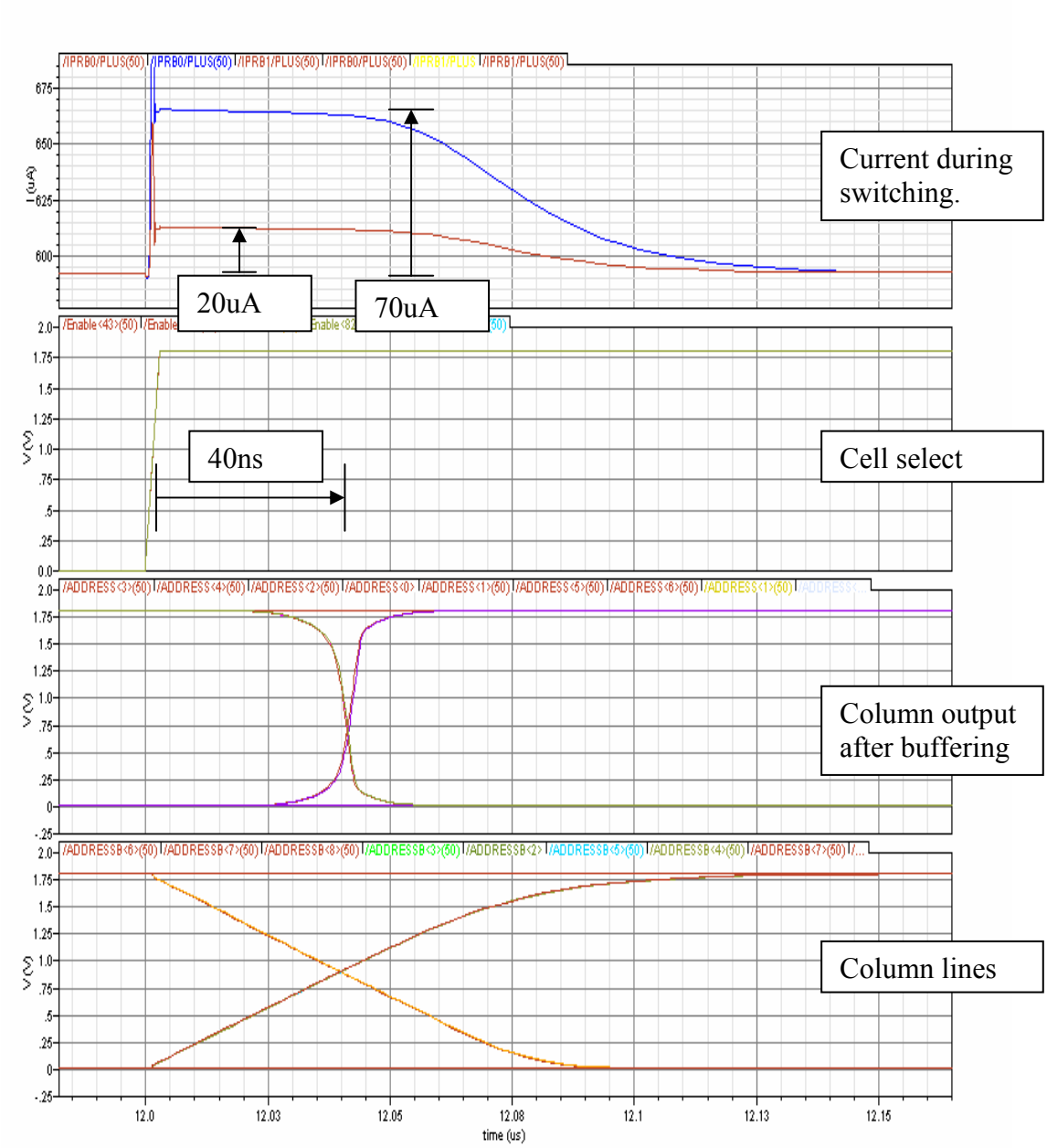
Simulations

A row-encoder of 84 cells is simulated.

Two RowReadActive signals are applied in sequence to the near and far ends to compare signal timing. An R-C model is used between each row encoder cell to model the likely real conditions. Current is monitored during switching.

A Verilog stimulus is used to individually excite each of the 84 enable (RowReadActive) signals and print to screen the address code that is seen at the base of the column. This text file is imported into excel and processed to check that no repeated codes are seen in the 84-code set.

Results Waveforms



Above: Row encoder outputs (for near and far cells) are shown at the column base. The current is limited to 20uA per switching signal (this may be adjusted by external bias current). The response time is effectively the time taken to charge the parasitic line capacitance with a DC current of 20uA. The line behaves as a single capacitor, not a transmission line, therefore no difference is seen between the response from the nearest/furthest cell.

Readout at 1Mhz could easily be achieved with this row encoder.

Hit 0

Hit 1

```
Address = 001111010
Address = 001111011
Address = 001111001
Address = 001111000
Address = 001101000
Address = 001101001
Address = 001101011
Address = 001101010
Address = 001101110
Address = 001101111
Address = 001101101
Address = 001101100
Address = 001100100
Address = 001100101
Address = 001100111
Address = 001100110
Address = 001100010
Address = 001100011
Address = 001100001
Address = 001100000
Address = 000100000
Address = 000100001
Address = 000100011
Address = 000100010
Address = 000100110
Address = 000100111
Address = 000100101
Address = 000100100
Address = 000101100
Address = 000101101
Address = 000101111
Address = 000101110
Address = 000101010
Address = 000101011
Address = 000101001
Address = 000101000
Address = 000111000
Address = 000111001
Address = 000111011
Address = 000111010
Address = 000111110
Address = 000111111
Address = 000111101
Address = 000111100
Address = 000110100
Address = 000110101
Address = 000110111
Address = 000110110
Address = 000110010
Address = 000110011
Address = 000110001
Address = 000110000
Address = 000010000
Address = 000010001
Address = 000010011
Address = 000010010
Address = 000010110
Address = 000010111
Address = 000010101
Address = 000010100
Address = 000011100
Address = 000011101
Address = 000011111
Address = 000011110
Address = 000011010
Address = 000011011
Address = 000011001
Address = 000011000
Address = 000001000
Address = 000001001
Address = 000001011
Address = 000001010
Address = 000001110
Address = 000001111
Address = 000001101
Address = 000001100
Address = 000000100
Address = 000000101
Address = 000000111
Address = 000000110
Address = 000000010
Address = 000000011
Address = 000000001
Address = 000000000
```

Hit 41

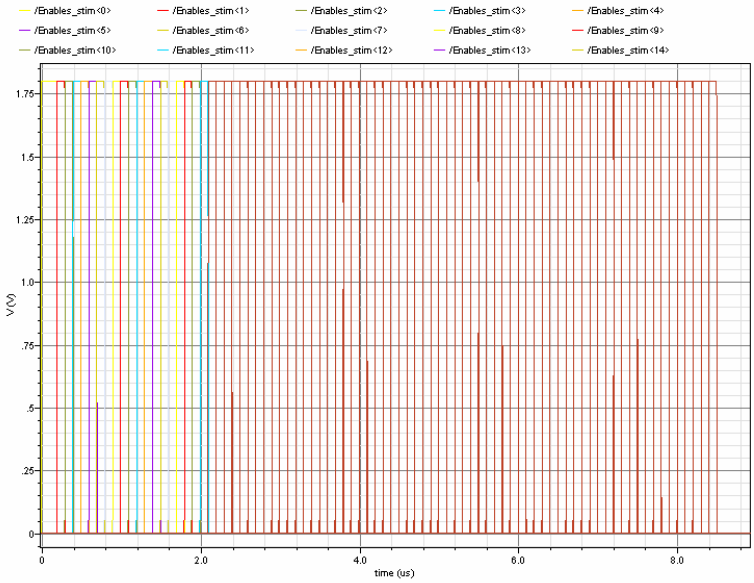
Hit 83

Left: Mixed-mode simulation output log excerpt.

Importing this data into excel, converting to decimal and sorting confirms there are no repeated codes.

Note that the bus ordering is such that the enables input bus' MSB corresponds to the address 0.

Below: Each enable signal is raised in turn to generate the data output shown on the left.

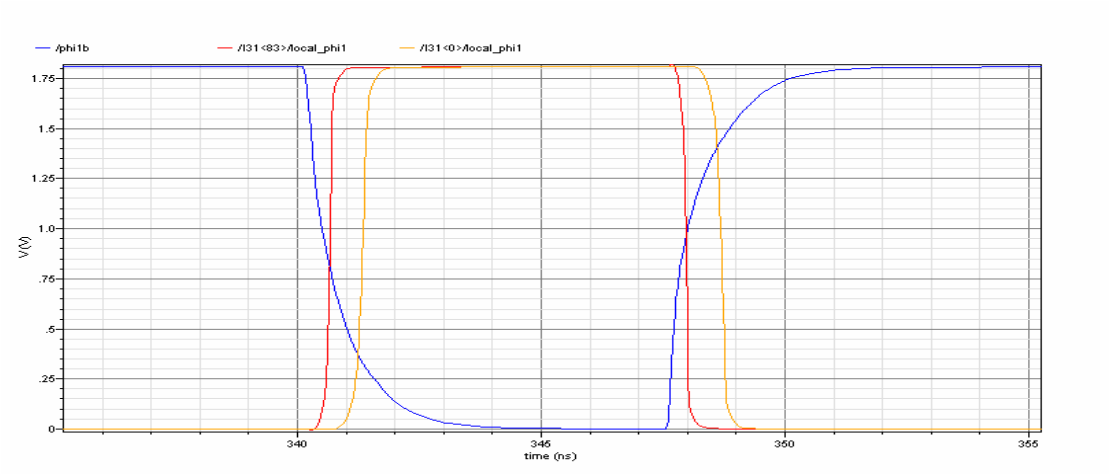


The condition when more than one enable signal is raised will cause conflict on the address bus and draw additional current. This scenario should be avoided – since the enable inputs are derived from the readout logic where only one cell is accessed at any one time this condition should be assured.

Logic Simulation: Driving Long Column Signals

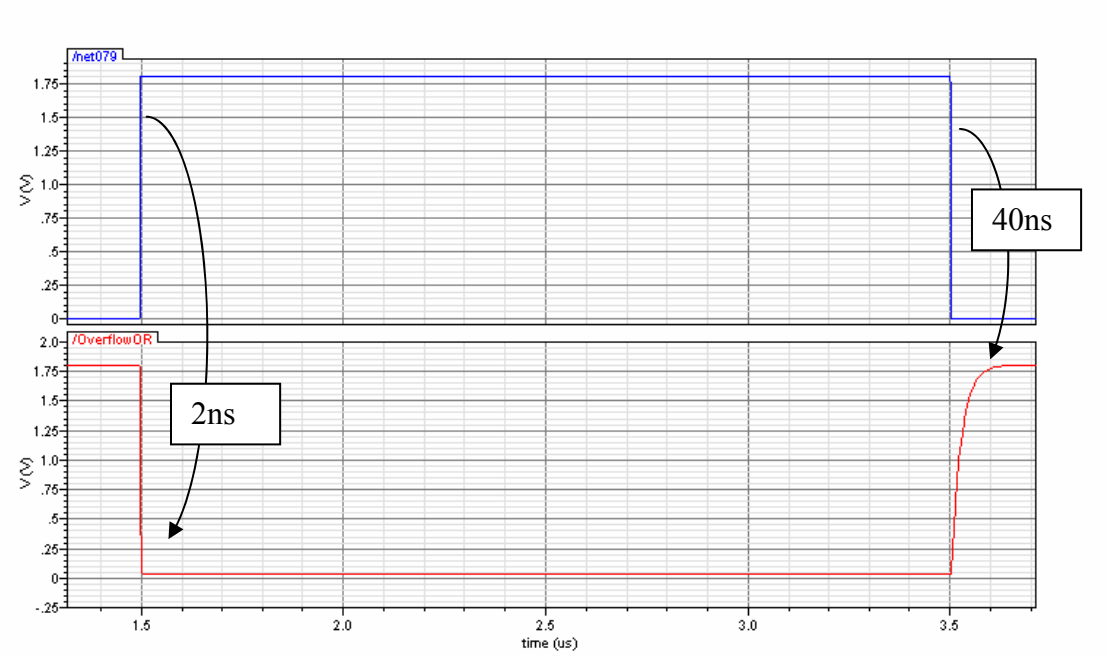
The master-controller logic is stripped of all functionality and only the input gates are left. This dummy block is instantiated 84 times with RC line models and driven with x4 strength buffers to check the clock and control signals can be operated at the required speeds.

Results Waveforms

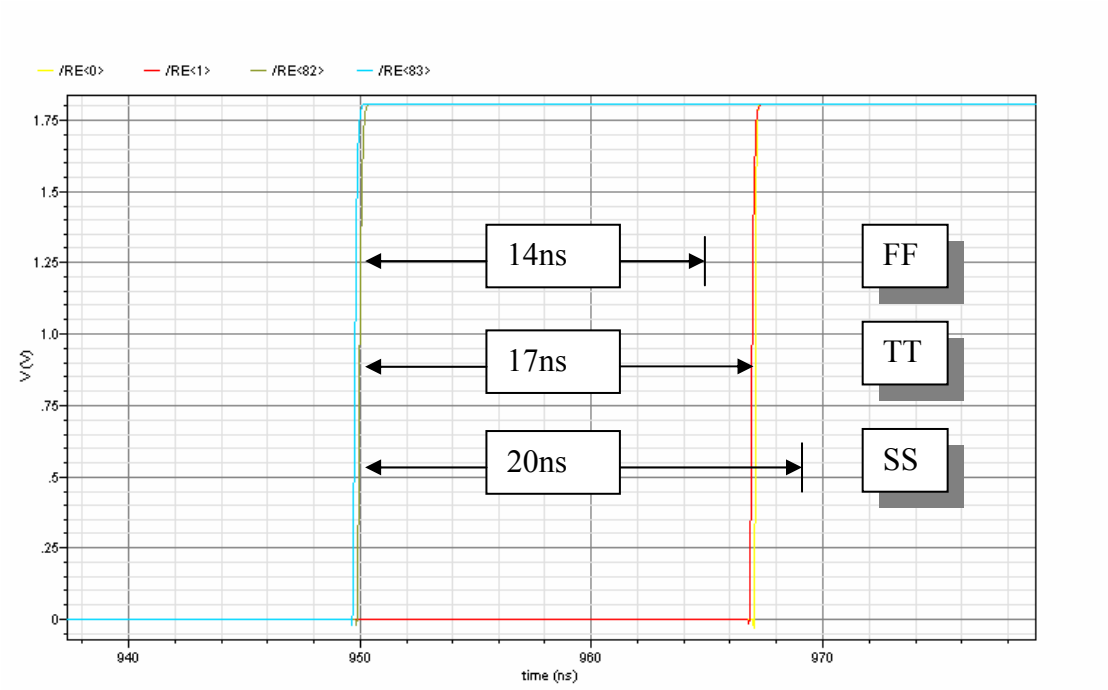


Above: Internal to each row, phi1 clock edges differ by <1ns between the top and bottom cells in the 84-high column. The blue trace shows the true clock signal on the shared column line that distributes the clock. The internal buffering within each row ensures the local clock edges are fast and true compliment.

The other control signals (fwd, hold, init etc) are buffered with the same x4 buffer, and loaded in the same way as the clocks (a single input port on an inverter/logic cell) at each row. This simulation is taken as verification for the buffering and operating speed of all these control signals.



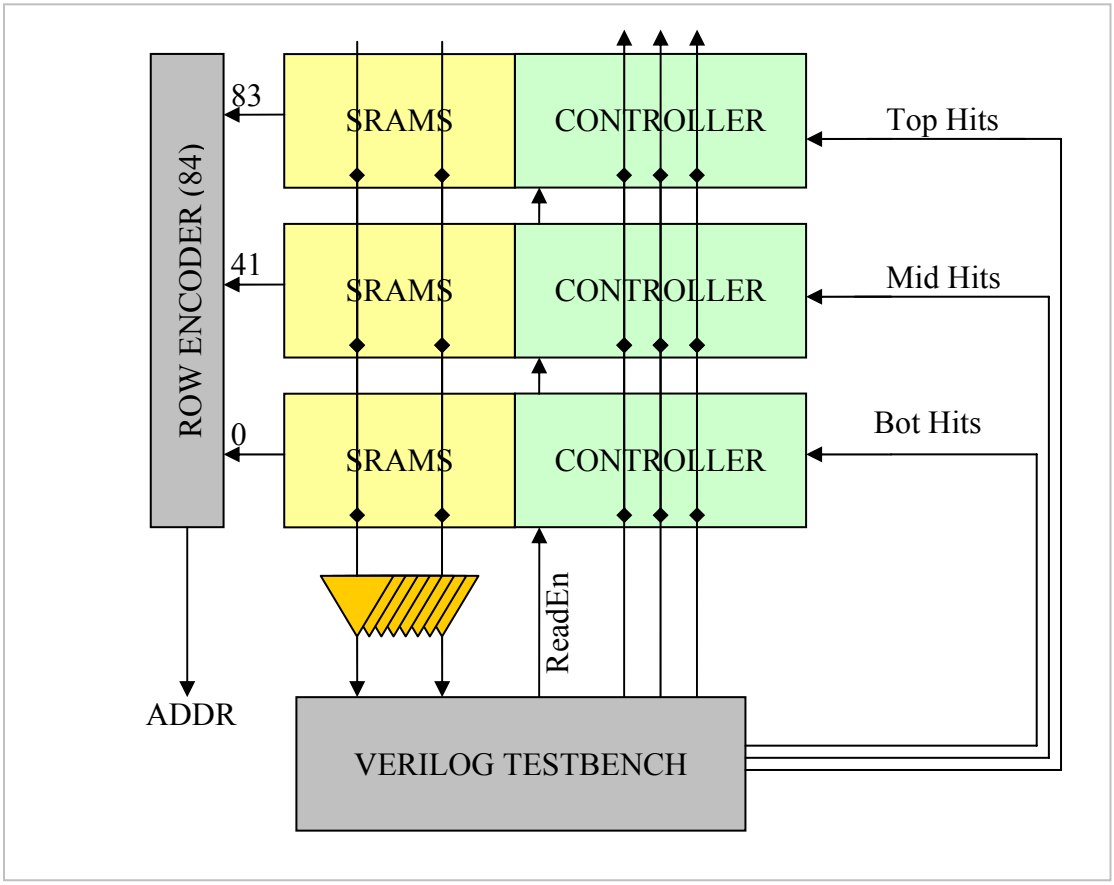
Above: The wired-or used to indicate overflow is seen to respond quickly on the occurrence of overflow, but is much slower to respond once the overflow has cleared. This is primarily a debug feature and therefore operation speed is unimportant; functionality is verified.



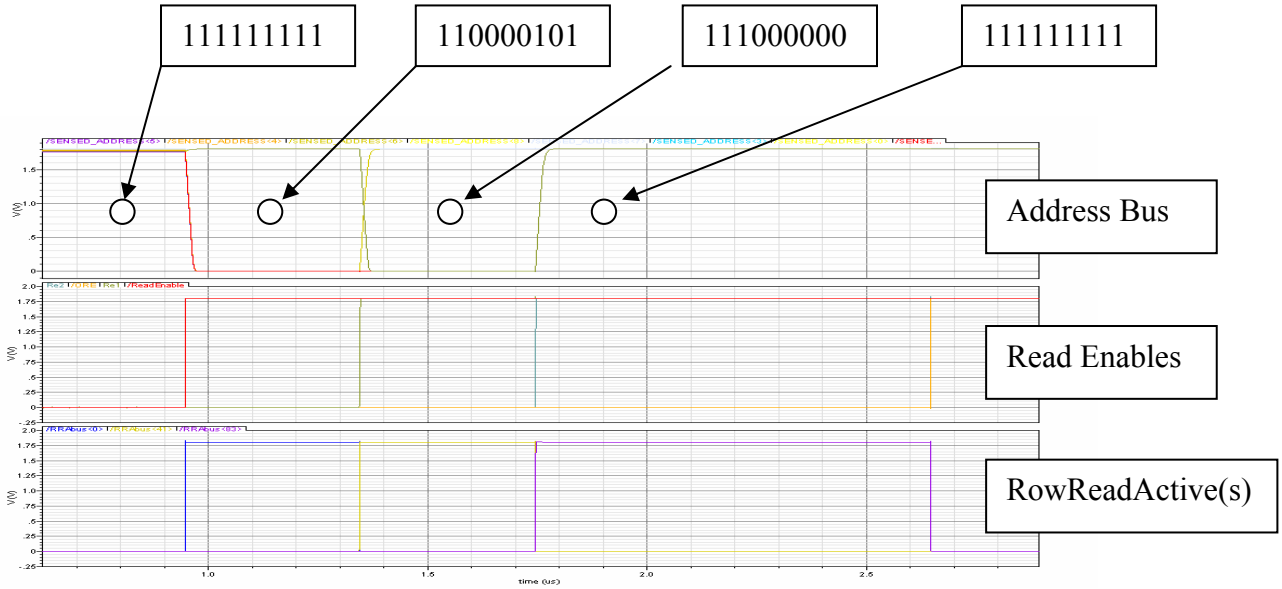
Above: The read-enable signal propagates from the bottom to the top of the column (since the dummy cells are all un-hit). The 17ns delay is the typical time taken for the read token to pass through all 84 cells this varies by up to 3ns in process corners. This sets a realistic maximum readout clocking rate of 25Mhz (assuming the token is collected and reset for a clock cycle after every 84-pixel stretch.) This operating speed is more than enough for the data rates of this project, especially if columns are operated in parallel.

Logic Simulation: Three Master-Controllers + Row Encoder

The diagram below depicts the system simulation to check integration of several key circuit blocks in a logic column formation. 17 hits are applied in 3 time divisions.



Results



Above: Inverted row addresses during readout from ‘Bot’, ‘Mid’ & ‘Top’ memories.
Below: Verilog simulation log excerpt with colour-coded annotations to aid reading.

Bank assignment codes =	100	101	111	110	010	011	001

```
Clear & Initialise the SR control registers
SR Reset complete
Top Hit pattern   = 000111000110000101000100000011000010000001 at time      173
Mid Hit pattern   = 000000000000000000000000000000000000000000 at time      173
Bot Hit pattern   = 100000000000000000000000000000000000000001 at time      173
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Latched.
Top Hit pattern   = 100000000000000000000000000000000000000001 at time      465
Mid Hit pattern   = 001110000000000000000000000000000000001000 at time      465
Bot Hit pattern   = 000000000000000000000000000000000000000000 at time      465
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Latched.
Top Hit pattern   = 000000000000000000000000000000000000000000 at time      657
Mid Hit pattern   = 100000000000000000000000000000000000000001 at time      657
Bot Hit pattern   = 011100000000000000000000000000000000000100 at time      657
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> Latched.
End of bunch train.
Readout phase:
SR Initialisation complete
```

Time stamp: 1, 2, 3

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000000011   (3)
     >>           BANK = 100
     >>           PATT = 011100
     >>           DATA = 0110111001111111111100
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000000011   (3)
     >>           BANK = 001
     >>           PATT = 000100
     >>           DATA = 1100001001111111111100
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000000001   (1)
     >>           BANK = 100
     >>           PATT = 100000
     >>           DATA = 0111000001111111111110
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000000001   (1)
     >>           BANK = 001
     >>           PATT = 000001
     >>           DATA = 1100000011111111111110
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000000011   (3)
     >>           BANK = 100
     >>           PATT = 100000
     >>           DATA = 0111000001111111111100
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000000011   (3)
     >>           BANK = 001
     >>           PATT = 000001
     >>           DATA = 1100000011111111111100
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000000010   (2)
     >>           BANK = 100
     >>           PATT = 001110
     >>           DATA = 0110011101111111111101
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000000010   (2)
     >>           BANK = 001
     >>           PATT = 001000
     >>           DATA = 1100010001111111111101
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000000010   (2)
     >>           BANK = 100
     >>           PATT = 100000
     >>           DATA = 0111000001111111111101
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000000010   (2)
     >>           BANK = 001
     >>           PATT = 000001
     >>           DATA = 1100000011111111111101
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000000001   (1)
     >>           BANK = 100
     >>           PATT = 000111
     >>           DATA = 0110001111111111111110
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>> HIT INFO: TIME = 0000000000001   (1)
     >>           BANK = 101
     >>           PATT = 000110
     >>           DATA = 0100001101111111111110
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000000001   (1)
     >>           BANK = 111
     >>           PATT = 000101
     >>           DATA = 0000001011111111111110
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000000001   (1)
     >>           BANK = 110
     >>           PATT = 000100
     >>           DATA = 0010001001111111111110
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000000001   (1)
     >>           BANK = 010
     >>           PATT = 000011
     >>           DATA = 1010000111111111111110

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000000001   (1)
     >>           BANK = 011
     >>           PATT = 000010
     >>           DATA = 1000000101111111111110

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
     >> HIT INFO: TIME = 0000000000001   (1)
     >>           BANK = 001
     >>           PATT = 000001
     >>           DATA = 1100000011111111111110
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
-
-
-
-
```