LDPC Error Correction Using Probability Processing Circuits

Ben Vigoda
CEO, Lyric Semiconductor, Inc.
www.lyricsemi.com

Bit errors are key barrier to flash growth

Bit errors are key barrier to flash growth

Flash Memory SUMMIT

Data stored in flash

Flash storage

Bit errors are key barrier to flash growth

Flash Memory SUMMIT

Data stored in flash

Flash storage

Data read out

1 bit wrong in every 100 stored!

Bit errors are key barrier to flash growth

Flash Memory SUMMIT

Data stored in flash

Flash storage

Data read out

1 bit wrong in every 100 stored!

Causes of bit errors:

- Shrinking flash cell size
- Multi-level cells (MLC, TLC, etc.)
- Program-erase (P/E) cycles

LDPC (Low Density Parity Check) is the solution for bit errors

LDPC (Low Density Parity Check) is the solution for bit errors

Flash Memory SUMMIT

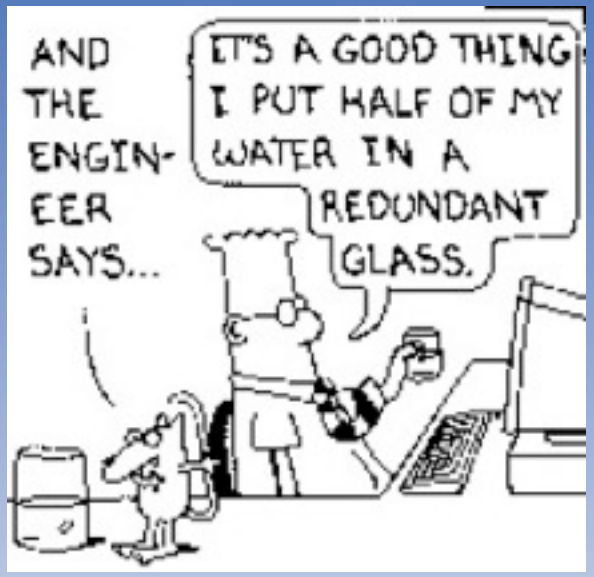
Raw error rate

LDPC (Low Density Parity Check) is the solution for bit errors

Raw error rate

LDPC

After LDPC



1 bit wrong in every 1e15 (1000 trillion) stored!

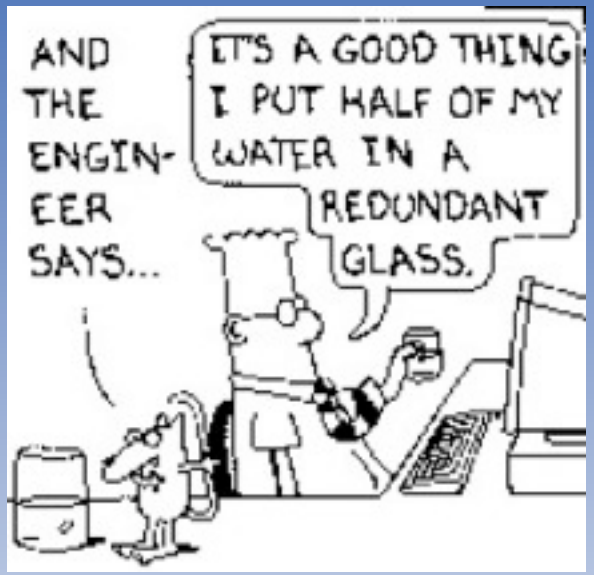
LDPC (Low Density Parity Check) is the solution for bit errors

Flash Memory SUMMIT

Raw error rate

LDPC

After LDPC



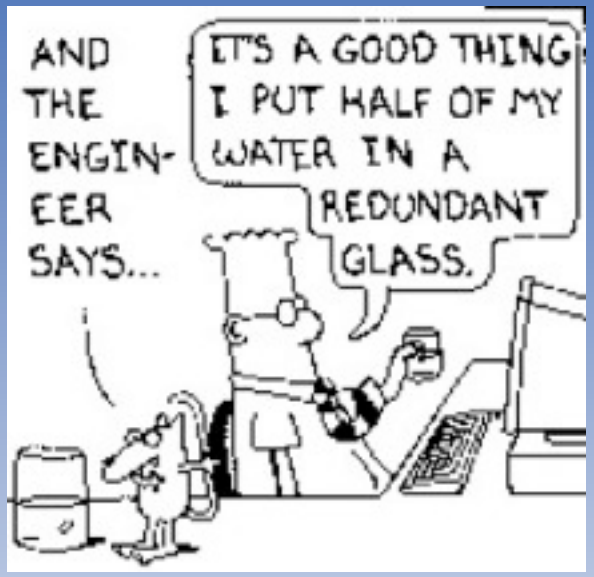
1 bit wrong in every 1e15 (1000 trillion) stored!

LDPC (Low Density Parity Check) is the solution for bit errors

Raw error rate

LDPC

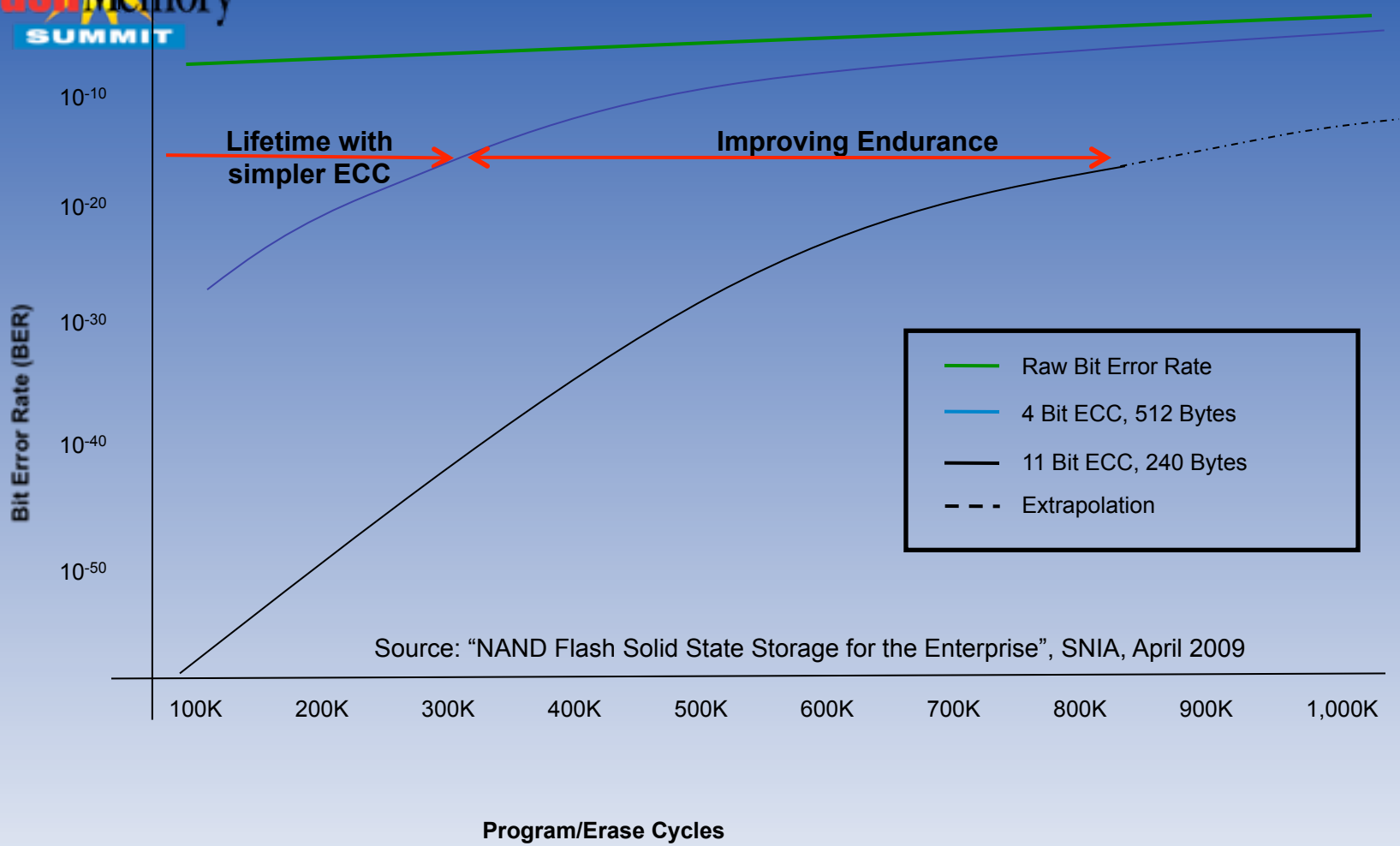
After LDPC



1 bit wrong in every 1e15 (1000 trillion) stored!

- **HDD has already begun to use LDPC**
- **SSD still needs LDPC** (or related soft iterative decoding)

LDPC Extends NAND Endurance



LDPC can combat increasing BER due to wear-out from Program/ Erase cycles

Motivation

Flash Memory SUMMIT

Today, the iPad contains MLC flash

Flash Memory SUMMIT

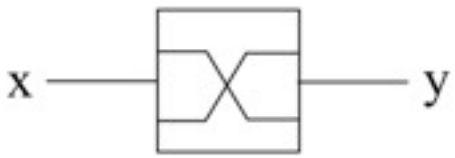
Today, the iPad contains MLC flash

- If we were using LDPC today, would 3BPC / TLC have the required longevity for use in the iPad?
- History shows that ECC is fundamental to storage

Bayesian logic gate example: Bayesian inverter

If x and y are opposites

$$x = NOT(y)$$
$$x \oplus y(mod2) = 1$$



x	y
0	1
1	0

their probabilities are also opposite

$$p_X(1) = p_Y(0)$$
$$p_X(0) = p_Y(1)$$

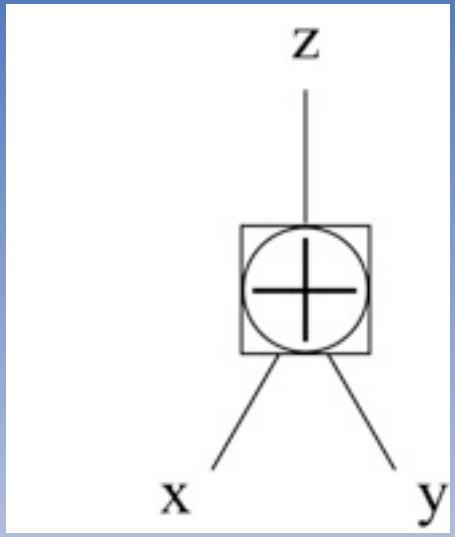
Bayesian logic gate example: Bayesian XOR

If x, y, and z satisfy a parity check constraint

$$z = XOR(x, y)$$

$$x \oplus y \oplus z(mod2) = 0$$

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0



their probabilities satisfy

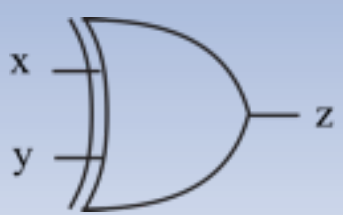
$$p_Z(1) = p_X(0)p_Y(1) + p_X(1)p_Y(0)$$

$$p_Z(0) = p_X(0)p_Y(0) + p_X(1)p_Y(1)$$

LDPC processor architecture

Digital processing:

- Built to process bits – 1s and 0s
- Bits flow through digital (Boolean) logic gates in one direction
- **A digital processor steps through a program performing each operation in sequence**

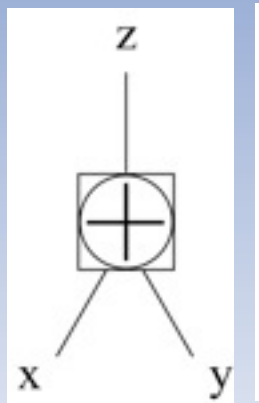


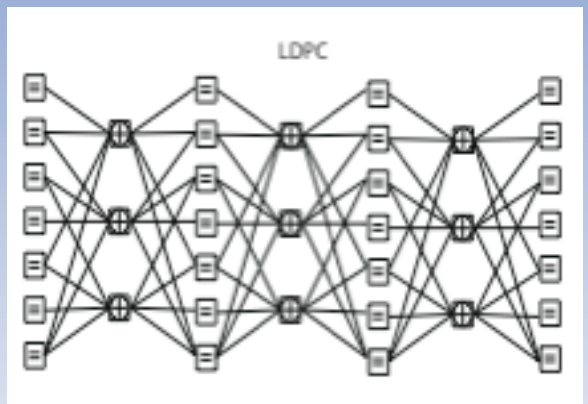
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

$$z = XOR(x, y)$$

Probability processing:

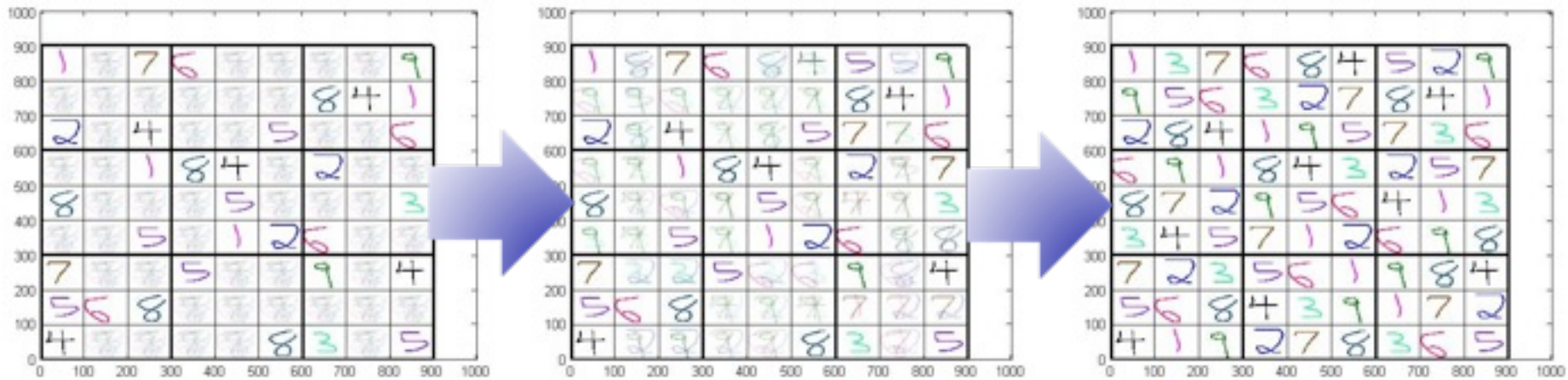
- Built to process "pbits" (probability bits) – prob(1) and prob(0)
- Pbits flow through probability gates multi-directionally
- **All the variables talk to each other – inherently highly parallel**





$$p_Z(1) = p_X(0)p_Y(1) + p_X(1)p_Y(0)$$
$$p_Z(0) = p_X(0)p_Y(0) + p_X(1)p_Y(1)$$

Lyric's PSBL Programming Language
(Probability Synthesis to Bayesian Logic)



```
perm = BuildPermutationFactorGraph(9);
sudokuVars = Variable(1:9,3,3,3,3);

fg = FactorGraph();

for i = 1:3
   for j = 1:3
      %   Entries in each row in Sudoku are distinct
      fg.addGraph(perm,reshape(sudokuVars(:,:,i,j),9,1));
      %   Entries in each column in Sudoku are distinct
      fg.addGraph(perm,reshape(sudokuVars(i,j,:,:),9,1));
      %   Entries in each 3x3 box in Sudoku are distinct
      fg.addGraph(perm,reshape(sudokuVars(:,i,:,j),9,1));
   end
end
```

```
%%%% Build permutation (sub-)factor graph on "N" objects
function fg = BuildPermutationFactorGraph (N)
    vars = Variable(1:N,N,1);
    fg = FactorGraph(vars);
    for i=1:N-1
       for j=i+1:N
          fg.addFunc(@unequal, vars(i), vars(j));
       end
   end
end

%%%% Unequal Gate:
function valid = unequal(a,b)
   valid = (a~=b);
end
```

```
%Set priors
for row = 1:9
  for col = 1:9
    if (puzzle(row,col) ~= 0 )
      ps = zeros(1,9);
      ps(puzzle(row,col)) = 1;
      vars(row,col).Priors = ps;
    end
  end
end

drawSudoku(vars);
```

The probability of meeting your true love

60%

40%

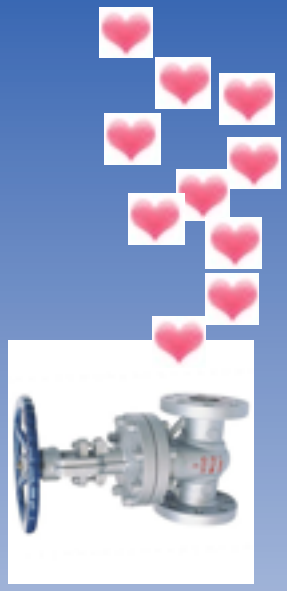
50%

50%

$$p(\text{true love}) = p(\text{A goes to empire}) \times p(\text{B goes to empire})$$

Probability multiply operation

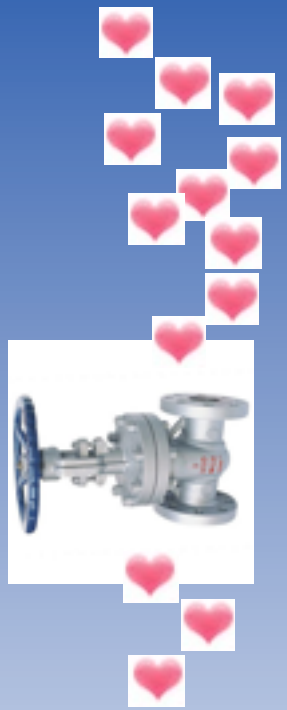
Start with 10 alternative days or universes



Probability multiply operation

Start with 10 alternative days or universes

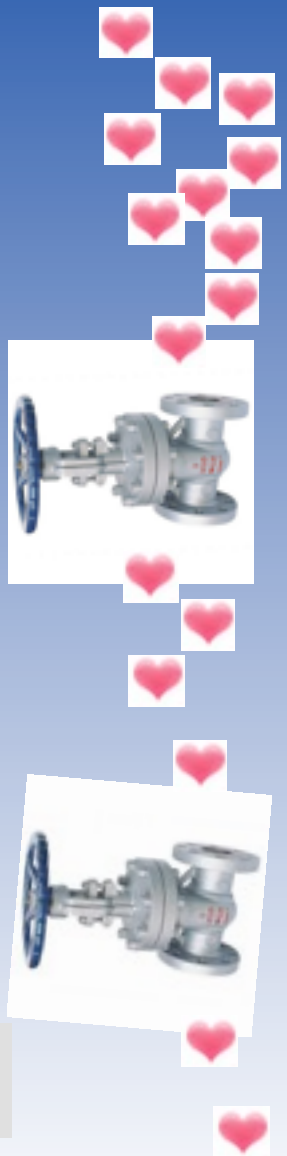
40%



Probability multiply operation

Start with 10 alternative days or universes

40%

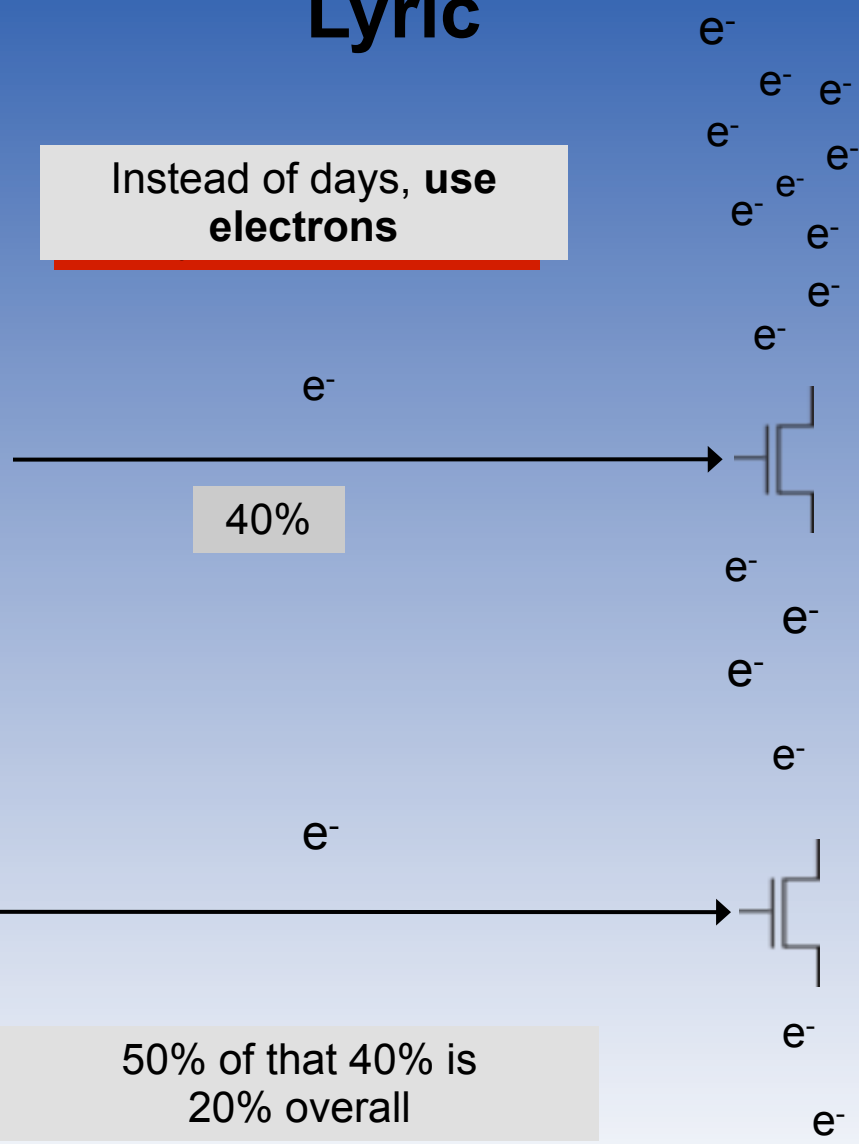


50% of that 40% is 20% overall

Probability multiply operation

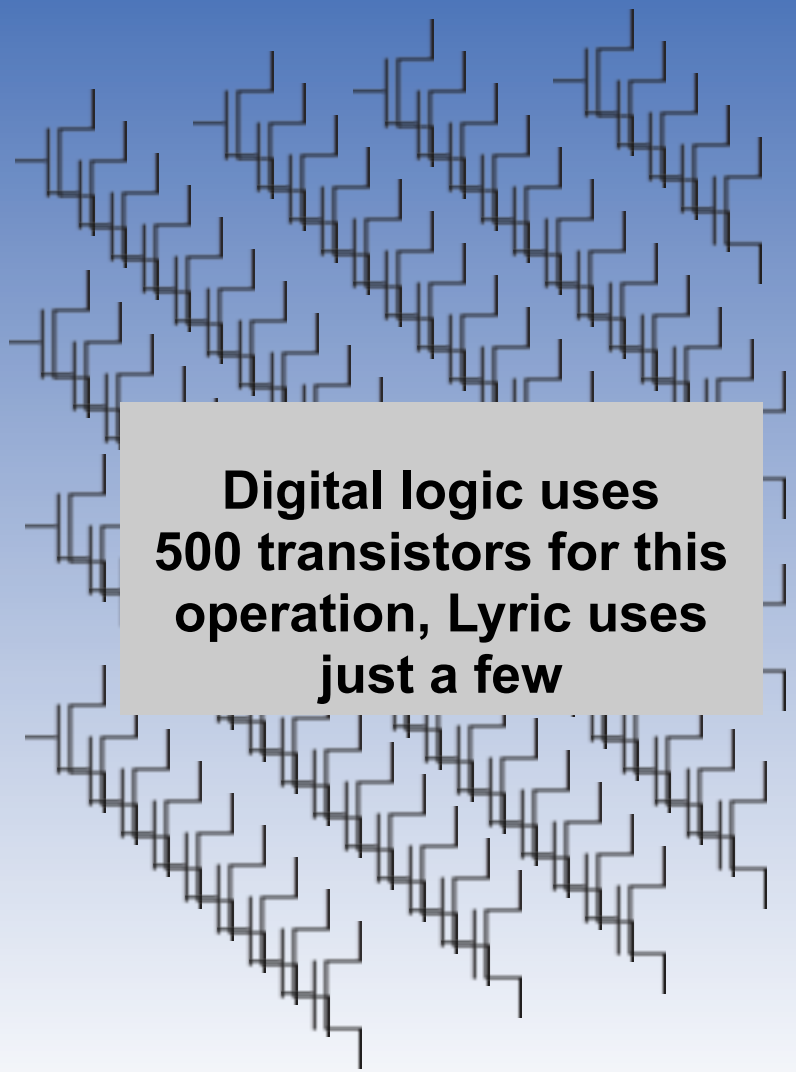
Lyric

Instead of days, **use electrons**



Probability multiply operation

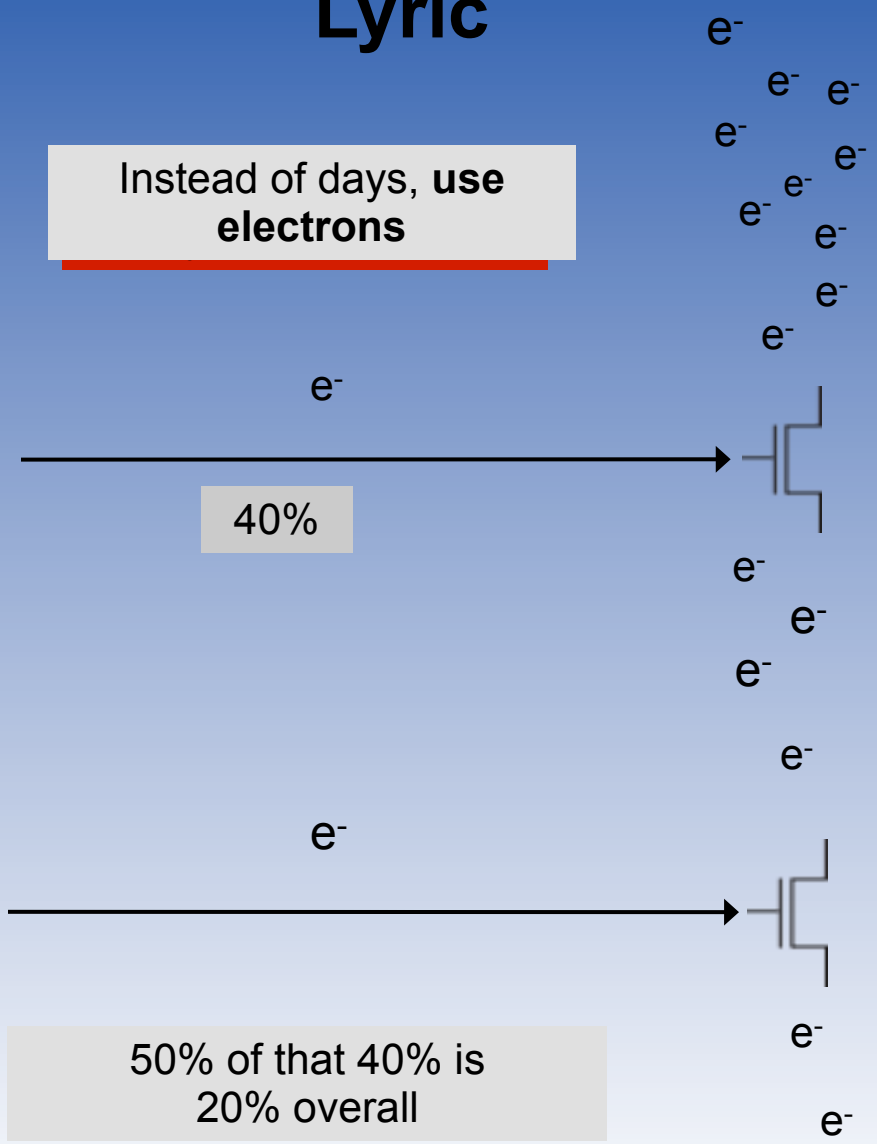
Digital



Digital logic uses 500 transistors for this operation, Lyric uses just a few

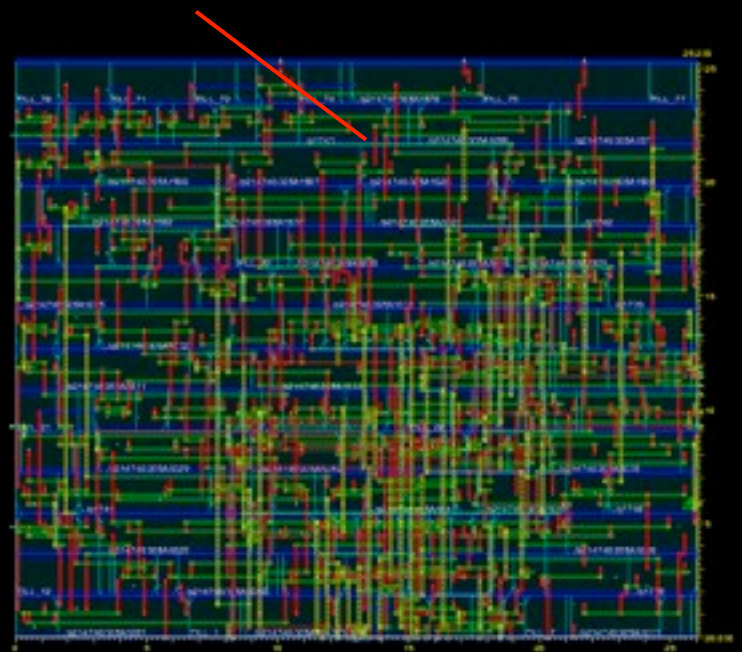
Lyric

Instead of days, **use electrons**



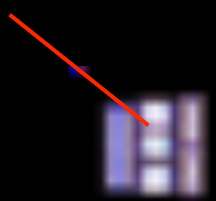
Probability processing operations in silicon (65nm CMOS)

Parity check cell implemented using digital gates in TSMC 65nm



Lyric's parity check cell in TSMC 65nm

(smaller and lower power than a digital implementation using 15nm CMOS)



scale: 10um

Lyric Error Correction (LEC™) for flash memory

Flash Memory SUMMIT

Advanced error correction implemented in digital

Standard digital approach

- **Large**
- **Power hungry**
- **Expensive**
- **Poor I/O bandwidth**

Advanced error correction using LEC technology

Higher performance advanced ECC

- **30X smaller at 1Gbps**
- **70X smaller at 6Gbps**
- **12X lower power**
- **4X I/O bandwidth between flash and controller**

Lyric Error Correction (LEC™) for flash memory

Flash Memory SUMMIT

Advanced error correction implemented in digital

Standard digital approach

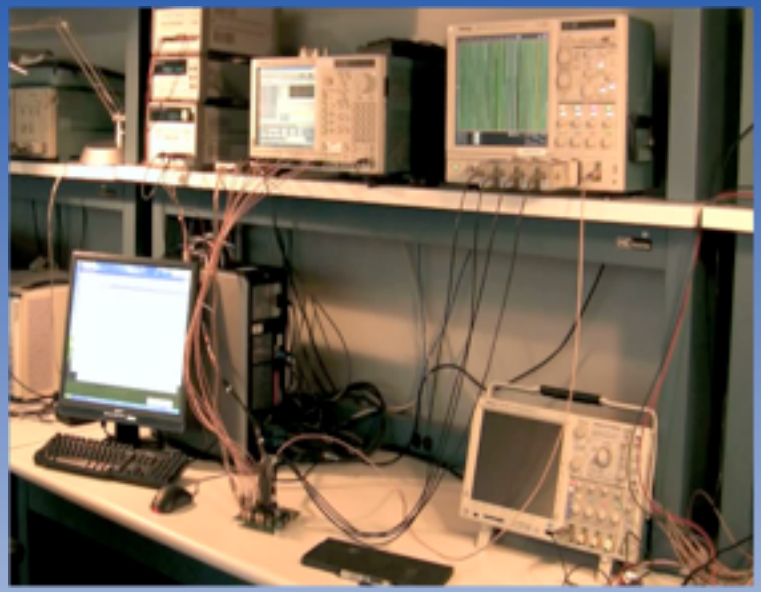
- **Large**
- **Power hungry**
- **Expensive**
- **Poor I/O bandwidth**

Advanced error correction using LEC technology

Higher performance advanced ECC

- **30X smaller at 1Gbps**
- **70X smaller at 6Gbps**
- **12X lower power**
- **4X I/O bandwidth between flash and controller**

LEC demonstration



Working with a top-tier flash manufacturer:

1. Encoded data with parity bits
2. Stored in 3Xnm TLC NAND flash
3. Baked in oven to simulate aging (raw BER approx. 10e-2)
4. Read out data including soft information
5. Processed this data through our LEC silicon using labview
6. BER rates are the same as benchmark digital implementations

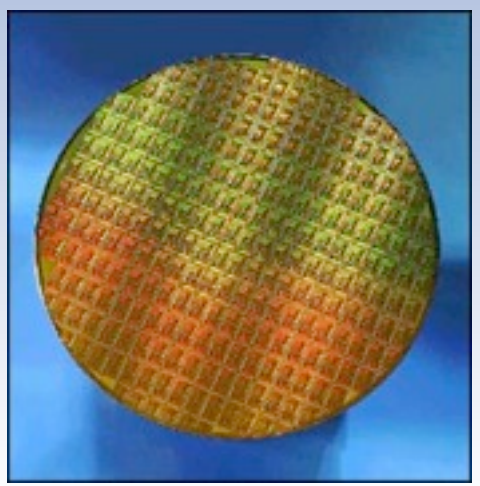
Potential applications

Mobile

Enterprise

- Improved IOPs per Watt and $/IOPs
- Extended longevity
- Lower latency

Foundry



- Improve effective quality of flash silicon
- Use MLC and TLC for high value applications
- Get to MLC and TLC into new nodes faster
- Better threshold detection for MLC, TLC, and beyond
- EZ-NAND: embed LEC with flash itself – output just the right bits

Potential applications

Flash Memory SUMMIT

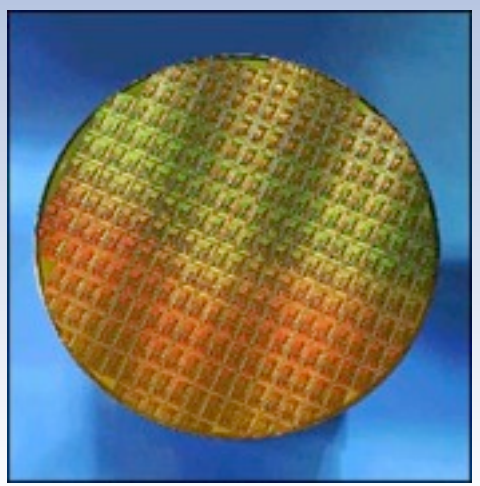
Mobile

Longevity even with 3BPC, lower power, lower cost

Enterprise

- Improved IOPs per Watt and $/IOPs
- Extended longevity
- Lower latency

Foundry



- Improve effective quality of flash silicon
- Use MLC and TLC for high value applications
- Get to MLC and TLC into new nodes faster
- Better threshold detection for MLC, TLC, and beyond
- EZ-NAND: embed LEC with flash itself – output just the right bits

Further reading

Further reading

Links to more technical information can be found at

Further reading

Links to more technical information can be found at www.lyricsemi.com