Digitally Synthesized Stochastic Flash ADC Using Only Standard Digital Cells

Skyler Weaver[1], Benjamin Hershberg[2], and Un-Ku Moon[2]
[1]Intel Corporation. Hillsboro, OR 97124. USA.
[2]School of EECS, Oregon State University, Corvallis, OR 97331. USA.

***Abstract*—It is demonstrated in this paper that it is possible to synthesize a stochastic flash ADC entirely from Verilog code and a standard digital library. An analog comparator is introduced that is constructed from two cross-coupled 3-input digital NAND gates, and can be described in Verilog. The synthesized comparators have random, Gaussian offsets that are used as virtual voltage references to make a flash ADC. A piecewise-linear inverse Gaussian CDF function is used to correct the nonlinearity introduced by the Gaussian offset distribution. The prototype IC is fabricated in 90nm CMOS and implements a 2047-comparator version of the proposed architecture. All components including the comparators, the ones adder, and the peicewise inverse Gaussian function are all implemented in Verilog. Conventional digital synthesis and place-and-route is then used to generate the physical layout, making this the first fully synthesized ADC. SNDR of 35.9dB (without calibration) is achieved at 210MSPS from the Verilog synthesized design.**

I. Introduction

An ADC is a mixed-signal system with an analog front-end and a digital back-end. Conventionally, ADCs depend heavily on good analog design, namely careful matching, layout, and linear circuits. As circuits scale into deep submicron, designing highly linear circuit components becomes increasingly difficult [1]. Therefore, an ADC architecture that can rely less on linear circuits is desirable.

The most essential component of an ADC is the comparator. It is the entity that ultimately does the translating from the analog world to the digital world. The circuits upstream from the comparator tend to be highly analog, and those downstream, digital. The digital circuits amplify signals all the way to the rails, so linearity is not important, only delay matters. Due to their innately low sensitivity to noise and physical layout, digital circuits lend themselves to automated synthesis. In order to minimize analog circuit requirements, it is appropriate to begin with an architecture that is already highly digital. Since the comparator defines the boundary between analog and digital realms, the flash ADC architecture will be considered, as it places the comparator as close to the analog input signal as possible (Fig. 1). If there is no pre-amplification, the only analog components in a conventional flash ADC are the analog voltage references and the comparators.

Flash ADCs use a reference ladder, in some form, to generate the comparator trip points that correspond to each digital code. Typically the references are either generated by a resistor ladder [2] or some form of analog interpolation [3], but the effect is the same: a reference is generated specifically for each comparator. First proposed in [4] is the stochastic ADC. A stochastic ADC eliminates an explicit reference generation and uses comparators' inherent input-referred offsets due to device mismatch as the trip-points. This has been shown that this can be an effective way to eliminate the need for a reference ladder [5]–[7]. Once reference generation is out of the picture, we are left with comparators and digital logic.

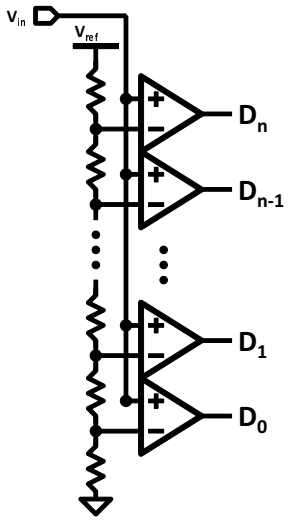


Fig. 1. A flash ADC (shown here as single-ended) is a set of monotonic references and a single comparator per reference level.

The remaining analog comparator, it will be shown in this paper, may also be constructed from digital blocks. For the first time, this allows for the possibility that the entire ADC can be constructed using standard digital synthesis flows; moreover, this type of stochastic flash ADC can be described in Verilog code.

First, the overall architecture will be discussed in Section II. Section III discusses the operation of an analog comparator that is constructed from standard digital NAND gates. Section IV deals with how to cause the flash ADC to have a linear transfer function even though the sampler thresholds are distributed in a nonlinear fashion. Section V will describe the specifics of fabricated test chip. The results of the said test chip will be presented in Section VI.

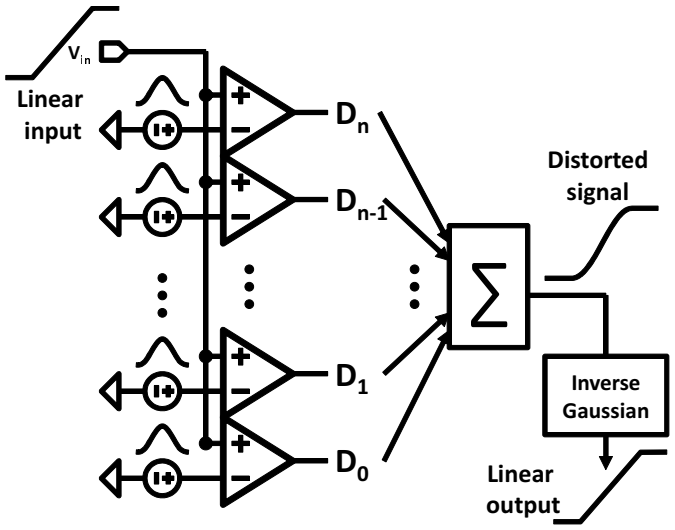


Fig. 2. A block diagram of the proposed stochastic flash ADC. The resistor ladder is replaced with random virtual references from comparator offsets. The random nature of the flash output calls for a ones adder. A inverse Gaussian CDF transformation compensates for the nonlinear distribution of the virtual references.

II. Synthesizable Stochastic Flash Architecture

In a conventional flash ADC, the input signal is connected to the inputs of a group of comparators. The threshold of each comparator is set precisely, by some sort of reference ladder, such that all comparator thresholds are equally spaced by 1 LSB. In reality there is also a random offset in each comparator that, in effect, readjusts each comparator threshold by a random amount. This random offset, due to device mismatches can be assumed to be a Gaussian distribution with a mean (μ) of zero and variance (σ^2) inversely proportional to comparator area [8].

In order to synthesize a flash ADC from Verilog code, there are a few key changes that must be made to the architecture. First of all, the resistor ladder must be removed. The differential input is then connected directly to the input of all of the comparators. Since there are no longer explicit voltage references, this architecture depends on the virtual voltage references that exist as comparator offsets due to random mismatch. If the mismatch is too small, then the input signal range will also be very small, so very small comparators are actually preferred.

In the case of a conventional flash ADC, the comparator outputs after a conversion can be expected to be a thermometer code since the comparator thresholds are monotonically increasing by design. Since comparator thresholds are random in a stochastic ADC, the order of the comparator outputs can also be expected to be random. The total number of comparators that evaluate high will still be monotonically increasing with an increase in the input voltage. Therefore a ones adder is required to sum the comparator outputs.

Finally, the raw output of the comparator outputs will be distorted by the non-uniform distribution the comparator offsets. A block is then required to un-distort the signal by passing it through the inverse function of the offset distribution. Section IV discusses this in detail.

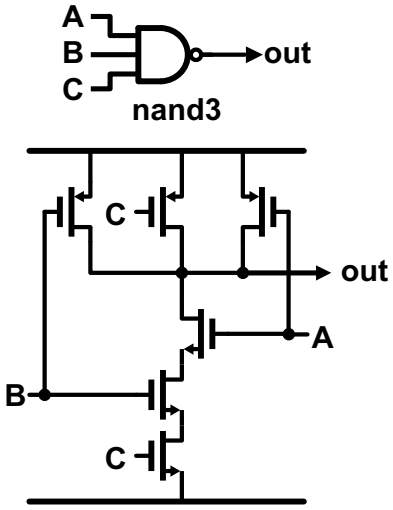


Fig. 3. A standard digital CMOS NAND3 gate and its internal transistor schematic.

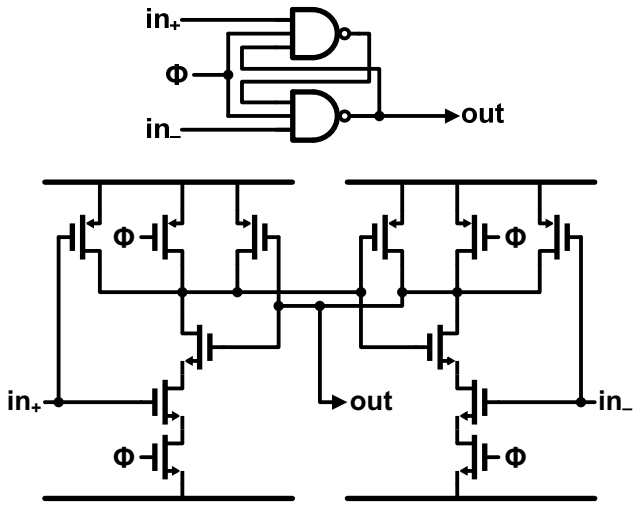


Fig. 4. An analog comparator made from standard digital NAND3 cells. The comparator operates of of a single phase clock Φ. When $\Phi = 0$ the comparator is reset through the parallel PMOS devices. When $\Phi = 1$ the discharge through the two series NMOS branches is determined by the differential input. Once the output drops low enough, regeneration occurs, which makes the digital decision. The input common-mode voltage must be high enough to not turn on the PMOS devices connected to the input.

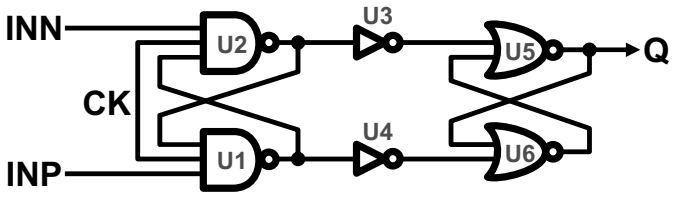
The block diagram of this architecture can be seen in Figure 2.

III. Analog Comparator from Digital Cells

Upon observation, the schematic of the transistors inside a CMOS NAND3 gate closely resemble half of a clocked analog comparator (Fig. 3). By connecting two NAND3 gates together as in Fig. 4, an analog-input comparator is created if the common-mode of the input is high enough to ensure that the PMOS transistors connected to the input are in the cutoff region of operation. When the clock Φ is low, both outputs are reset to the positive supply rail. When the clock goes high, the outputs will begin to discharge through the three series NMOS devices. The discharge rate depends on the capacitance on the

```
1  module comparator(INP, INN, CK, Q);
2   output Q;
3   input INP, INN, CK;
4
5   wire op, on, opn, onn, qn;
6   nand3x1 U1 ( .A(op), .B(INP), .C(CK), .Y(on) );
7   nand3x1 U2 ( .A(on), .B(INN), .C(CK), .Y(op) );
8   invx1 U3 ( .A(op), .Y(opn) );
9   invx1 U4 ( .A(on), .Y(onn) );
10  nor2x2 U5 ( .A(qn), .B(opn), .Y(Q) );
11  nor2x2 U6 ( .A(Q), .B(onn), .Y(qn) );
12
13  endmodule
```

(a)



(b)

Fig. 5. a) Verilog module 'comparator' which implements a NAND3 based comparator (lines 6-11). Inverters buffer the comparator output to a SR-latch. This is to remove any memory effect caused by the latch. b) Gate-level schematic representation of the code.

output node and the current through the three series devices. Since one of the series devices is connected to the analog input, the discharging current is proportional to the input. Once one of the outputs discharges to below a PMOS threshold voltage, the cross-coupled connection creates positive feedback that causes the comparator to force the outputs all the way to the supply rails. Implementing such a comparator can be done by explicitly referencing the standard library cells in the RTL Verilog code as in Fig. 5(a). In this example, a static SR-latch is added to the output of the comparator. The SR-latch holds the output data valid while the comparator is reset. The SR-latch input is buffered with inverters to reduce a memory-effect on the comparator due to the SR-latch.

Although this circuit is inherently compatible with digital synthesis, the synthesizer will assume that the circuit is actually a digital one, and will try and optimize it by replacing some of the gates or changing the circuit entirely while maintaining the same digital function. This digital optimization may render the circuit nonfunctional from an analog perspective, so here the synthesis directive `set_dont_touch comparator`, or equivalent, will prevent the synthesizer from altering the comparator module.

IV. Gaussian Distribution Mapped to a Uniform Distribution

The probability density function (PDF) of random comparator offset is influenced by many factors such as random variation of threshold voltage and current factor [9]. The Central Limit Theorem indicates that since comparator offset is a sum of independent random variables with finite mean and variance the PDF will be approximately Gaussian [10]. When a ramp signal is applied to the input of a stochastic flash ADC, the output will follow the cumulative distribution function (CDF) of comparator offset; therefore, the voltage transfer function of a stochastic flash ADC is the CDF of the random comparator offset. The number of comparators in the stochastic flash ADC must be enough such that the actual transfer function resembles the comparator offset CDF to the desired degree.

To calculate the number of comparators required for a desired N effective bits, let us consider an ADC with random comparator thresholds with a uniform distribution. Let us say that there are n comparator thresholds within the range 0 to 1, and the number of thresholds within the range 0 to v (where v is value between 0 and 1) is equal to k. This implies that the remaining thresholds within the range v and 1 are equal to $(n-k)$. For a random uniform distribution of n, the random variable k is a binomial distribution [11] with a probability mass function (PMF) of

$$\mathrm{PMF}_k(n,v) = \binom{n}{k}(v)^k(1-v)^{n-k}. \tag{1}$$

The quantization error of this ADC as a function of v is then

$$Q_{error}(v) = \frac{k}{n} - v. \tag{2}$$

Calculating the SQNR allows to find the effective number of bits, N. The quantization power is calculated by finding the variance of the quantization error and integrating over v,

$$\begin{aligned} Q_{power} &= \int_0^1 \sum_{k=0}^{n} \left(\frac{k}{n} - v\right)^2 \binom{n}{k}(v)^k(1-v)^{n-k}dv \\ Q_{power} &= \int_0^1 \frac{v - v^2}{n}dv \\ Q_{power} &= \frac{1}{6n}. \end{aligned} \tag{3}$$

With the quantization power known, SQNR of the stochastic ADC can be calculated as

$$SQNR = \frac{\sqrt{P_{signal}}}{\sqrt{P_{noise}}} = \frac{\sqrt{\frac{1}{12}}}{\sqrt{\frac{1}{6n}}} = \sqrt{\frac{n}{2}}. \tag{4}$$

This leads to the results that the number of n comparators required for N effective bits is $2 \cdot 4^N$ for a random uniform distribution of comparator thresholds [12][13]. When DC offset of the ADC can be ignored, as in the usual case, the comparator requirement is reduced to 4^N[14].

In this work, the expectation is not that comparator levels will have a uniform distribution; but rather, a Gaussian distribution of comparator thresholds is expected. To achieve an output that is linear, the ADC must take advantage of the fact that we have the knowledge of the shape of the random offset distribution a priori: it is Gaussian. A functional block with an inverse Gaussian CDF transfer function can be placed after the raw ADC output to linearize the output.

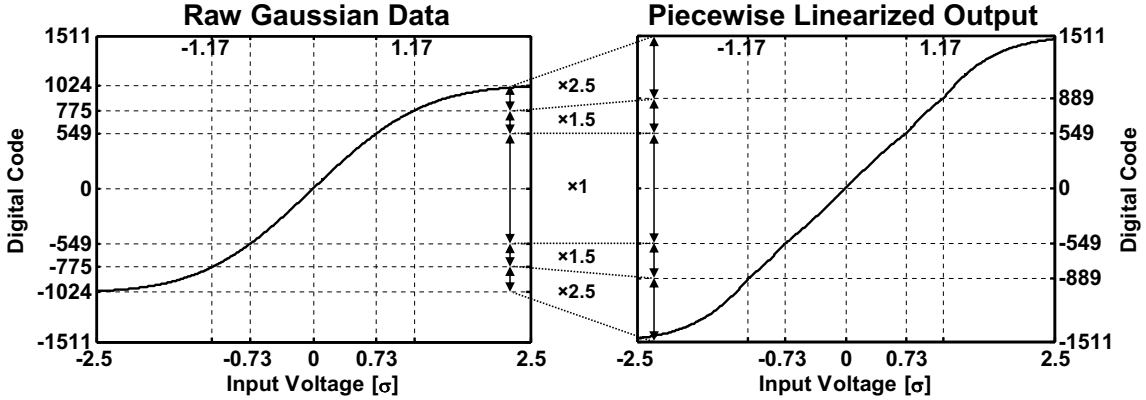


Fig. 6. This figure shows the piecewise linear inverse Gaussian CDF function for a 2047-comparator stochastic ADC. If the digital code is between -549 and +549, the output is untouched. As the signal goes beyond code 549, the slope of the transfer function is decreasing, so a multiplier is used to compensate. Multipliers of 1, 1.5 (1+1/2), and 2.5 (2+1/2) are chosen for the low hardware overhead of multiplication and devision by factors of 2. The final result gives a more linear transfer function.

The inverse Gaussian block can be implemented as either as a lookup table or a digital mathematical function. There are grave disadvantages to implementing this as a lookup table in hardware. A lookup table is a large hardware requirement since it is implemented as a memory that must be able to output as fast as the ADC. Depending on the accuracy that is being designed for, a piecewise linear approximation of an inverse Gaussian CDF may be sufficient. Fig. 6 is an example of such a piecewise function defined by,

$$f(v) = \begin{cases} 2.5v - 0.512 & : & 1.166\sigma < & v & \\ 1.5v - 0.134 & : & 0.732\sigma < & v & \leq 1.166\sigma \\ v & : & -0.732\sigma \leq & v & \leq 0.732\sigma \\ 1.5v + 0.134 & : & -1.166\sigma \leq & v & < -0.732\sigma \\ 2.5v + 0.512 & : & & v & < -1.166\sigma \end{cases}, \tag{5}$$

where v is the output of the transfer function and is in terms of standard deviation σ. This example only has five piecewise regions. More regions can be added for more accuracy. For the design that was targeted for this work, five regions were sufficient. The fact that the inverse Gaussian function is a function of the raw ADC *output* is very important. By being a function of the output, it does not matter what the input characteristics are, or whether or not there is an offset to the distribution.

If the mean or standard deviation of the comparator offset distribution should change due to variation in process, voltage, temperature, or any other reason, the CDF transfer function would be shifted and scaled with respect to input voltage; however, the shape of the CDF remains the same. The digital output only represents the shape of the CDF, e.g. code 0 (signed) always represents the mean of the distribution, and code +699 (for 2047 comparators) always represents one standard deviation above the mean. Therefore, no calibration or tuning is required; the inverse Gaussian CDF can be hard coded into the chip.

When implementing (5) for an actual ADC the values for the piecewise function can be rounded to the nearest code.

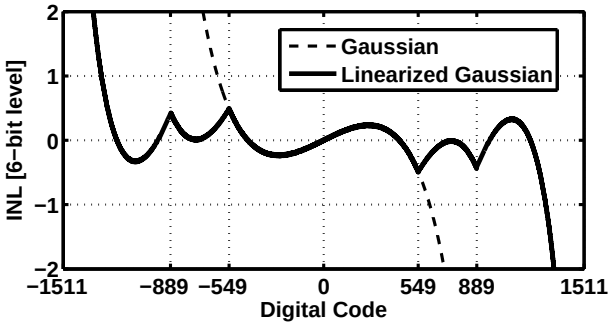


Fig. 7. A plot of the integral nonlinearity for a 2047-comparator stochastic flash ADC due to the shape of the Gaussian distribution. The remaining integral nonlinearity after applying piecewise linear approximation function is also plotted. Note that the 6-bit linear range is nearly doubled. This implies that almost twice the number of comparators lie within the "useful range" halving the number of comparators required.

As an example, consider a 2047-comparator stochastic flash ADC. The function that would actually be implemented is,

$$f(v) = \begin{cases} 2v + v/2 - 1049 & : & 775 > & v & \\ v + v/2 - 274 & : & 549 < & v & \leq 775 \\ v & : & -549 \leq & v & \leq 549 \\ v + v/2 + 274 & : & -775 \leq & v & < -549 \\ 2v + v/2 + 1049 & : & & v & < -775 \end{cases}. \tag{6}$$

Note that the coefficients of v were chosen to be factors of two as to reduce the hardware complexity of implementation, since multiply-by-two and divide-by-two have nearly no hardware cost in binary arithmetic. The effect of this linearization on overall INL can be seen in Fig. 7. The INL is only within the desired 6-bit level for a small range for the uncorrected Gaussian transfer function. By adding this correction, the effective linear range is increased. The end result is that an inverse Gaussian function placed after the ADC output effectively transforms the random comparator thresholds to a uniform distribution. This makes the 4^N comparator requirement true for the linearized range of the distribution. This can be verified mathematically and is verified

```
1  module adc(inp, inn, clk, dec_en, out);
2   input inp, inn, clk, dec_en;
3   output [4:0] out;
4
5   wire [6:0] q;
6   wire [4:0] fastout;
7   reg [4:0] out;
8   reg [2:0] dec_cnt;
9
10  comparator U1(.INP(inp),.INN(inn),.CK(clk),.Q(q[0]));
11  comparator U2(.INP(inp),.INN(inn),.CK(clk),.Q(q[1]));
12  comparator U3(.INP(inp),.INN(inn),.CK(clk),.Q(q[2]));
13  comparator U4(.INP(inp),.INN(inn),.CK(clk),.Q(q[3]));
14  comparator U5(.INP(inp),.INN(inn),.CK(clk),.Q(q[4]));
15  comparator U6(.INP(inp),.INN(inn),.CK(clk),.Q(q[5]));
16  comparator U7(.INP(inp),.INN(inn),.CK(clk),.Q(q[6]));
17
18  dsp U8 ( .c0b0(q), .clk(clk), .final(fastout) );
19
20  always @(negedge clk) begin
21   if (dec_en == 1) begin
22    if (dec_cnt == 0)
23     out <= fastout;
24    dec_cnt <= dec_cnt + 1;
25   end
26   else
27    out <= fastout;
28  end
29
30  endmodule
```

Fig. 8. Verilog module 'adc' which creates an 7-comparator ADC and includes a decimate by 8 option (lines 20-28).

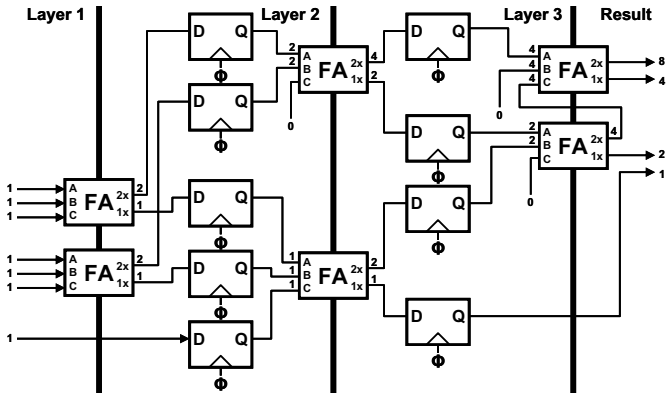


Fig. 9. Diagram of a pipelined Wallace tree ones adder for 7 inputs. D-flip-flops add latency in exchange for increasing the allowable throughput.

```
1  module dsp(c0b0, clk, final);
2   output final;
3   input c0b0, clk;
4
5   wire clk;
6   wire [6:0] c0b0;
7
8   reg [4:0] final;
9   reg [3:0] sum;
10  reg [4:0] sum2;
11
12  reg c1b0[2:0];
13  reg c1b1[1:0];
14
15  reg c2b0;
16  reg c2b1[1:0];
17  reg c2b2;
18
19  always @(negedge clk) begin
20
21  {c1b1[0],c1b0[0]} <= c0b0[0]+c0b0[1]+c0b0[2];
22  {c1b1[1],c1b0[1]} <= c0b0[3]+c0b0[4]+c0b0[5];
23  c1b0[2] <= c0b0[6];
24
25  {c2b1[0],c2b0} <= c1b0[0]+c1b0[1]+c1b0[2];
26
27  {c2b2,c2b1[1]} <= c1b1[0]+c1b1[1];
28
29  sum <={c2b2,c2b1[0],c2b0}+{1'b0,c2b1[1],1'b0};
30
31  sum2 <={c2b2,c2b1[0],c2b0}+{1'b0,c2b1[1],1'b0}-4;
32
33  if(sum > 7)
34   final <= {sum2[3:0],1'b0}+{sum2[4],sum2[4:1]}-4;
35  else if(sum > 6)
36   final <= (sum2) + {sum2[4],sum2[4:1]} - 1;
37  else if(sum >= 2)
38   final <= (sum2);
39  else if(sum >= 1)
40   final <= (sum2) + {sum2[4],sum2[4:1]} + 1;
41  else
42   final <= {sum2[3:0],1'b0}+{sum2[4],sum2[4:1]}+4;
43
44  end
45
46  endmodule
```

Fig. 10. Verilog module 'dsp' which implements the pipelined Wallace ones adder (lines 21-29) and piecewise Gaussian-to-uniform function(lines 33-42).

experimentally by the prototype IC that was implemented.

V. Implementation Details

As shown in Figure 2, there are three functional blocks in the prototype ADC: a group of 2047 comparators, an 11-bit output ones adder, and the piecewise inverse Gaussian function. All of these circuits were implemented in Verilog code, digitally synthesized, and automatically placed and routed for the final layout.

For simplicity, the example code in this paper describes an implementation of a 7-comparator stochastic flash ADC. The actual fabricated ADC included 2047 comparators, but its code can be extrapolated from the example code shown.

The Verilog code that implements the ADC at the top level

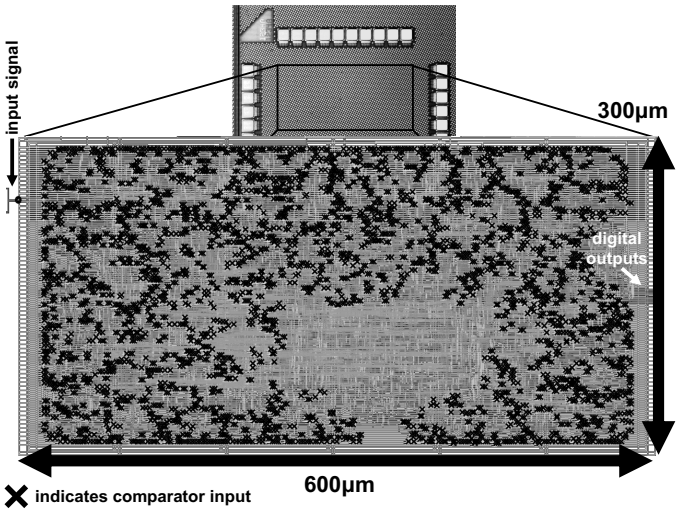


Fig. 11. Screen capture of the prototype IC with the comparator inputs each marked as a black x. The input network is automatically routed with default routing options to demonstrate true digital synthesizability. Practical implementations would direct the router to reduce the RC-delay of the input network. Dimensions are 300μm by 600μm.

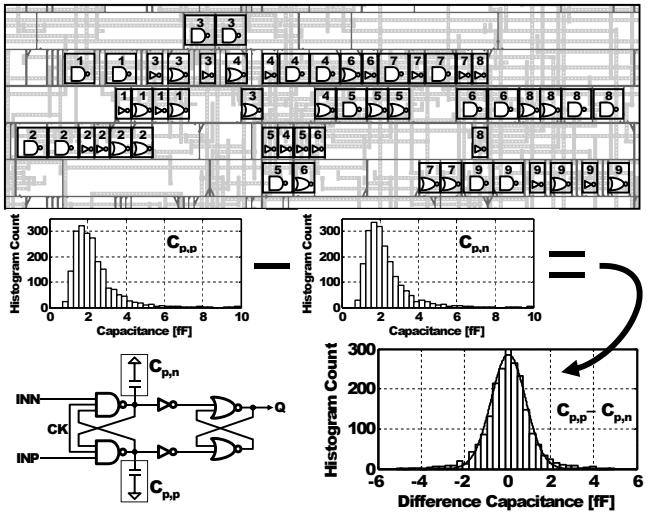


Fig. 12. Layout detail of synthesized comparator layouts and resulting systematic parasitic capacitance. The systematic variation is also found to follow a Gaussian distribution.

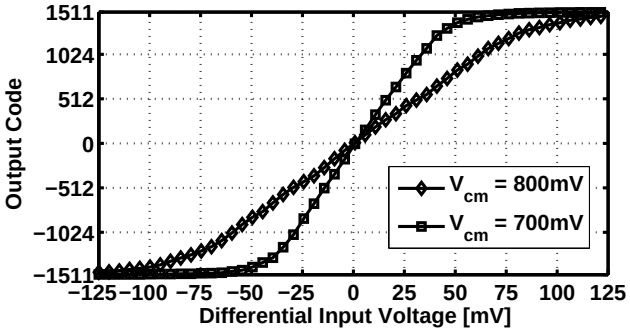


Fig. 13. Measured output transfer function of the ADC for different input common mode voltages.

can be seen in Fig. 8. The outputs of the seven comparators, `q[0]`, `q[1]`, ..., `q[7]` are passed the the module `dsp` which consists of a ones adder and the piecewise inverse Gaussian function. Lines 20–28 implement a decimate-by-8 for testing purposes: when decimation is enabled by the input `dec_en` being set to "1" (`dec_en` is connected to an external pin) the output `out` is updated whenever the free-running counter `dec_cnt` is equal to zero. Since `dec_cnt` is defined as a 3-bit register, it will be equal to zero every eight cycles.

A Wallace tree is typically used for efficient and high speed ones addition [15]; an example of which can be seen in Fig. 9. A Wallace tree uses single-bit binary full-adders as 3:2 compressors. Each full adder takes in three binary bits of the same bit-weight and outputs the same total value: one bit of the same bit-weight as the inputs, and one bit of twice the bit-weight as the inputs. The summation then proceeds as follows. Take some number of inputs of the same bit-weight, seven, for example. For every set of three or two, sum them in a full-adder; single remainders pass to the next compression layer as-is. At the next layer, group all of the bits of the same bit-weight and repeat. Continue operating until there are two or less of each bit-weight and sum the result in a conventional carry-look-ahead adder. This method of adding many bits of the same bit-weight can be made very fast since the time delay from one layer of inputs to the next is only a single full-adder delay. Contrast this to a ripple-carry-adder tree where the time delay from one layer to the next may be many full-adder delays. To increase the speed further, D-flip-flops are placed between each layer of full-adders to pipeline the adder tree. This creates many cycles of latency, in this case 14 cycles, in exchange for a higher clock rate. The implementation of the Wallace tree from Fig. 9 is done in Verilog as lines 21–29 of Fig. 10.

The piecewise linear approximation of the inverse Gaussian function is implemented as a series of `if` statements in lines 33–42 of Fig. 10. In this example, there are only 7 comparators. For the actual ADC that was fabricated with 2047 comparators, the function from (6) was implemented.

VI. Measurement Results

The test chip was fabricated in a 90nm digital CMOS process with a total area of 0.18 mm^2, with a final digital gate count of 23,143 gates. The top metal layer was covered with dummy metal fill, so there was nothing interesting to see in the die micrograph, so a screen capture of the layout can be seen in Fig. 11. At the point where the input net connects to each comparator input, a bold "x" has been placed to show the detail of the comparator placement. The comparator placement is resemblant of a fractal tree because of the fractal-like nature of a large Wallace tree ones adder.

Once the layout was synthesized, a parasitic extraction would give insight into any systematic comparator offset due to layout. It was speculated that if the place and route tool could only place the comparators in, for example, three unique placements this could have an adverse effect on the overall offset distribution. Upon observation of the layout, there are many unique comparator placements (Fig. 12). Since

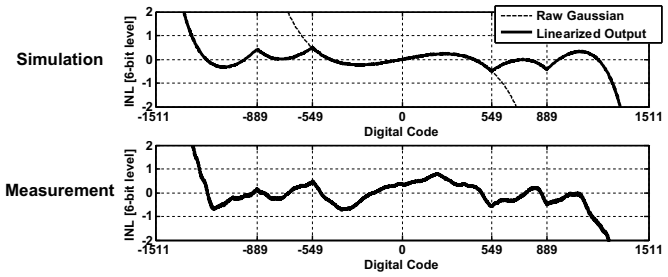


Fig. 14. Simulated and measured INL. The effect of the piecewise inverse Gaussian function can be seen to follow the expected trend.

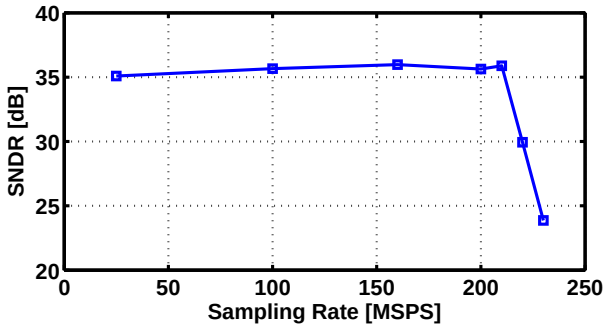


Fig. 15. Measured SNDR from a sine-wave test as function of sampling rate. The abrupt cliff at 210 MSPS is due to setup time violations occurring in the digital logic.

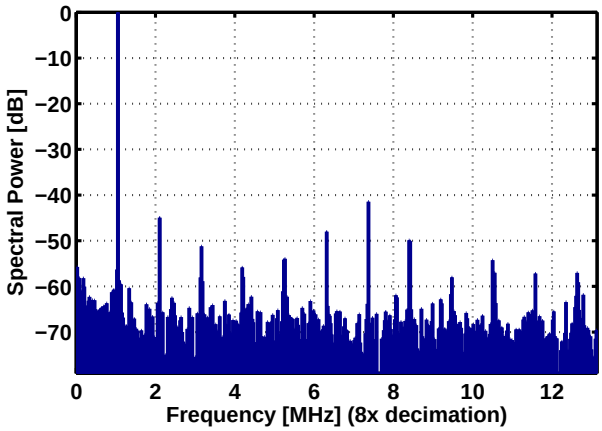


Fig. 16. Spectral plot of 1MHz input sine-wave at 210MSPS achieving 35.9dB SNDR. 8x decimation was used.

the comparators are sensitive to the capacitance seen at the outputs of the NAND gates, the routing parasitic capacitance at each NAND output was extracted. The result of the extraction shows that the parasitic capacitance follows a log-normal distribution. More importantly, the difference in capacitance between the positive and negative output nodes follows a zero-mean Gaussian distribution. Since the systematic offset distribution is Gaussian, and the random offset distribution is also expected to be Gaussian, the final distribution will also be Gaussian.

To test the functionality of the linearizion of the piecewise inverse Gaussian approximation, a ramp input was applied to the ADC to obtain the transfer function (Fig. 13). It is also observed that changing the common-mode of the input signal affects the variance of the comparator offset distribution. Varying supply voltage and temperature will also have an effect on the variance of the comparator offset.

Obtaining the transfer function also makes it possible to measure the INL of the ADC (Fig. 14). The trend of the INL from simulation can be clearly seen in the measurement.

The maximum sampling rate for a supply voltage of 1.2V is determined to be 210MSPS by applying a 1MHz input signal and increasing the sampling rate until the SNDR drops off appreciably (Fig. 15). The steep drop seen above 210MSPS is due to parts of the digital circuit not having the new data ready to be latched by the D-flip-flops in the Wallace adder. The maximum sampling rate is very close to the specification of the target clock period given to the synthesizer. A shorter clock period could have been specified, but this clock period was chosen to limit the need for larger, more powerful gates in order to meet the overall area constraint.

Fig. 16 shows the spectrum of a 1MHz input and a sampling rate of 210MSPS (with 8x decimation of the output). Sine-wave SNDR is 35.89dB with a 41.46dB spurious-free dynamic range. To achieve this performance no calibration or post-processing was required. The inverse Guassian function on chip does all of the work of linearizing the output. The standard deviation of random comparator offset is measured, at a common-mode input voltage of 800mV and a 1.2V supply, to be about 46mV. The signal range, being between $\pm 1.6\sigma$ is about 280mV differentially. By using the range of $\pm 1.6\sigma$, almost 90% of all of the comparators lie within the signal range. Without linearizing the Gaussian distribution this would not be possible.

In a conventional flash ADC, decreasing the signal amplitude decreases the SNDR by decreasing the signal power while the noise power remains fixed. Measured random input-referred noise of this ADC is $293\mu V/\sigma$ while the effective LSB is 2.44mV. In a stochastic flash ADC, the stochastic requirement of 4^N comparators for N effective bits means that there are so many more comparators than a conventional flash ADC that decreasing the signal amplitude from full scale does not necessarily mean that the SNDR will decrease proportionally. This is actually a feature of a stochastic flash ADC. Measured data of SNDR from a sine-wave test as a function of input amplitude can be seen in Fig. 17. Over the range of input amplitude from 1σ to 1.6σ, the magnitude of the output in the code-domain is larger; however, the linearity is more or less constant. This could be especially valuable in cases where the magnitude of an incoming signal may not be fixed, but the required linearity is constant.

There is a significant load at the input (approximately 2.5pF) due to all of the comparator gate capacitances and the parasitic routing capacitance. This total capacitance is distributed and connected through resistive vias and metal traces that can not be considered negligible. Due to parasitic filtering of the input through the automatically synthesized input nets, there is an observed decrease in SNDR as a function of input frequency (Fig. 18). The -3dB bandwidth is measured to be around 87 MHz. This is predicted from extracted simulation of the RC

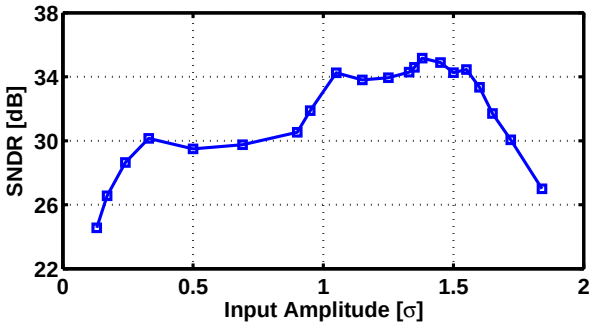


Fig. 17. SNDR from a sine-wave test as a function of input amplitude.

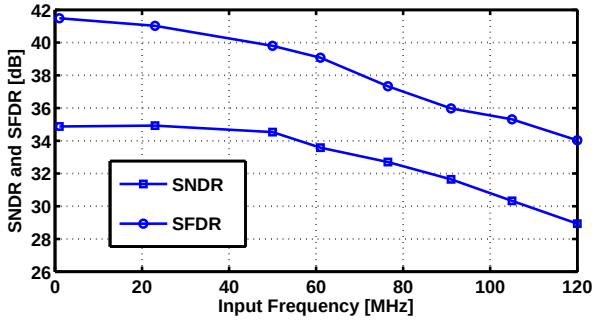


Fig. 18. SNDR and SFDR from a sine-wave test as a function of input frequency. The -3dB bandwidth at around 87-MHz is due to the RC filtering of the input network.

input network and taking the characteristic of Figure 17 into account. This filtering could be easily mitigated by placing the comparators together and having their inputs ported to higher metal, where wider metal could then connect all of the inputs at the top level. In this work, this was intentionally avoided in order for the authors to make the claim that the design is fully synthesized with no manual interference.

VII. Conclusion

This prototype IC proves that synthesizing an ADC entirely from Verilog is possible. The stochastic flash ADC naturally lends itself to being synthesized by the mere fact that it is designed to expect high variability; automated place-and-route will cause problems in many layout-sensitive designs, but not the stochastic flash ADC. A comparator that is implemented as two cross-coupled 3-input NAND gates has been demonstrated to work effectively as a true analog comparator. By using a piecewise linear approximation of the inverse function of a Gaussian CDF, 90% of the comparators of a single Gaussian group become an effective uniform distribution to the accuracy required. More importantly, this linearizing technique requires absolutely no calibration or post processing of any kind. The result is a truly all digital ADC with the only analog input being the input signal.

TABLE I
Performance Summary

Technology	90 nm CMOS	
Max Sampling Rate	210 MSPS	
Comparator Offset Standard Deviation	45 mV	
Input Range	280 mVpp (differential)	
Supply Voltage	1.2 V	700 mV
Sampling Rate	210 MSPS	21 MSPS
Input Frequency	1 MHz	1 MHz
SNDR	35.89 dB	34.61 dB
SFDR	41.46 dB	40.81 dB
Total Power	34.8 mW	1.11 mW
Total Active Area	0.18 mm^2	

Acknowledgment

This work was supported by the Semiconductor Research Corporation (SRC), and the Center for Design Analog-Digital Integrated Circuits (CDADIC).

References

[1] S. Wolf, *Silicon Processing for the VLSI Era: Deep-Submicron Process Technology*. Sunset Beach: Lattice Press, 2002.
[2] M. Shaker, S. Gosh, and M. Bayoumi, "A 1-gs/s 6-bit flash adc in 90 nm cmos," in *Circuits and Systems, 2009. MWSCAS '09. 52nd IEEE International Midwest Symposium on*, Aug. 2009, pp. 144 –147.
[3] H. Tang, H. Zhao, X. Wang, L. Lin, Q. Fang, J. Liu, A. Wang, S. Fan, B. Zhao, Z. Shi, and Y. Cheng, "Capacitive interpolated flash adc design technique," in *SoC Design Conference (ISOCC), 2010 International*, Nov. 2010, pp. 166 –169.
[4] J. Ceballos, I. Galton, and G. Temes, "Stochastic analog-to-digital conversion," in *48th Midwest Symp. on Circuits and Syst.*, Aug 2005, pp. 855–858.
[5] T. Sundsröm and A. Alvandpour, "Utilizing process variations for reference generation in a flash ADC," *IEEE Trans. Circuits Syst. II*, vol. 56, pp. 364–368, May 2009.
[6] D. Daly and A. Chandrakasan, "A 6b 0.2-to-0.9v highly digital flash ADC with comparator redundancy," *IEEE J. Solid-State Circuits*, vol. 44, no. 11, pp. 3030–3038, Nov 2009.
[7] C. Donovan and M. Flynn, "A 'digital' 6-bit ADC in 0.13 μm CMOS," *IEEE J. Solid-State Circuits*, vol. 37, no. 3, pp. 432–437, Mar 2002.
[8] K. Lakshmikumar, R. Hadaway, and M. Copeland, "Characterisation and modeling of mismatch in mos transistors for precision analog design," *Solid-State Circuits, IEEE Journal of*, vol. 21, no. 6, pp. 1057 – 1066, Dec 1986.
[9] S. Wong, K. Pan, and J. Ma, "A CMOS mismatch model and scaling effects," *IEEE Electron Device Letters*, vol. 18, no. 6, pp. 261–263, Jun 1997.
[10] H. Stark and J. Woods, *Probability and Random Processes with Applications to Signal Processing*. Upper Saddle River: Prentice Hall, 2001, ch. 4, pp. 225–230.
[11] ——, *Probability and Random Processes with Applications to Signal Processing*. Upper Saddle River: Prentice Hall, 2001, ch. 1, pp. 40–41.
[12] S. Weaver *et al.*, "Stochastic flash analog-to-digital conversion," *IEEE Trans. Circuits Syst. I*, vol. 57, pp. 2825–2833, Nov 2010.
[13] S. Weaver, B. Hershberg, and U.-K. Moon, "Enob calculation for adcs with input-correlated quantization error using a sine-wave test," in *Microelectronics (ICM), 2010 International Conference on*, Dec. 2010, pp. 5 –8.
[14] A. D. Inc., *Data Conversion Handbook*. Burlington: Newnes, 2005, ch. 2, p. 69.
[15] F. Kaess *et al.*, "New encoding scheme for high-speed flash ADC's," in *Int. Conf. on Solid-State Integrated Circuit Tech.*, Jun 1997, pp. 5–8.

Skyler Weaver (S'06–M'11) received his H.B.S. degree in electrical engineering from Oregon State University in 2006. In 2010, he also received his Ph.D. in electrical engineering from Oregon State University for work in highly scalable and synthesizable ADCs. He is currently working with Intel Corporation in Hillsboro, Oregon researching and designing scalable mixed signal circuits for future applications. Skyler's current focus is in forward-looking high-speed serial links.

Benjamin Hershberg (S'06–M'12) received H.B.S. degrees in Electrical Engineering and Computer Engineering from Oregon State University in 2006. He earned the Ph.D. degree in Electrical Engineering from Oregon State University in 2012 for his work in scalable, low-power switched-capacitor amplification solutions, including the techniques of Ring Amplification and Split-CLS. Currently, Benjamin is at the Interuniversity Microelectronics Center (imec) in Leuven, Belgium working on software-defined reconfigurable radio analog frontends.

Un-Ku Moon (S'92–M'94–SM'99–F'09) received the B.S. degree from the University of Washington, Seattle, in 1987, the M.Eng. degree from Cornell University, Ithaca, NY, in 1989, and the Ph.D. degree from the University of Illinois at Urbana-Champaign in 1994.

He has been with the School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, since 1998. Before joining Oregon State University, he was with Bell Laboratories from 1988 to 1989, and from 1994 to 1998. His technical contributions have been in the area of analog and mixed-signal circuits including high linearity filters, timing recovery, PLLs, data converters, and low voltage circuits for CMOS.

He is the Editor-in-Chief of the IEEE JOURNAL OF SOLID-STATE CIRCUITS. He has served as a Distinguished Lecturer of the IEEE Solid-State Circuits Society, as an Associate Editor of the IEEE JOURNAL OF SOLID-STATE CIRCUITS and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II, as Editor-in-Chief and Deputy Editor-in-Chief of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II, as IEEE Solid-State Circuits Society representative to IEEE Circuits and Systems Society, and as a Technical Program Committee member of the IEEE International Solid-State Circuits Conference, IEEE VLSI Circuits Symposium and the IEEE Custom Integrated Circuits Conference.