A Level-Crossing Flash Asynchronous Analog-to-Digital Converter

Filipp Akopyan, Rajit Manohar, Alyssa B. Apsel*
Computer Systems Laboratory
Electrical and Computer Engineering
Cornell University
Ithaca, NY 14853, U.S.A.

Abstract

Distributed sensor networks, human body implants, and hand-held electronics have tight energy budgets that necessitate low power circuits. Most of these devices include an analog-to-digital converter (ADC) to process analog signals from the physical world. We describe a new topology for an asynchronous analog-to-digital converter, dubbed LCF-ADC, that has several major advantages over previously-designed ADCs, including reduced energy consumption and/or a simplification of the analog circuits required for its implementation.

In this paper we describe the design of the LCF-ADC architecture, and present simulation results that show low power consumption. We discuss both theoretical considerations that determine the performance of our ADC as well as a proposed implementation. Comparisons with previously designed asynchronous analog-to-digital converters show the benefits of the LCF-ADC architecture. In 180 nm CMOS, our ADC is expected to consume 43 μW at 160 kHz, and 438 μW at 5 MHz.

1 Introduction

There is significant interest in ultra low power circuits today due to their applications in embedded sensing systems. Such systems may be deployed for environmental monitoring applications, or may be implanted into human body for health and brain activity monitoring. The major requirement for these circuits is that they must be able to perform the necessary application-specific analysis while running on small portable energy sources. A high-level overview of such embedded systems is shown in Figure 1. Common sensors include devices that measure temperature, pressure, vibration, or acceleration. The output of the system could be transmitted over a wireless link, or could be used in-situ depending on the nature of the application.

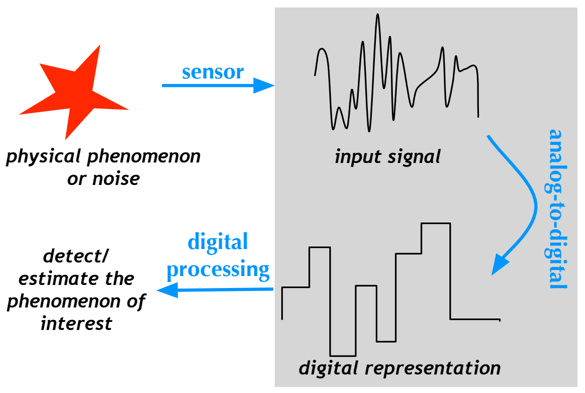

Figure 1. Data flow in an embedded sensing system

An analog-to-digital converter (ADC) is an integral part of the overall system, as it enables the sensor to interface with a low-power processor. The sensor networking processor's power consumption may be on the order of nW [4]. Ensuring that the ADC's power consumption is low, is an important part of reducing the overall system's power consumption.

Input Signal Considerations. Conventional signal processing techniques are based on Nyquist sampling. In such systems, the samples are taken periodically, with the clock set at least at twice the maximum frequency of the input signal. In most cases, the clock frequency is higher than twice the maximum frequency and the converters oversample the input signal to increase the effective precision. Such circuits consume a constant amount of power even if there is no change in the input signal [9].

A wide class of measured signals have the property that the signals don't change rapidly; they are constant for some time, then change their value and return back to idle state. Consequently, a lot of power in a Nyquist sampling ADC is wasted; in signal processing terms, the signal is non-

*E-mail: {filipp,rajit,apsel}@csl.cornell.edu stationary and therefore the optimal sampling rate should adapt based on the signal characteristics. For the purpose of eliminating the circuit power consumption when the input is stable, we use two existing techniques to design the ADC: (i) we adopt an asynchronous event-driven circuit approach, these circuits idle when there is no change at the input; (ii) we adopt an implicit sampling approach, also known as level-crossing sampling.

Sampling scheme. The level-crossing scheme is non-uniform in time domain; it is based on the concept of predetermined reference levels, where a sample is taken only if a reference level is crossed. The samples are not taken at a constant rate, but are only obtained if there is a sufficient change in the input analog signal. A simple version of this approach that only detects the sign of the input signal (zero-crossing sampling) has been extensively studied by the signal processing community beginning with the work of Rice [23]. Kedem provides an excellent overview of the work in this area [10]. The concept of level-crossing sampling has been previously studied as a method of sampling utilized in analog-to-digital conversion and in data compression [18, 25]. More recently, Renaudin et al. developed an ADC using the level-crossing approach [2].

Contributions. We present a new topology for an asynchronous level-crossing-based ADC (LCF-ADC). Unlike Renaudin's approach, the time in the LCF-ADC is not tracked explicitly. Each level-crossing event is represented by a one-bit data token produced at the output. The value "1" indicates that the input signal has crossed the given level from below; the value "0" indicates that the input signal has crossed the level from above. Using the obtained bit sequence (differential encoding: only 1-s and 0-s), downstream processing of the binary stream can be implemented [11]. This type of encoding is known as asynchronous delta modulation; it was proposed by Watanabe [8] and was previously used by Tsividis [14, 15]. In addition, the LCF-ADC can be configured in such a way that it detects the situation when the input signal exceeds its design parameters, and this detection can be used to activate a more sophisticated ADC if necessary. As stated in the abstract, in 180 nm CMOS, our ADC is expected to consume 43 μW at 160 kHz, and 438 μW at 5 MHz.

Related Work. There is an enormous body of literature on ADCs, including several books on the subject [9]. A comparison of using synchronous and asynchronous methods [13] for ADCs studied the effect of using asynchronous logic in ADC implementations; previously, a flash-type ADC was implemented using micropipelines [12].

Recently, several schemes utilizing level-crossing were developed [1, 2]. However, the goal of those designs is signal reconstruction. Those systems record sample time and reconstruct the original signal to perform signal processing operations. Instead, our idea is to process the level crossings *without* reconstructing the original input. As shown by Kedem [11], such processing is possible by directly formulating detection and estimation problems with the implicit signal representation. Tsividis has demonstrated advantages of continuous time signal processing utilizing only a quantizer [27].

In our architecture we *do not* record the times at which the samples are taken. This technique reduces the power consumption even further by eliminating the circuitry that deals with time-tracking. In case we need to reconstruct the original signal, the time information is still embedded in the relative delays of the output samples. This information can be extracted at any time if necessary, by the means of a simple digital counter (and calculating the time elapsed since the previous sample has appeared at the output), which can be easily added to our circuit. Our approach adopts a completely different ADC topology: instead of using the feedback-based approach adopted by Renaudin (and, following Renaudin, by Shepard [14]), we use a parallel architecture that resembles a flash-type ADC.

2 LCF-ADC Architecture

The LCF-ADC architecture is a mixed-signal circuit, containing analog parts and asynchronous digital parts. For the design of the asynchronous digital circuits we use the quasi-delay-insensitive (QDI) style [20] due to its robustness to process, voltage, and temperature variations. The circuits are implemented using Martin's synthesis method [19] that translates a high-level design description to circuits through handshaking expansions, and production rules.

When designing the LCF-ADC, our main goal was to minimize the ADC's power consumption, while still maintaining a reasonable frequency of operation. Our LCF-ADC was optimized for the input signal bandwidth of 5 MHz. We chose this requirement based upon a survey that examines bandwidths of commercially available sensors including pressure sensors, temperature sensors, and accelerometers.

The proposed ADC structure consists of the following main components, shown in Figure 2. The analog comparators compare the analog input signal to corresponding reference levels. Voltage controlled regenerative comparators were used to increase noise immunity of the ADC as described in Section 3. The comparator's output value, as well as the internal state variables of the digital circuit, control the digital trigger that sends a request to *process* the sampled data. As soon as the request becomes active, it is processed by the asynchronous digital logic that outputs one

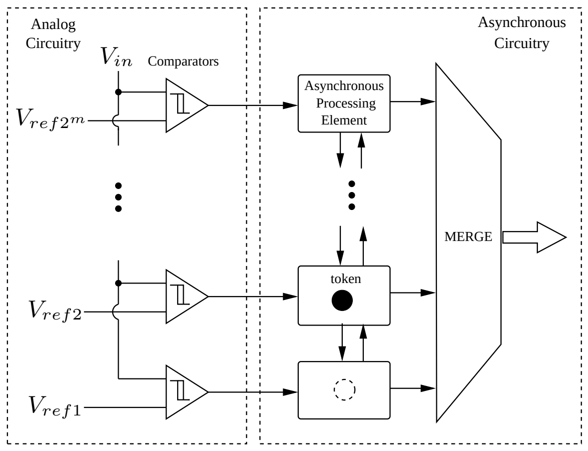

Figure 2. Proposed ADC Structure

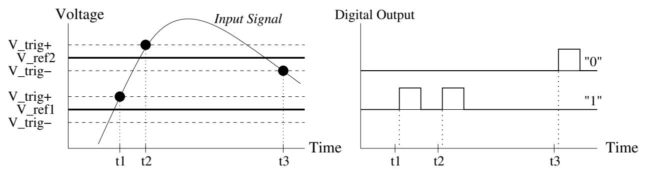

Figure 3. Output of the LCF-ADC

bit. Each bit on the output represents a level-crossing. If the level was crossed by the input signal from below, the circuit sends a “1” on a dual-rail output channel, and if the input signal crossed the level from above, the circuit outputs “0” on the same dual rail channel, as shown in Figure 3.

The levels of the LCF-ADC are specified by a resistive or capacitive divider, as described later. Once a level is crossed by the input signal, the comparator changes its output. If the signal was above and crossed down, the output of the comparator changes from high to low and vice versa. As soon as the digital trigger identifies the change in the comparator’s output, it checks the states of the previous and next asynchronous processing elements and if all the variables indicate that the crossing has occurred and no conditions were violated, the trigger sends a request to the asynchronous logic.

The asynchronous processing element checks whether it has the permission to output the value corresponding to the crossing, and as soon as the permission is granted, it outputs a “0” or a “1.” Mutual exclusion is maintained by using a token-based scheme implemented in the asynchronous digital part.

Operation continues until $Reset$ occurs or until the system is shut down. All the processing elements except the top and the bottom elements function identically. The precision of the ADC can be increased by replicating the comparator, processing element, digital trigger parts and scaling the output merge appropriately. The replicated nature of the ADC also simplifies the physical design of the overall architecture.

3 Design of the LCF-ADC

The LCF-ADC circuits have three major parts: the analog circuits, the digital trigger that interfaces between the analog circuits and the digital parts, and the asynchronous digital circuits that include the processing element and the output merge.

3.1 Analog Circuitry

Voltage Divider. We have designed our voltage divider to generate reference voltages for a set of comparators with high input impedances. As a result, no current is drawn from the divider circuit nodes; and either a capacitive or a resistive divider may be used. We choose to use a capacitive divider to minimize power drawn from the supply. If necessary, the levels on the divider can be periodically reset using series transistors between those nodes. The resistive divider approach is more suitable when using aggressively scaled process technologies that have substantial gate leakage, or lack linear capacitors.

Regenerative Comparator. To prevent spurious outputs, the comparator must be able to suppress a noisy input signal so that changes are not detected when the input is fluctuating at the reference level. The comparator must operate at least at 5 MHz (higher than the maximum frequency of operation of the targeted sensors and implants).

In our application, the comparator is used to convert a slowly changing input signal into digital logic levels with sharp edges. If the environment is noisy (meaning the signal is fluctuating), or if the noise from the rest of the circuit is high, several problems can arise. If the response time of the comparator is faster than the variation of the input signal around the threshold level, the output will chatter between ’high’ and ’low’ logic values as the input crosses the threshold. This output variation can be eliminated by employing positive feedback that forces a comparator to behave as a regenerative latch. In this case, the circuit will exhibit *hysteresis*. In our design we have decided to implement comparators with hysteresis to tolerate noisy inputs.

The differential regenerative comparator with hysteresis utilized in our design is presented in Figure 4 and is explained by Gregorian [5]. Similar topologies were previously utilized in ADC designs [24, 26]. The major differences between a common Schmitt trigger and the configuration used in our design will be discussed.

The differential topology uses only eleven transistors, and no resistors, unlike the common Schmitt trigger con-

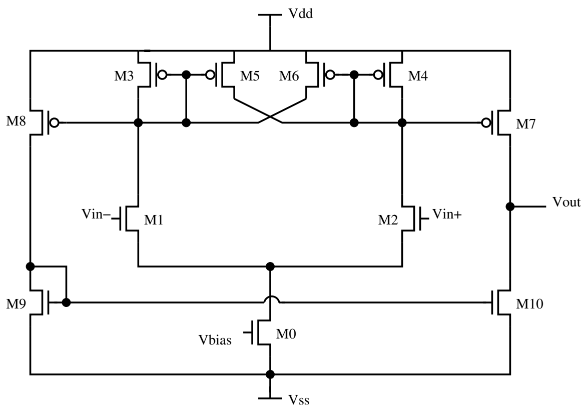

Figure 4. Regenerative Comparator

figuration. The design is based on the differential pair M1 and M2, the bias transistor M0, load devices M3 and M4, and the cross-coupled pair M5 and M6. The only purpose of the current mirror and transistors M7, M8, M9, M10 is to provide additional gain for sharper output edges and to convert the differential output to a single-ended output. The bias transistor M0 sets the current flowing through the circuit. The higher the current, the more power is consumed as the output impedance is lowered and higher bandwidth may be achieved. Power consumption can be traded off for bandwidth of the comparator to some extent depending on the application of the LCF-ADC. M3 and M4, along with g_m of M1 and M2, set the gain of the first stage: the larger the load devices, the higher the gain is and the more parasitic capacitance is introduced by these devices. M5 and M6 provide positive feedback and determine the amount of hysteresis for the comparator.

High precision operation may be achieved only if the transistors are properly matched. Otherwise, the trip points of the circuit, as well as the gain and bandwidth may be shifted. Good matching is achieved by proper transistor sizing and layout techniques. The bandwidth of the comparator may be improved by increasing the transistor widths of M1 and M2 as the load devices are shrunk (to maintain the same gain of the comparator), or as mentioned before, by increasing the bias voltage at the gate of M0, at the cost of increased power consumption. The trip points of the circuit are mainly determined by the ratio of M5 to M3 (M6 to M4 have to be matched). The differential gain is large due to relatively high output resistance of the circuit.

The full derivation of the comparator's trip points, as well as the stability conditions are presented in Gregorian's work [5]. However, we mention the formulas that determine the trip points of the comparator for completeness, since these values specify the amount of hysteresis present in the circuit. If we label the currents through M1 and M2, as i_1 and i_2 respectively and their sum as i_0, it can be shown that the switching points can be calculated as follows:

$$V_{trig+} = \sqrt{\frac{i_0}{K'(W/L)_1}} \cdot \frac{\sqrt{\alpha}-1}{\sqrt{1+\alpha}}$$
$$V_{trig-} = \sqrt{\frac{i_0}{K'(W/L)_1}} \cdot \frac{1-\sqrt{\alpha}}{\sqrt{1+\alpha}}$$
$$\alpha = \frac{(W/L)_5}{(W/L)_3} = \frac{(W/L)_6}{(W/L)_4}$$
$$K' = \frac{1}{2} \cdot \mu \times C_{ox}$$

In these formulas,

W/L - transistor width-to-length ratio

C_{ox} - oxide capacitance

μ - mobility of holes

α - positive feedback factor

The hysteresis, as mentioned before, is a function of the ratio of the sizes of the cross-coupled to the load transistors, as well as the sizes of the input transistors. In order to obtain regenerative behavior the value of α has to be greater than 1. In our design, α was chosen to be 1.33 to obtain a region around the reference voltage that rejects noise in the input signal. The value of α should be selected depending on the application of the LCF-ADC (depending on the noise level of the environment). Under normal operation, the amount of hysteresis on one side of the reference level, should not exceed half of the voltage difference between the two adjacent bits (i.e. LSB and the next bit).

The positive voltage supply used in the circuit is 1.8 V and the negative voltage supply has to be set to -400 mV in order to obtain the full range of 1.8 V at the input of the comparators (to be able to have voltages down to 0 Volts as reference levels). Otherwise, if the negative voltage supply is just GND, then the lowest value that the comparator can recognize as a reference voltage is approximately 400 mV, which limits the input range of the ADC. If the negative voltage supply is undesired, the NMOS and PMOS pairs may be flipped and a single voltage supply of 2.2 V will then be used.

The transistors in the comparator must be sized carefully. Transistors that are too small are susceptible to poor matching, which can result in nonlinearities in the output. Transistors that are too large present significant parasitic capacitive loads which may limit the speed of the circuit. There is a natural tradeoff between the bandwidth of the comparator and transistor matching (precision). We have chosen transistors that are several times bigger than the minimal size transistors to obtain the necessary bandwidth and to provide reliable matching. In general, if lengths of transistors are chosen to be greater than 1 μm, the threshold voltages of the transistors are considered to be well-matched.

The current through the comparators can be varied externally if desired, since we propose to route the bias voltage node to the outside of the chip in order to be able to control the bandwidths (and gains) of comparators. One external control signal can be used to tune the bandwidths of all comparators identically.

3.2 Digital Circuitry

The main purpose of the asynchronous logic in the LCF-ADC is to ensure the proper recognition of level crossings and to output the valid data. We use serial data transfer for the purpose of minimizing the number of wires that have to connect to our ADC. The output of the LCF-ADC represents the change of the input signal with respect to the previous sample and not the complete n-bit sample. This scheme decreases the bandwidth required to send the output and is advantageous from the data compression perspective.

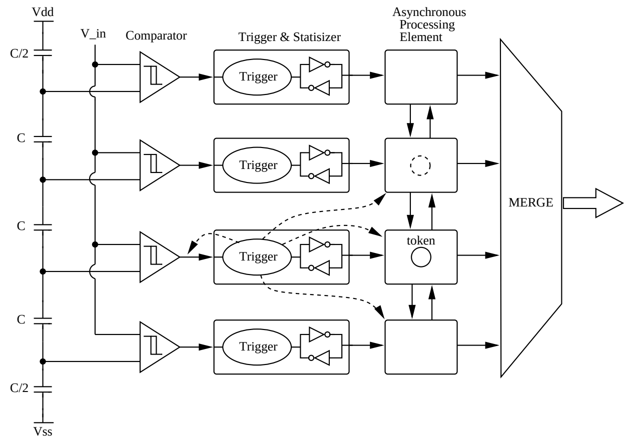

Figure 5. Overall Structure of the LCF-ADC

The overall detailed diagram of a two-bit (four-level) LCF-ADC is presented in Figure 5. As described in Section 3.1, a capacitive divider is used to set the reference voltages. The analog comparators detect whether the input signal is above or below the set threshold. The output of each comparator is connected to a digital trigger. The trigger checks the state of the two neighboring asynchronous processing elements to determine whether the reference level was crossed or not. If the crossing condition (as described later in this section) is true, the trigger performs a handshake on a dataless channel connected to its corresponding asynchronous processing element which then generates the appropriate output. The functionality of the asynchronous processing element will be described in the subsequent section. After processing the sample, the asynchronous element outputs the direction of the crossing to the output merge, which combines all its exclusive inputs into a single output channel.

The digital circuitry initializes (and resets) in a state where no requests are sent by the trigger and all the communication channels of the processing element are in known states. The initial state is achieved by the means of reset transistors connected to the asynchronous digital gates. The design also allows the ADC to track the signal if on startup the input is somewhere between the first and the last reference levels. The same tracking is utilized if the maximum frequency of the ADC is exceeded.

Asynchronous Processing Element. The basic structure of the asynchronous processing element is shown in Figure 6. Each element has two *local variables*: k and t. The variable t indicates whether the given element has the token (there is only one token in the system); possession of which gives permission to *this* element to output data. If the token is present, k indicates whether the input signal is above or below the given level. In the case of absence of the token, k indicates whether the token is above or below *this* element.

Each processing element has 5 external channels. Channel "INPUT" represents the dataless channel from the digital trigger. The "OUTPUT" channel is used to output data to the merge. Channel "RECEIVE from ABOVE" gets activated when the element above requests the token; the given element then passes the token up. Similarly, channel "RECEIVE from BELOW" gets activated when the element below requests the token; the given element then passes the token down. The other two channels "SEND UP" and "SEND DOWN" are used if the given element needs to request the token from the neighboring elements; the adjacent element then passes the token to the given element. At all times presence of the token represents the last level crossed.

By design of the LCF-ADC, the token cannot skip elements, thus if a request is obtained by the processing element from the trigger and the given element does not have the token, the token is either immediately above or immediately below the given element depending on the value of k (due to the design of the digital trigger). The complete high-level description of a processing element (in CHP notation, summarized in the Appendix) is presented in Program 1.

The variable legend is as follows:
A - request from the trigger
D - request for the token from below
U - request for the token from above
t - variable that is true if the token is present

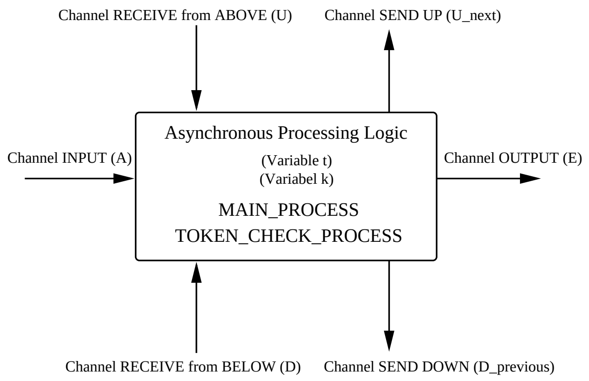
Channel RECEIVE from ABOVE (U)
Channel SEND UP (U_next)
Asynchronous Processing Logic
Channel INPUT (A)
(Variable t)
(Variabel k)
Channel OUTPUT (E)
MAIN_PROCESS
TOKEN_CHECK_PROCESS
Channel RECEIVE from BELOW (D)
Channel SEND DOWN (D_previous)

Figure 6. Asynchronous Processing Unit

Program 1 CHP: LCF-ADC, One Processing Element

```
$MAIN\_PROCESS \equiv$
*[[ $\overline{A} \longrightarrow Y; t\uparrow; E!(\neg k); k := \neg k; A$
 ▯ $\overline{D} \longrightarrow D;\ t\downarrow;\ k\downarrow$
 ▯ $\overline{U} \longrightarrow U;\ t\downarrow;\ k\uparrow$
]]

$TOKEN\_CHECK\_PROCESS \equiv$
*[[$\overline{Y} \wedge \neg t \longrightarrow$ [$k \longrightarrow U\_next$ ▯ $\neg k \longrightarrow D\_previous$]
 ▯$\overline{Y} \wedge t \longrightarrow$ $skip$
 ]; $Y$
]
```

k - true if token is above, false if below
E - output channel that indicates direction of crossing
Y - internal channel used for a “process call”

Each level of the LCF-ADC starts out in an inactive mode with all variables initialized to predetermined values. Specifically, variable t is 'false' in all processing elements except the bottom one. In the bottom element, t is initialized to value 'true', since there is only one token in the system. The variable k is initialized to a low value in all processing elements.

The two processes in the CHP that are executed infinite number of times are: TOKEN_CHECK_PROCESS and MAIN_PROCESS. The MAIN_PROCESS performs a deterministic selection between the input probes, which can only become true one at a time (probes have to be mutually exclusive). As soon as one of the request probes becomes true, one of the following holds: either the trigger indicates that the level was crossed, or the element above requests for the token, or the element below requests for the token.

If the trigger indicates that the level was crossed, the request is generated and TOKEN_CHECK_PROCESS is started to verify the presence of the token on the given level. If the token is not present (t is false), the element sends the request to the neighboring element, immediately above or immediately below the given element depending on the value of k. If k is true the request is sent up; if k is false the request is sent down. If the token is present (t is true), the execution of the selection is terminated with a *skip*.

When the TOKEN_CHECK_PROCESS is finished, the element “knows” that the token is present on its own level, thus, the value of t becomes true (if it was not true before). At this point (since there is only one token in the system), it is safe to produce an output. A value of '1' at the output indicates that the level was crossed up and the value '0' indicates that the level was crossed down. This bit assignment is identical to sending the value of $\neg k$.

After the output is sent, the value of k is inverted, indicating that the signal has crossed the threshold in either upward or downward direction (depending on the previous value of k). After that action is finished, the handshake on channel A is completed indicating that the element is done processing the input request. The processing element is now ready for the next request.

If the D probe becomes true, the element below the given element is requesting the token. The communication on D is performed right away to speed up the token transfer. The value of t on the element is lowered, indicating that it does not have the token any more. The value of k becomes false indicating that the token is below this level.

If the U probe becomes true, the element above the given element is requesting the token. The communication on U is performed and the value of t on the element is lowered, indicating that it does not have the token any more. The value of k becomes true indicating that the token is above this level.

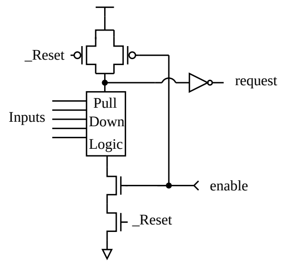

Figure 7. Digital Trigger Implementation

Digital Trigger. The circuit that wakes up the asynchronous digital processing element is the digital trigger. The trigger is constructed as shown in Figure 7.

The $\_Reset$ signal is used for resetting (usually initiated off-chip by the user); this signal initializes the trigger in a state with no active requests. The two transistors that have $enable$ (the inverted sense of the acknowledge of the dataless channel connecting the trigger to the processing element) connected to their gates indicate that the previous request was properly served. The $request$ signal comes out of the trigger through the inverter. The trigger becomes active if one of the following conditions is satisfied, where the following signals are the $Inputs$ indicated on the diagram:

1. (i)-th comparator's output is high, and (i)-th processing element indicates that the input signal was below before, and (i-1)-th level indicates that the input signal is above, and (i-1)-th element has completed processing its request;

2. (i)-th comparator's output is low, and (i)-th processing element indicates that the input signal was above before, and (i+1)-th level indicates that the input signal is

below, and (i+1)-th element has completed processing its request.

These rules for request generation enable tracking of the input at startup and during violation of the maximum allowable frequency.

The output of the trigger is latched by a staticizer, since the output of the comparator can change spontaneously. However, the request to the asynchronous processing unit must remain stable, even in the face of a changing analog input signal. The request stays active until the processing unit finishes serving that request.

Once the trigger sends the request to the processing unit, the request is considered to be served when the output of the processing unit is sent to the environment. The request is then cleared automatically by the $enable$ signal of the trigger.

Output merge. The output merge is a standard deterministic merge (no arbiter required) between all the one-bit channels, and is implemented in the standard way [17]. The CHP description of this process is:

$$*[[\overline{E_0} \longrightarrow E!(E_0?) \;[]\; \ldots \;[]\overline{E_{n-1}} \longrightarrow E!(E_{n-1}?)]]$$

4 Power Analysis

During the period of inactivity in the input signal, none of the levels are crossed and thus, the outputs of analog comparators remain constant. The triggers discussed in the previous section do not send any requests to the asynchronous processing elements. The asynchronous elements stay in the "idle" mode until the level corresponding to this element is crossed. The only power that is consumed by the asynchronous elements is the leakage power.

Comparator array. In order to minimize the power consumption of the analog comparators, we have used the following approach. At any moment, only two comparators are *on*, meaning the power is supplied to only two comparators. Note that this is very different from a conventional flash ADC, where all the comparators must be turned on for the ADC to operate.

Since the input signal is always between two levels, the comparators corresponding to those two levels are the ones enabled. Each comparator can determine whether or not it should be enabled by examining the state of the corresponding asynchronous processing element and its adjacent elements in the array. We use an additional circuit that generates a $sleep$ signal that is used in the digital logic to control the comparator's power supply. The power to the comparator is supplied if at least one of the following holds, where (i), (i-1) and (i+1) represent instances of the current, the previous and the next levels respectively:

1. The signal is above the (i-1) processing element and below this (i) processing element;
2. The signal is above this (i) processing element and below the (i+1) processing element;
3. The request is being processed by the current element (this condition is needed to avoid a glitch on the comparator's power signal during the period when the value of k is toggling).

Thus, if $sleep$ signal is 'true', the power should be cut-off, otherwise the power should be on. The circuit that controls the power to a comparator has to also *disconnect* the output of the comparator when the power is turned off to preserve the state of the trigger. When the power of the comparator is off, the output signal will slowly drop to -0.4V; as a result, the digital trigger may send a false request to the processing element. In order to avoid that, we put a transmission gate at the output of the comparator that isolates the comparator from the digital trigger when the comparator's power is off. This circuit is shown in Figure 8. However, when the transmission gate disconnects the comparator's output from the trigger, we need to store the last value outputted by the comparator in order to provide correct information to the trigger. This is done by the means shown in Figure 8. The "modified staticizer" keeps state only when the transmission gate is not conducting. For this reason the signals that open and close the transmission gate also control the staticizer's feedback.

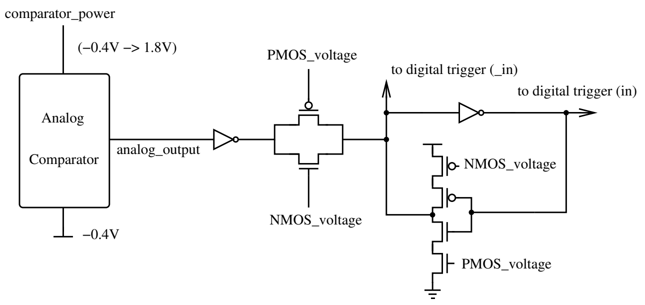

Figure 8. Comparator Output Latch

The circuit that generates the signals $comparator\_power$, $PMOS\_voltage$, and $NMOS\_voltage$ shown in Figure 8 needs to perform several functions. The circuit has to be "edge-sensitive", i.e. behave differently depending on whether the sleep signal changes from '0' to '1', or '1' to '0'.

If the $sleep$ signal was false and becomes true; first, the transmission gate stops conducting the output of the comparator to the digital trigger and the last value of the comparator's output is stored by the staticizer. Next, the power to the comparator turns off after some delay. This delay is minimal and its implementation is described in the next section. The delay allows the staticizer to store the correct value of the comparator's output.

If the $sleep$ signal was true and changes to false, the comparator's power needs to turn on first. After some delay, the transmission gate starts conducting the comparator's output to the digital trigger and the staticizer's feedback is turned off. The required delay in this case is longer in order to let the comparator's output settle to the correct value before this value is passed to the digital trigger.

This scheme drastically decreases the power consumption of the LCF-ADC. Instead of having all 2^n comparators working constantly, we have 2 comparators powered-up both in the normal mode of operation and in the idle mode. This property leads to a reduction in power consumption of the ADC.

5 Evaluation

There are a number of considerations when designing the LCF-ADC. We discuss the theoretical signal processing aspects of the sampling scheme, followed by an evaluation of the LCF-ADC design in terms of its power consumption.

5.1 Theoretical Considerations

Level-crossing sampling is an extension of the simple zero-crossing sampling concept. An analysis of zero-crossing originated in the work of S.O. Rice [23]. Logan analyzed the information content in the zero-crossings of bandlimited signals, and argued that such signals can be reconstructed from their zero-crossings up to a scale if they do not share any zeros with their Hilbert transform [16]. The work of Beutler rigorously establishes the result that a bandlimited signal can be reconstructed from samples as long as the average number of samples exceeds the Nyquist rate [3].

It is interesting to note the theoretical benefits as well as limitations of a level-crossing ADC. A conventional uniform sampling ADC has two limitations: (i) The circuits for the ADC have a finite bandwidth B; (ii) The sampling clock has a fixed frequency that induces a second limitation F_s on the input signal. If the input signal exceeds bandwidth F_s, the ADC output exhibits aliasing since the replica copies of the signal in the frequency domain will overlap within the band specified by F_s [22]. Instead, a level-crossing ADC only imposes a limit based on the bandwidth B rather than an externally imposed sampling frequency. Therefore, aliasing cannot occur unless the requirements of the circuits of the level-crossing ADC are exceeded. One way to prevent this from occurring is by introducing an analog filter at the input to the level-crossing ADC.

A level-crossing ADC also effectively quantizes the output. A conventional ADC that quantizes the output introduces additional noise (the classic $\Delta^2/12$ quantization error power, where Δ is the spacing between the levels). However, the absence of sampling significantly reduces the in-band noise for a level-crossing ADC [27].

Any realization of a level-crossing ADC implementation has an upper bound M on the maximum number of output samples in a fixed time interval T. (Lifting this assumption is analogous to having an infinite sampling rate in a conventional Nyquist-rate ADC.) A bandlimited signal $x(t)$ can be expanded in the $sinc$ basis as $\sum_{k=-\infty}^{\infty} x_k \text{sinc}((t - kW)/W)$ with $W = 2\pi/\omega_s$ where ω_s is the maximum frequency component in the signal, and x_k are the coefficients that characterize $x(t)$. The coefficient x_k is the sample of the signal at time kW, sometimes denoted $x[kW]$. By Nyquist we know that we need at least T/W samples to be able to reconstruct the original signal. Since there is an upper bound M on how many samples a level-crossing ADC will produce in a given amount of time (which directly determines the bandwidth B), we can reduce W where $T/W > M$. At that point, a level-crossing ADC will not provide a sufficient number of samples to reconstruct the original signal, resulting in erroneous reconstruction.

However, a level-crossing ADC does have the benefit of being *bandwidth-adaptive*—the number of samples it produces is a function of the local Nyquist rate of the input. For example, the number of zero-crossings of a stationary Gaussian process is given by $1/\pi\sqrt{-\rho''(0)}$, where $\rho(\tau)$ is the correlation coefficient of the process, and we know that $-1/\pi^2\rho''(0) = \frac{\int_{-\infty}^{\infty} 4f^2 S(f)\,df}{\int_{-\infty}^{\infty} S(f)\,df}$, which is the normalized second moment of the power spectral density S. The square root of this quantity is a measure of the support of $S(f)$ in the frequency domain and is upper bounded by $2f_{max}$, and so on average the number of crossings should be of the same order as the Nyquist rate [7].

5.2 Design Evaluation

As mentioned earlier, it is important to make sure that the comparators in the LCF-ADC have the bandwidths of greater than or at least equal to the frequency with which the requests will be processed.

Our circuit was simulated using Nanosim and HSpice analog simulators in the TSMC 0.18 μm process. The maximum throughput that was achieved for the asynchronous part alone with minimal transistor sizing was 220 MHz. This sizing does not provide a shortest possible delay, but provides minimal power consumption. At the frequency of 220 MHz, the average power consumption of the asynchronous circuitry is approximately 280 μW. As we will show, the maximum throughput limits the frequency of the input signal. If desired, transistors can be resized for greater performance (higher throughput, smaller latency) at the expense of higher power consumption. However, 220 MHz is sufficient to meet our design requirements.

For a full swing signal, the maximum number of crossings that can be correctly interpreted by our LCF-ADC is determined by the signal bandwidth (BW) and the maximum throughput of the asynchronous circuitry (f_{max}):

$$BW \cdot \frac{\text{number of events}}{\text{cycle}} \leq f_{max}$$

Where the 'number of events' represents the number of crossings, i.e. the number of requests issued to the asynchronous processing element by the digital trigger.

If we assume a full swing periodic input signal with $2n$ crossings in one period (like a triangular wave or a sinusoid), the equation becomes:

$$BW \cdot 2n \leq f_{max}$$

Thus, with f_{max} for minimal power consumption at around 220MHz, the maximum number of levels that can be correctly interpreted is:

$$n \leq \frac{110}{\text{BW}}$$

with BW as the bandwidth of the input analog signal in MHz.

According to the above relationship, the maximum number of reference levels that can still be properly interpreted by the converter with a full swing input signal of 1 MHz is 110 levels, which corresponds to a 6 bit converter. If higher precision in terms of the number of bits is desired, the speed of the asynchronous circuitry has to be increased.

In our simulations, the LCF-ADC was optimized for a 5 MHz full swing input signal. In order to achieve this bandwidth and still have a short conversion time of the sample, the bias voltage of the comparators was set to the value of 0.25 V. The delays in the logic that control the pass gate and the comparator power were implemented in the following way. The comparator power-off delay was implemented by two minimally sized inverters, because the required delay there is very short (just enough to disconnect the transmission gate and latch the correct value). However, the delay that controls the $NMOS\_voltage$ and $PMOS\_voltage$ signals and allows the transmission gate to conduct, needs to be much longer. Once the power of the converter is on, sufficient time has to be given to the comparator in order for its output to settle to the correct value. This delay was implemented by an RC network. The delay at the chosen bias voltage is approximately 3 ns.

For all simulations a 4-bit converter (16 levels) was examined (for comparison with the current topologies, however this topology may be trivially extended to a higher-precision ADC). The layout area of the LCF-ADC increases linearly with the number of levels. An input sinusoid of various frequencies with full swing was used to test the LCF-ADC. In the simulations, we have used TSMC 0.18 μm process transistor models. The transistors were sized minimally for minimal power consumption, except for the transistors in the analog comparators (in the comparators the transistors were on the order of μm to provide better matching). The RC parameters for the transmission gate delay were chosen according to the specification above. The capacitive divider was implemented as a resistive divider for the simulation purposes, because the circuit simulator does not model capacitor's physical effects. The resistor values were maximized to decrease the power consumption, each resistor had the value of $R_{divider} = 1M\Omega$. All the simulation data is presented in the table below.

LCF-ADC Simulation Data

Signal BW	Power (μW)	Energy/sample (nJ)
1 kHz	34.41	34.4
100 kHz	42.48	0.42
114 kHz	43.57	0.38
160 kHz	46.84	0.29
1 MHz	114.14	0.11
5 MHz	437.81	0.087

The dynamic power of the LCF-ADC at any frequency in the operating range is approximately linearly proportional to the dynamic power at the reported frequencies.

At lower frequencies (less then 100 kHz) most of the power consumption is quiescent (approximately 17 μW); it comes from the two comparators that are constantly on (the power consumption of the resistive divider is negligible). Consequentially, each sample accumulates more energy at lower frequencies than at higher frequencies. So if the LCF-ADC is designed for low frequency operation (less than 5 MHz), the comparator bandwidth should be decreased to match the operating frequency, which would result in a drastic power consumption reduction. For low frequency operation, the comparator power consumption can be reduced to nano-Watt range as shown by Pister, Boser and Scott in a similar comparator implementation [26]. As the frequency of the input signal increases, the fraction of total power consumed by the resistive divider and the comparators becomes much smaller and the energy consumed per each sample decreases (as seen in the table).

At a constant frequency of 5 MHz (and higher), we observed that turning the comparators off did not result in major reduction in power consumption. We found that average power is decreased, but only by a factor of 1.23. Thus, at higher operating frequencies, it is simpler to just leave all the comparators on and eliminate the circuits associated with sleep logic. However, for our application, the comparator power control circuits are important as we expect the ADC to be idle for significant periods of time.

Renaudin came up with a different topology for an asynchronous ADC. He has compared his A-ADC to the conventional synchronous ADCs [2]. The comparison (using his figure of merit) shows that his A-ADC is superior to all the previous implementations. Here we present the comparison of our ADC to Renaudin's ADC, however the concept of SNR is not discussed since were are not reconstructing the signal, and in our case the conventional SNR expressions are not applicable. Since the signal is not reconstructed in our case we don't make a direct comparison between synchronous ADCs and LCF-ADC. The power consumption of an asynchronous level-crossing ADC designed using a successive-approximation topology by Renaudin et al. (A-ADC) depends on the value of the up-down counter (from 0.898 mW to 1.603 mW in the idle mode) [2]. In our case the power consumption is always the same (at constant frequency input signal) and it is much lower than the power of the A-ADC. Our average dynamic power consumption is also significantly lower—the power consumed by the A-ADC is 1.716 mW. The maximum reported input signal bandwidth for the A-ADC is 114 kHz (hardware resolution is 4 bits). Our ADC simulations were performed with varying signal bandwidth (up to 5 MHz) with 4 bits of hardware resolution as well, and we use significantly less power than the A-ADC over the entire range. The A-ADC, however, was designed in a different fabrication technology (0.25 μm); however, a more recent version of the same architecture in 0.12 μm CMOS shows a maximum bandwidth of 160 kHz, and a power consumption of 180 μW, even though their results correspond to a better fabrication technology [1]. An implementation similar to Renaudin's A-ADC was presented by Shepard et al. [14, 15]; the power consumption was in the milli-Watt range – higher then in Renaudin's A-ADC. Lower power consumption in our ADC can be attributed to the architecture of the LCF-ADC, and the representation of the output. Note that associating a timestamp with the output would only require an additional counter, which would not require significant additional power [2] (approximately 15 μW). However, the data presented by Renaudin [1] is a result of measurements, and all the data presented in this paper is a result of simulations and will be confirmed after the LCF-ADC is fabricated.

6 Conclusions and Future Work

In this paper, we have described a new architecture for asynchronous analog-to-digital conversion. The configuration that was presented resembles a flash ADC, but uses completely different digital processing circuits. In contrast with a flash ADC that has all its comparators active, the LCF-ADC proposed here only has a fixed number (2) of active comparators regardless of the hardware precision of the ADC. Our design showed significant reduction in power consumption compared to previously designed level-crossing asynchronous ADCs.

Our LCF-ADC can operate in the MHz regime (optimized for full swing input signals of 5 MHz frequency). It can also work at much higher frequencies with appropriate transistor sizing at the cost of higher power consumption. The architecture also minimizes the transmission bandwidth by outputting only the increments or decrements of the input signal from the previous sample and not the complete sample value (differential encoding). LCF-ADC enables the designers to perform further asynchronous signal processing (triggering, data collection and analysis) on an implicit representation of the input signal as a non-uniform in time sequence of bits. The applications of our LCF-ADC include embedded sensing systems, such as environmental monitoring applications and human body implants.

Future work could examine further reduction in power consumption of the comparators. We are also interested in the design of circuits that operate directly on the implicit signal representation to perform standard signal processing operations.

Acknowledgments. We would like to thank Yannis Tsividis for discussions on analog circuits and theoretical aspects of continuous time analog-to-digital conversion, and Anna Scaglione for discussions on non-uniform sampling theory. This work was supported in part by the NSF under grants ITR 0428427 and NeTS 0435190.

A. Summary of CHP Notation

The CHP notation we use is based on Hoare's CSP [6]. A full description of CHP and its semantics can be found in [21]. What follows is a short and informal description.

- Assignment: $a := b$. This statement means "assign the value of b to a." We also write $a\uparrow$ for $a := true$, and $a\downarrow$ for $a := false$.

- Selection: $[G1 \rightarrow S1 \,[]\, ... \,[]\, Gn \rightarrow Sn]$, where G_i's are boolean expressions (guards) and S_i's are program parts. The execution of this command corresponds to waiting until one of the guards is $true$, and then executing one of the statements with a $true$ guard. The notation $[G]$ is short-hand for $[G \rightarrow skip]$, and denotes waiting for the predicate G to become true. If the guards are not mutually exclusive, we use the vertical bar "|" instead of "[]."

- Repetition: $*[G1 \rightarrow S1 \,[]\, ... \,[]\, Gn \rightarrow Sn]$. The execution of this command corresponds to choosing one of the $true$ guards and executing the corresponding statement, repeating this until all guards evaluate to $false$. The notation $*[S]$ is short-hand for $*[true \rightarrow S]$.

- Send: $X!e$ means send the value of e over channel X.
- Receive: $Y?v$ means receive a value over channel Y and store it in variable v.
- Probe: The boolean expression $\overline{X}$ is $true$ iff a communication over channel X can complete without suspending.
- Sequential Composition: $S; T$
- Parallel Composition: $S \parallel T$ or S, T.

References

[1] E. Allier, J. Goulier, G. Sicard, A. Dezzani, E. Andre, and M. Renaudin. A 120nm low power asynchronous adc. In *Proceedings of ISLPED 2005*, August 2005.

[2] E. Allier, G. Sicard, L. Fesquet, and M. Renaudin. A new class of asynchronous A/D converters based on time quantization. In *Proceedings of the 9th Annual International Symposium on Asynchronous Circuits and Systems*, pages 196–205, May 2003.

[3] F. Beutler. Error-free recovery of signals from irregularly spaced samples. *SIAM Review*, 8(3), 1966.

[4] V. N. Ekanayake, C. Kelly IV, and R. Manohar. Bitsnap: Dynamic significance compression for a low-energy sensor network asynchronous processor. In *Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems*, pages 144–154, March 14-16 2005.

[5] R. Gregorian. *Introduction to CMOS Op-Amps and Comparators*. John Wiley & Sons, 1999.

[6] C. A. R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, 1978.

[7] Yao-Win Hong, Anna Scaglione, and Rajit Manohar. Dense sensor networks are also energy-efficient: When "more" is "less". In *Proceedings of MILCOM 2005*, 2005.

[8] H. Inose, T. Aoki, and K. Watanabe. Asynchronous delta modulation system. *Electronics Letters*, 2:95–96, 1966.

[9] P.G.A. Jespers. *Integrated Converters, D to A and A to D Architectures, Analysis and Simulation*. Oxford University Press, 2001.

[10] B. Kedem. Spectral analysis and discrimination by zero-crossings. *Proceedings of the IEEE*, 74(11):1477–1493, November 1986.

[11] B. Kedem. *Time Series Analysis by Higher Order Crossings*. IEEE Press, 1994.

[12] D. Kinnement and A. Yakovlev. Low power, low noise micropipelined flash A-D converter. *IEEE Proceedings on Circuits Devices Systems*, 146(5), 1999.

[13] D. Kinnement, A. Yakovlev, and B. Gao. Synchronous and asynchronous A/D conversion. *IEEE Transactions on VLSI Systems*, 9(2), 2000.

[14] Y. W. Li, K. L. Shepard, and Y. P. Tsividis. Continuous-time digital signal processors. In *Proceedings of the 11th Annual International Symposium on Asynchronous Circuits and Systems*, New York City, USA, March 14-16 2005.

[15] Y. W. Li, K. L. Shepard, and Y. P. Tsividis. A continuous-time programmable digital fir filter. *To be published in Journal of Solid-State Circuits*, 2005.

[16] B.F. Logan. Information in the zero crossings of bandpass signals. *Bell Systems Technical Journal*, 56(4):487–510, April 1977.

[17] Rajit Manohar. Asynchronous VLSI systems. Class Notes for ECE 574 at Cornell University, 1999.

[18] J. Mark and T. Todd. A nonuniform sampling approach to data compression. *IEEE Transactions on Communications*, COM-29(4), 1981.

[19] Alain J. Martin. Compiling communicating processes into delay-insensitive VLSI circuits. *Distributed Computing*, 1(4), 1986.

[20] Alain J. Martin. The limitations to delay-insensitivity in asynchronous circuits. In William J. Dally, editor, *Proceedings of the 6th Conference on Advanced Research in VLSI*, pages 263–278. MIT Press, 1990.

[21] Alain J. Martin. Programming in VLSI: From communicating processes to delay-insensitive circuits. In C. A. R. Hoare, editor, *Developments in Concurrency and Communication, UT Year of Programming Series*, pages 1–64. Addison-Wesley, 1990.

[22] A. Oppenheim, A. Wilsky, and I. Young. *Signals and Systems*. Prentice-Hall, 1995.

[23] S.O. Rice. Statistical properties of a sine wave plus random noise. *Bell Systems Technical Journal*, 27:109–157, January 1948.

[24] J. Sauerbrey, D. Schmitt-Landsiedel, and R. Thewes. A 0.5V, 1W successive approximation ADC. In *ESSCIRC proceedings*, Firenze, Italy, September 24-26 2002.

[25] N. Sayiner, H. Sorensen, and T. Viswanathan. A level-crossing sampling scheme for A/D conversion. *IEEE Transactions on Circuits and Systems II*, 43(4), 1996.

[26] M. Scott, B. Boser, and K. Pister. An ultra low-power ADC for distributed sensor networks. In *ESSCIRC proceedings*, Firenze, Italy, September 24-26 2002.

[27] Y. P. Tsividis. Digital signal processing in continuous time: a possibility for avoiding aliasing and reducing quantization error. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, May 17-21 2004.