Continuous Sampling and Storage Using Matched Delays

Mark Clements, Wentai Liu, Jin-Ku Kang, Ralph Cavin III

Department of Electrical and Computer Engineering
North Carolina State University
Raleigh, NC 27695–7911

December 28, 1993

Abstract

This paper describes a scheme for continuous sampling and storage of digital data using the matched delay sampler. This sampler, which can sample data at very high time resolutions, has been previously reported. We will describe the additions to the sampler that are required for data to be continuously sampled and stored. The frequency of the sample clock necessary for continuous sampling will be determined. The steps needed to deskew and synchronize the latch outputs and store them in an acquisition memory are detailed. This article provides an architectural description that can be applied to designs based on any circuit family. An example architecture is described.

1 Introduction

High resolution digital sampling capability is extremely important for a number of applications in the high speed communications and test and measurement areas. Examples include clock and data recovery circuits, digital phase locked loops, and high speed logic timing analyzers. However, high resolution sampling typically has required very fast latches and high speed clocks. As a result, the sampling of very high speed signals has been expensive, if possible at all. The matched delay sampler provides a way to avoid this difficulty.

The operating principle of the matched delay sampler is that of a tapped delay line where *both* the data and clock signals are delayed at each stage. This structure provides a timing resolution between stages that is equal to the difference between the data delay and clock delay. Its primary benefit is that a very high effective sampling frequency is possible without requiring latches and clocks that operate at the sampling frequency. The structure has been used to make precise time measurements between two pulses over a limited range [1]. Another group has used it to build a circuit capable of high resolution sampling of digital data with multiple edges for a short time window [2].

The matched delay sampler can acquire digital data with very high timing resolution, but the work that has been done to date addresses only the case of a single sampling clock pulse. In this case, only a very short time window of continuous data can be acquired. It is possible, however, to drive the sampler with a periodic sample clock of the appropriate frequency so that continuous data can be acquired indefinitely. An acquisition memory is needed to store the data, and its size is the limiting factor on the duration of the time window. Some additional circuitry is needed between the sampler and the memory to synchronize the sampler outputs.

Here, we will develop an architecture, built around the sampler, that can acquire a much larger number of samples at the same high resolution. This system is driven by a periodic clock to sample continuous high resolution data and store it in an acquisition memory. The system retains the advantages of the sampler in that the system clock is much slower than the sample rate, and so components that operate at the sample rate are not needed. After reviewing the operation of the matched delay sampler itself, we will examine the additional problems that must be solved to achieve continuous sampling. Specific design examples will be described.

2 The Matched Delay Sampler

Figure 1 shows a block diagram of the basic matched delay sampler [2, 3]. The following quantities will be used to describe an architectural model of a sampler of arbitrary size and sample period:

N – the number of stages in the sampler

D_d – the data delay time

D_c – the clock delay time

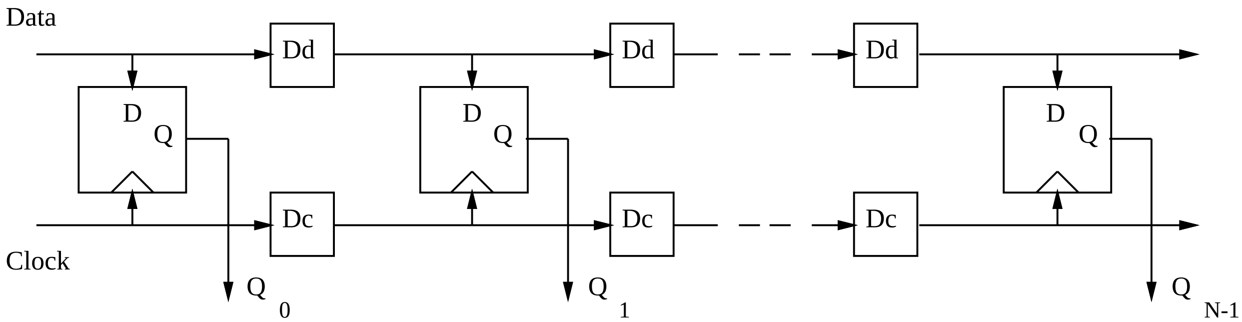

Figure 1: Matched Delay Sampler

Δt – the sampling interval

The sampler has N stages, and each stage consists of a latch, a data delay, and a clock delay. Data and clock are simultaneously propagated through delays D_d and D_c respectively. The latches sample the local data at each stage with the local clock. Adjacent latches ($i-1$ and i) sample data at a time interval of Δt , which is the sampling resolution. The definition of Δt in terms of D_c and D_d as can be derived as follows. If $D_i(t)$ is the value of the data at stage i at time t and Q_i is the output of latch i, then

$$Q_{i-1} = D_{i-1}(t_0)$$

where t_0 is the particular time when the clock reaches latch $i-1$. The clock will reach latch i at time $t_0 + D_c$, and so

$$Q_i = D_i(t_0 + D_c)$$

Since adjacent latches sample data that is Δt apart in time, we can define Q_i in terms of D_{i-1}:

$$Q_i = D_{i-1}(t_0 + \Delta t)$$

The data that is at latch i at time $t_0 + D_c$ was at latch $i-1$ at time $t_0 + D_c - D_d$, so

$$Q_i = D_{i-1}(t_0 + D_c - D_d)$$

The sample interval is thus equal to the absolute difference between the data and clock delays:

$$\Delta t = |D_c - D_d|$$

The absolute value of the difference is taken since either the clock delay or data delay may be larger. Note that the above derivation assumes $D_c > D_d$. If D_c is larger, then a clock edge latches data samples that are increasingly later in time as it propagates through the sampler. If D_d is larger, then the clock passes through the sampler faster than the data, and so successive samples are earlier in time.

This matched delay scheme has an important advantage over conventional sampling schemes. It allows a very high sampling frequency to be obtained without requiring the elements of the sampling system to run at that frequency. Also, while schemes in which either the data or clock are delayed have the same advantage, their resolution is limited by the minimum possible delay of the delay elements. The matched delay sampler can have a sampling interval that is much smaller than the smallest possible absolute delay. In fact, the difference between the clock and data delays can be made arbitrarily small, and so the resolution is effectively limited by noise and process variation. The matched delay sampling scheme has been implemented as a VLSI device in a 1.2 μm CMOS process. The device consists of 64 stages, and is able to achieve a 25 ps sampling resolution at a 1 Gbps data rate. This type of sampler serves as the basis for the work to be described here.

A single clock pulse sent into the clock delay chain can acquire continuous data only for N samples, or a time window of $t = N\Delta t$. If a larger duration is required, then the sampler must be repetitively clocked, and the data from each clock stored. In the following section, we will determine the correct period for the sampling clock.

3 Continuous Clocking

It can be seen that if a sample clock reaches the first latch at time $t = 0$, its final sample will be the data at time $t = (N-1)\Delta t$, and so the next clock must be sent at $t = N\Delta t$. Therefore, the sample clock period, T, for a sampler with any values for D_c and D_d is defined as

$$T = N\Delta t$$

The same conclusion can be reached in a slightly different way that considers the operation of the sampler in more detail. If at $t = 0$, both the first clock edge, $C0$, and a particular data point, $D0$, are at the first stage of the sampler, $D0$ will "fall off" the far end of the sampler at $t = ND_d$, and $C0$ at $t = ND_c$. The final sample taken by $C0$ will be latched at $t = (N-1)D_c$. The data that would have been latched by $C0$ at $t = ND_c$, if there were one additional stage at the end of the sampler, must be latched by the next clock, $C1$, *when* that same data is at the beginning of the sampler. Therefore, the sample clock period T is the difference between the time required for a clock to traverse the sampler and the time for a data point to traverse the sampler:

$$T = |ND_c - ND_d| = N|D_c - D_d| = N\Delta t$$

Since either D_c or D_d may be larger, the absolute value of the difference is used. The above discussion assumes $D_c > D_d$. If D_d is larger, then the samples represent earlier data as a clock propagates through the sampler. The description above would be modified to say that the first sample missed when a clock reaches the end of the sampler is latched by the *previous*, rather than the next clock.

Note that T determines not only the period of the sampling clock but the frequency of operation of the entire continuous sampling system. The synchronization hardware and the storage memory cycle with period T. These other elements of the system, and how they are related to the sampling clock will be described in later sections.

Figure 2 illustrates the operation of the sampler with repetitive clocking. It is based on a particular set of sampler parameters that we will use as an example in the remainder of this article. The parameters are:

$N = 16$

$\Delta t = 1$

$D_d = 3$

$D_c = 4$

$T = 16$

The figure consists of a series of snapshots that show the locations within the sampler of data points and clocks edges at the particular points in time that are indicated to the left. The upper arrows represent data samples propagating through the data delay chain, and the lower arrows represent sampling clock edges. The numbers in parentheses next to the clocks are the times of the data samples latched by that edge at that time.

There are several important observations to be made from figure 2. They are indications of the other problems that must be solved to achieve continuous sampling. First, there are several sample clocks propagating through the sampler at once. The clocks divide the sampler into *sections*. The example sampler has four sections, with four channels in each section. Second, the valid times of the channels within each section are skewed relative to each other. Third, the samples present in the sections at a given time are not consecutive. The samples must be synchronized before they can be sent to the memory. The following section describes how this synchronization is done.

4 Synchronizing the Latch Outputs

Ideally, we would like save N time continuous data bits into a register and send them to the acquisition memory every T seconds. However, the sampler as described to this point does not present all N continuous bits on the latch outputs simultaneously. The data valid times of the latch outputs are not synchronous because of the nature of the clocking scheme. There are two problems that must be solved. First, the delay of the clock at each sampler stage skews the valid times of adjacent latches by D_c . Second, there are several clocks propagating through the sampler at once, and consecutive clocks do not produce consecutive samples.

4.1 Clock Delay Skew

Because the clock signal is delayed before each latch, the data valid times of adjacent latches are skewed by D_c (figure 3). This skew makes it difficult to store all the samples simultaneously into a register so that they can be sent to the memory. By delaying the earlier

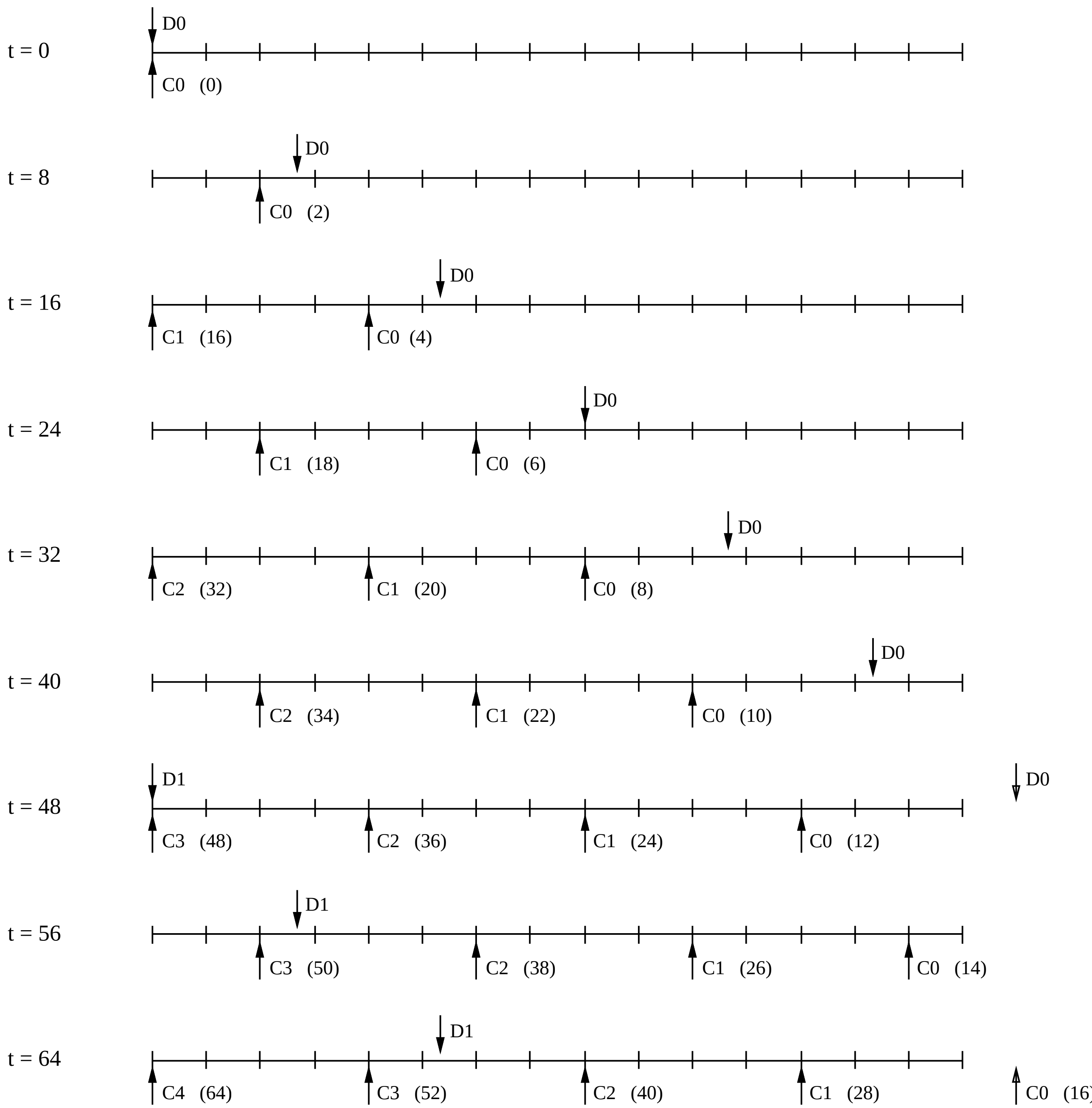

Figure 2: Continuous Clocking of the Sampler

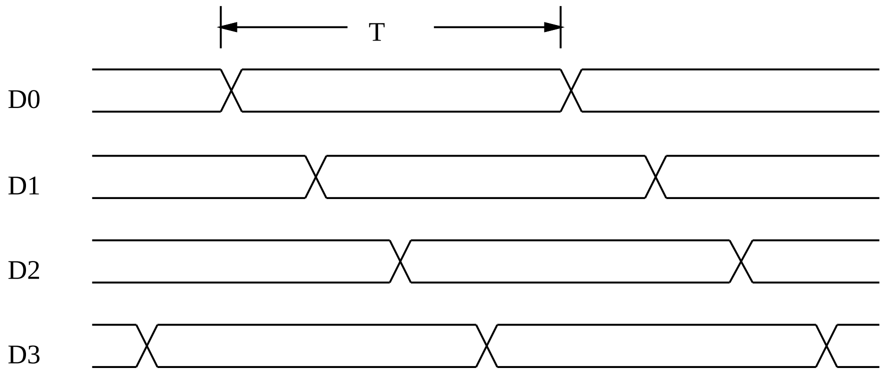

Figure 3: Latch Output Skew

latch outputs by the appropriate amount, we can re-align the samples for storage. Precise deskewing can be achieved by delaying each output by the correct number of delay units with the same delay value as the clock delays. In reality, however, the samples need only be deskewed to the extent necessary to successfully latch them into the register. If all the outputs meet the setup and hold criteria of the register, then the deskewing is successful, and all N samples can be stored at once.

There are two basic deskewing schemes, and each has certain advantages. The first is based on adding delays to the latch outputs within a section to realign them. The second approach relies on latching, and thus delaying some of the earlier channels to align them with the later ones.

4.2 Delay-Based Deskewing Scheme

The delay-based scheme allows more precise deskewing of the latch outputs within a sampler section. In fact, the channels can be completely deskewed, so that the maximum possible setup / hold window for the synchronizing FIFO's can be provided. This exact deskewing creates a window equal to the sample clock period less rise and fall times. A maximum setup / hold window allows for the greatest immunity to process variation and jitter of the FIFO and memory clocks. However, exact deskewing with delays requires a relatively large chip area to implement because of the number of delay elements that are needed. If only clock delay units are used, then the number of delays required is:

$$n_d = \tfrac{1}{2} n_s(n_s - 1) * n_c$$

where

$n_s = T/D_c$ is the number of sampler stages per section.

and

$n_c = D_c \,/\Delta t \ = N/n_s$ is the number of sections.

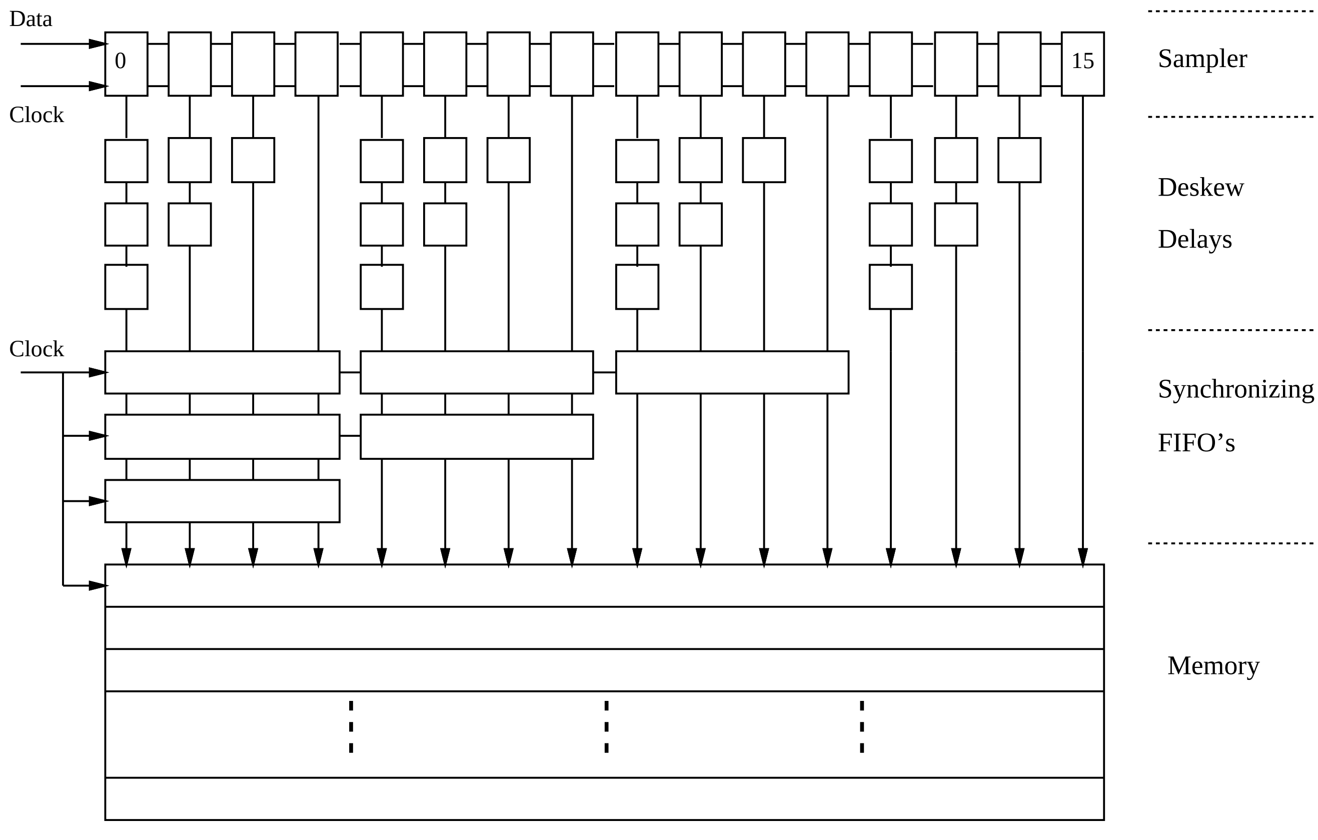

Figure 4: Continuous Sampling Architecture

Both n_s and n_c are required to be integers. For our example sampler, $n_s = 4$ and $n_c = 4$, and so $n_d = 24$ delay units. The configuration of the delays is shown in figure 4. This number of delays is not large, but the example sampler is a simple case. For a more realistic sampler, the number of stages per section is likely to be larger, and so the number of delays required will be much larger. For one 64-stage sampler that has been studied, $n_s = 16$ and $n_c = 4$, which yields $n_d = 480$.

Note that the above examples represent the worst case. Fortunately, precise deskewing is not necessary. The real requirement is that the latch outputs be deskewed enough so that they can be successfully latched by the synchronizing FIFO's. The required deskewing can likely be achieved using significantly less real estate than than would be needed for the worst case. There are several steps that can be taken to reduce the required area. First, only the number of delays needed to provide just enough deskewing can be used. Second, using a smaller number of coarser delays elements can reduce the area further. Finally, an element of a given delay value can potentially be significantly smaller than the clock delay, because it does not have to be as accurate or be adjustable. For example, in a given CMOS process, a minimum size static inverter should have a larger delay than the clock delay element, yet it is much smaller in area.

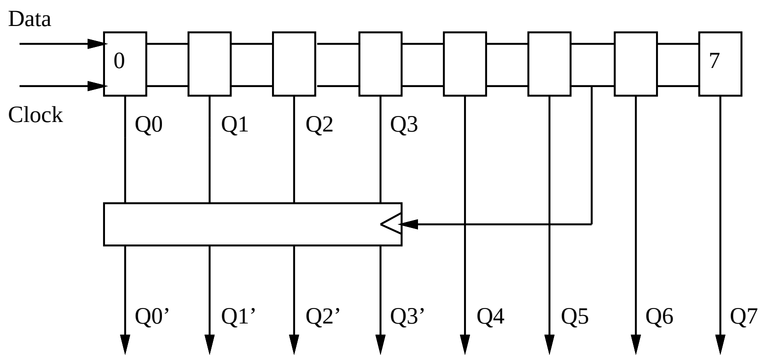

Figure 5: Half-section Register Deskewing

4.3 Half-Section Register Deskewing Scheme

The half-section register deskewing scheme saves significant real estate compared to the delay scheme, at the cost of less complete deskewing. It can provide a setup / hold window that is slightly larger than $T/2$. This method uses latches rather than delay elements. Figure 5 illustrates the half-section scheme for one eight-channel section. The channels in the upstream half of each section of the sampler are latched by a version of the sample clock that has its phase matched to the downstream half of the section. The upstream channels are thus delayed by about half a sample clock period, and so their skew relative to the downstream channels is mostly cancelled. If the phase of this deskew clock is correct relative to the synchronization FIFO clock, then the setup / hold window for the sync FIFO's will be determined by the skew among the downstream channels of the section. Neglecting rise and fall times, this window will be:

$$t_{s/h} = T/2 + D_c$$

Note also that the setup / hold window for the deskewing latches is the same.

The clock for the deskew latches can be derived simply by taking the delayed version of the sample clock from the appropriate stage of the sampler. The correct stage to provide the deskew clock should be somewhere near the middle of the downstream half of the section, or about three quarters of the way down the section. Figure 6 shows the results of the half-section scheme for the eight-channel section.

4.4 Hybrid Deskewing Scheme

If more precise deskewing than can be provided by the half-section latch approach is required, a hybrid of both techniques should be able to completely deskew the channels while still saving significant area compared to the delay technique. Adding just a few delay elements to the half-section architecture could reduce the skew among the downstream channels and thus increase the setup / hold window for the FIFO's.

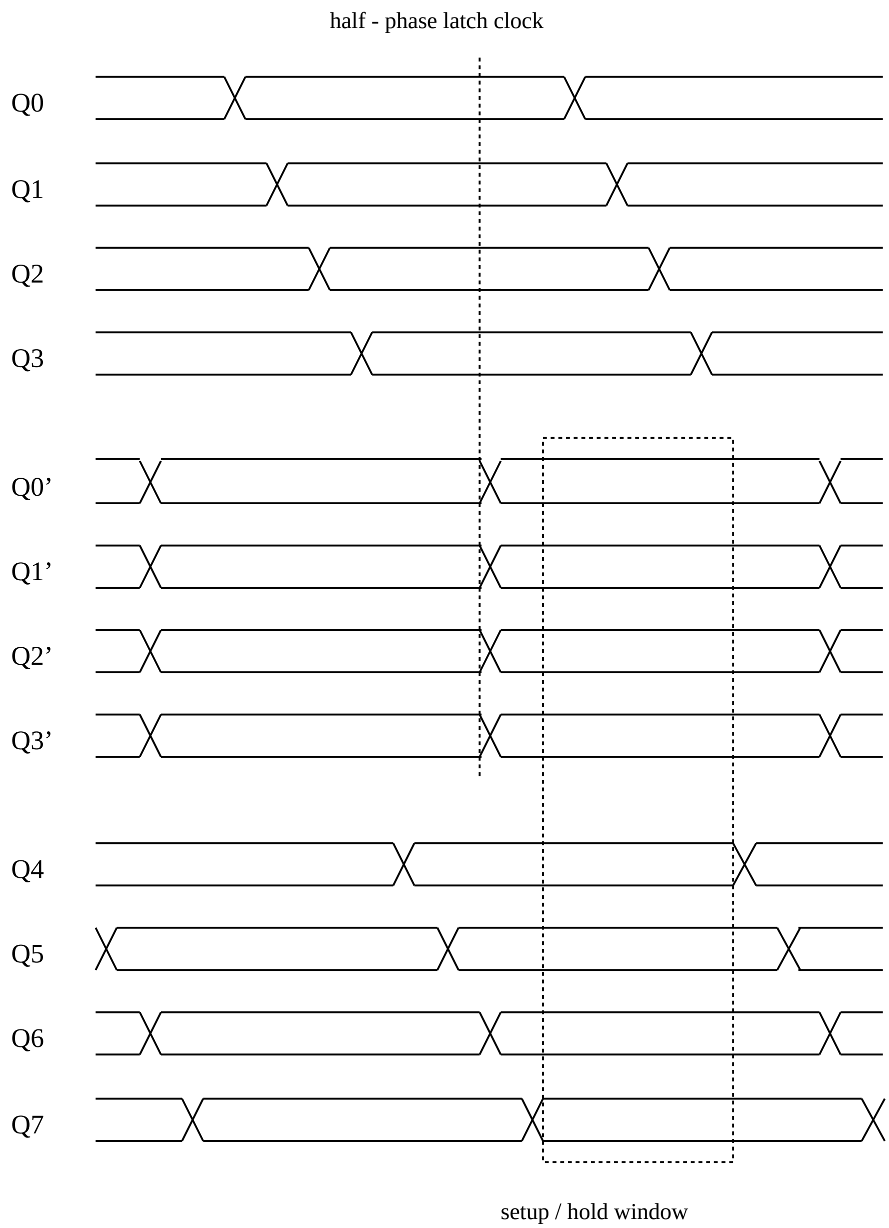

Figure 6: Half-section Register Deskewed Outputs

4.5 Non-Consecutive Sections

For the matched delay sampler, Δt is significantly smaller than D_c , and so there are several sample clock cycles propagating through the sampler at once. Each clock travels only part way through the sampler during each memory cycle, and the section of data latched by one clock is not consecutive with the other sections that are stored on the same memory cycle. In the example sampler, there are four clock cycles present at once, and so each clock participates in four memory cycles. The times represented by the sixteen samples stored on each of three sequential memory cycles would be:

$(t = 48)$	48 49 50 51	36 37 38 39	24 25 26 27	12 13 14 15
$(t = 64)$	64 65 66 67	52 53 54 55	40 41 42 43	28 29 30 31
$(t = 80)$	80 81 82 83	68 69 70 71	56 57 58 59	44 45 46 47

The samples can be re-synchronized by delaying the sections with clocked FIFO's of the appropriate number of stages. The FIFO's are clocked at the same rate as the sampler and memory, a period of T. Some circuit-dependent phase adjustment of the FIFO clock relative to the sampler clock would be required for correct operation.

In our example sampler (figure 4), the first, second, third, and fourth sections are delayed by three, two, one, and zero clocks, respectively. The synchronizing FIFO's allow the sampler to present N consecutive samples to the memory every T seconds.

4.6 Additional Synchronization Considerations

There is one additional practical consideration related to the synchronization of the sampler sections. If D_c is not an integral multiple of Δt , then the clocks in the sampler will be out of phase, and so the synchronization FIFO for each section will have to be clocked by a separate clock. The complexity of the system will be increased significantly.

For some applications the non-consecutive nature of the section outputs might not be a problem, and so the synchronization registers would not be needed. The system software could synchronize the data after the acquisition is completed.

5 The Memory System

The memory system stores N-bit words that represent the samples spanning a single time window of T seconds. Therefore, since the samples are consecutive, it must have a cycle time of T seconds. The size of the memory limits the duration of the overall time window of data that can be stored by the sampling system. For a time window of T_w seconds, the number of memory bits, M_b, needed is

$$M_b = T_w/\Delta t$$

and the number of N-bit memory words, M_w, required is

$M_w = T_w / N\Delta t$

Actually, a fully functional random access memory is not needed. An N-bit wide, clocked FIFO of depth M_w will work, because the data stream is purely sequential. A mechanism for clocking the acquired data out of the other end of the FIFO after an acquisition is completed is needed. This mechanism should operate with a clock that is completely separate from the sampling clock.

6 Conclusions

We have described a new method of sampling very high speed digital data for long time durations. The method has the potential to increase the highest possible sample rates and reduce the costs of sampling at a particular rate because it has the ability to sample data at frequencies that significantly exceed the maximum switching rate of the circuits and minimum cycle time of the memory. It is based on the previously reported matched delay sampler [2, 3]. We have shown the architectural requirements for a system that can use the matched delay sampler to continuously acquire and store data. The frequency of the sampling clock required for continuous sampling with the matched delay sampler was derived. The details of the additional circuitry necessary to deskew and synchronize the sampler outputs, and the requirements for the memory system, were presented. The scheme can be applied to many possible circuit-level designs.

References

[1] T. Rahkonen, J. Kostamovaara, and S. Saynajakangas. Time interval measurement using integrated tapped CMOS delay lines. In *32nd Midwest Symposium on Circuits and Systems*, pages 201–205, 1990.

[2] W. van Noije, C. T. Gray, W. Liu, T. A. Hughes, R. K. Cavin, and W. J. Farlow. CMOS sampler with 1 gbit/s bandwidth and 25 ps resolution. In *Custom Integrated Circuits Conference*, pages 27.5.1–27.5.4, San Diego, CA, May 1993.

[3] T. A. Hughes, C. T. Gray, W. Liu, and R. K. Cavin. Method and apparatus for high speed digital sampling of a data signal, March 25 1992. U.S. Patent Pending 07/857,457.