Instantaneous Clockless Data Recovery and Demultiplexing

Behnam Analui and Ali Hajimiri

***Abstract*—An alternative architecture for instantaneous data recovery for burst-mode communication is introduced. The architecture can perform 1:n demultiplexing without additional clock recovery phase-locked loop or sampling blocks. A finite-state machine (FSM) is formed with combinational logic and analog *LC* transmission line delay cells in a feedback loop. The FSM responds to input data transitions instantaneously and sets the outputs. The system reduces unit interval jitter by a factor of n. The new architecture is demonstrated via a SiGe 1:2 clockless demultiplexer circuit that operates at 7.5 Gb/s.**

***Index Terms*—Burst-mode communication, combinational logic circuits, delay circuits, delay lines, demultiplexing, emitter coupled logic, finite-state machines (FSMs), passive circuits, transmission lines.**

I. Introduction

BURST-MODE communication relies on very fast acquisition circuitry to achieve low network latency. If the data stream is bursty, the receiver must be able to synchronize with the data instantaneously to maintain reliable communication. For Burst-mode communication, conventional clock recovery (CR) methods based on narrow-band phase-locked loops (PLLs), such as the ones designed for SONET applications, are not applicable. PLL-based CR circuits in SONET have stringent jitter transfer specifications to avoid jitter accumulation. In addition, they are required to tolerate long sequences of identical bits [1]. These constraints impose a narrow-band PLL that will have long acquisition time. For instance, [1] reports minimum of 50-μs acquisition time for a 155-MHz CR circuit. In this case, approximately the first 8000 bits of data will be lost. Although designing a wide-band PLL reduces the number of lost bits, it still requires preamble bits and coding schemes to alleviate this problem.

Gated oscillator clock recovery provides instantaneous lock to the first data transition. Such circuits have been reported at several hundred megabits per second [2]–[4]. Gated oscillator clock recovery relies on two oscillators that are activated with rising and falling edges of the input signal and are, hence, resynchronized with every transition. The frequency of the gated oscillators are equal to the bit rate and the right value is typically maintained via another replica oscillator in a PLL.

We introduce an alternative method for instantaneous data recovery and show a prototype that works at 7.5 Gb/s. A finite-state machine (FSM) combines data recovery and demultiplexer functions. It receives the data and decides on the output values based on the current input data and the previous state. The previous state is provided to the FSM input with a bit period delay. While decisions are synchronized with every incoming data transition, no oscillator is required. Although the jitter transfer function is flat similar to gated oscillator-based approach, there is a reduction by a factor of n, the demultiplexing ratio, in output jitter due to the integrated demultiplexer function.

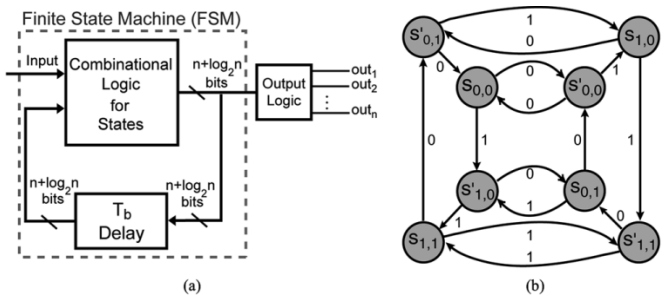

Fig. 1. (a) Clockless 1:n demultiplexer. (b) State diagram of the 1:2 demultiplexer.

We first introduce the new general architecture for 1:n clockless demultiplexer and discuss the complexity of the FSM for different demultiplexing ratios, n. Then, we describe the design procedure for a 1:2 demultiplexer and discuss different possibilities for implementing the delay cells. Lastly, we demonstrate the experimental results of the fabricated prototype.

II. Clockless 1:n Demultiplexer

Fig. 1(a) shows the general block diagram of the proposed clockless data recovery and the 1:n demultiplexer. The FSM consists of a combinational states logic block and bit period delay cells. It maps the combination of the input and previous state to a current state. Previous state is the output of the FSM at the last bit period, T_b, and is fed back to the input of the states logic block with a delay of T_b. The output logic block generates the demultiplexed outputs based on the current state. Both logic blocks respond to their inputs instantaneously. Therefore, each data transition at the input immediately affects the outputs of the demultiplexer, if logic gate propagation delay is neglected.

In a conventional 1:n demultiplexer, each output changes based on its corresponding bit at the input and then keeps its value for nT_b periods. The combination of the FSM and output logic operates similarly. Each input bit is directed to the proper output and the values of the other outputs are kept constant in the memory of FSM state. Therefore, the information stored in

Manuscript received June 4, 2004. This paper was recommended by Associate Editor A. G. Andreou.

The authors are with the California Institute of Technology, Pasadena, CA 91125 USA (e-mail: behnam@caltech.edu; hajimiri@caltech.edu).

Digital Object Identifier 10.1109/TCSII.2005.850453

TABLE I
STATE-OUTPUT CORRESPONDENCE FOR 1: 2 DEMULTIPLEXER

Current State	out_1	out_2	Current State	out_1	out_2
$S_{0,0}$	0	0	$S_{0,1}$	0	1
$S'_{0,0}$	0	0	$S'_{0,1}$	1	0
$S_{1,0}$	1	0	$S_{1,1}$	1	1
$S'_{1,0}$	0	1	$S'_{1,1}$	1	1

TABLE II
ASSOCIATED CODE WORD TO EACH STATE

State	Code($y_0y_1y_2$)	State	Code($y_0y_1y_2$)
$S_{0,0}$	001	$S_{0,1}$	100
$S'_{0,0}$	000	$S'_{0,1}$	011
$S_{1,0}$	010	$S_{1,1}$	111
$S'_{1,0}$	101	$S'_{1,1}$	110

one state of the FSM consists of the value of current bit (1 bit), the values of the unaffected bits ($n-1$ bits), and the binary address of the affected output ($\log_2 n$ bits). As a result, for a $1{:}n$ demultiplexer a $n+\log_2 n$ bit FSM is required, as Fig. 1(a) demonstrates. Consequently, the number of states in the FSM is $n \cdot 2^n$.

The delay cell in Fig. 1(a) guarantees that the information of the previous state is available whenever a data transition occurs, i.e., every bit period. The output is updated for every data transition and thus there is no explicit jitter rejection. However, the input data jitter at any transition impacts only one of the n outputs. Moreover, each output has $1/n$ the data rate of the input. Therefore, effective output data jitter in unit intervals (UI), i.e., normalized to nT_b, is reduced to $1/n$ of the input data jitter.

Next, we demonstrate the design of a 1:2 demultiplexer based on the clockless data recovery method.

III. DESIGN OF 1:2 DEMULTIPLEXER

The major advantage of the clockless demultiplexer is that it responds to data transitions instantaneously. After a long quiet period with zero input and zero outputs, the first data transition initiates the state transitions of the FSM. Thereafter, the FSM state is updated every period, T_b, synchronous to data transitions. If the delay is not exactly T_b, any incoming data transition resynchronizes the FSM. The output of FSM will be valid as soon as the data transition arrives at the input provided that the logic propagation delay is negligible.

Fig. 1(b) illustrates the states and state transitions of the 1:2 clockless demultiplexer. Each arrow corresponds to a state transition in FSM. The binary value on the arrow represents the current value of the input. For the 1:2 demultiplexer, the FSM has a total of eight states, as shown in Fig. 1(b). The FSM stays in each state for a period of T_b. Then it transitions to the next state based on the input bit. The prime superscript in the state name is equivalent to the *select* line of a conventional demultiplexer. States with prime superscript correspond to the ones for which input bit affects out_2. The first subscript in the state name is the current input bit and the second subscript is the previous input bit stored to hold the unaffected output. For example, when FSM is in $s_{1,0}$ it corresponds to the state when the input bit, "1," is transferred to out_1 and stored previous bit, "0," is transferred to out_2. If after T_b, a data transition occurs and input bit is "0," FSM transitions to $s'_{0,1}$, for which input bit, "0," is now transferred to out_2 and stored previous bit, "1," is transferred to out_1. Table I summarizes the two output values for all the eight states.

A 3-bit FSM represents the state diagram in Fig. 1(b). Each state is assigned a 3-bit code word. To avoid erroneous transitions to other states, i.e., races, codes are assigned such that only one bit changes in every state transition. Therefore, delay mismatches in the implementation cannot cause errors and the FSM is race free. The code words are presented in Table II.

We associate the binary variables y_0, y_1, and y_2 to the three bits that code the FSM states. Therefore, when y_0, y_1, and y_2 are updated every T_b, we say FSM has transitioned to the next state. The updated binary values for each of y_0, y_1, or y_2 is determined from the state diagram and the code word table based on the current values of y_0, y_1, or y_2 and the input bit. The next value of each of the three binary variables is described as

$$y_i^* = f_i(y_0, y_1, y_2, x), \qquad i = 0, 1, 2 \tag{1}$$

where "$*$" signifies the updated value of y_i. Function f_i is a logic function and the arguments are the current values of the binary variables. Variable x corresponds to the current input bit. The logic functions are designed based on standard methods such as sum of products (SOP) using Karnaugh maps [5].

Similarly, out_1 and out_2 can be represented as functions of y_0, y_1, y_2, and x. Based on the concept of the 1:2 demultiplexer we predict the logic function for the two outputs as

$$\mathrm{out}_1^* = \mathrm{out}_1 \cdot S + x \cdot \bar{S} \tag{2}$$

$$\mathrm{out}_2^* = \mathrm{out}_2 \cdot \bar{S} + x \cdot S \tag{3}$$

where S is a binary function representing which output should change and is equivalent to the select line of an ordinary demultiplexer. The "$\cdot$" and "$+$" are logical *AND* and *OR* operators, respectively. For instance, in (2), the next value for out_1 is the current value of out_1 if $S = 1$, and is the input if $S = 0$. The change is synchronous with x.

From Fig. 1(b) and Table II we can show that

$$S = y_0 \cdot (y_1 \oplus y_2) + \overline{y_0} \cdot \overline{(y_1 \oplus y_2)} \tag{4}$$

where $\oplus$ is the "*exclusive or*" operator. Additionally, we can show $\mathrm{out}_1 = y_1$ and $\mathrm{out}_2 = y_0$. Therefore, the output logic block in Fig. 1(a) is omitted and the outputs are tapped directly

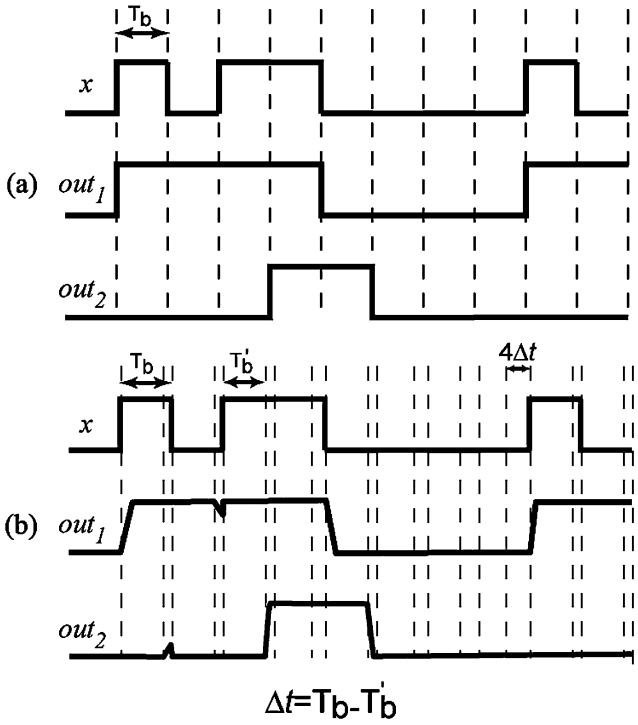

Fig. 2. Demultiplexer outputs for 1 011 000 010 input sequence (a) Ideal case. (b) Delay cell has smaller delay than bit period.

from the FSM output variables. Hence, the simplified output functions are

$$\mathrm{out}_1 = y_0 \cdot (y_1 \oplus y_2) + x \cdot (y_0 + y_1 \oplus y_2) \quad (5)$$

$$\mathrm{out}_2 = y_1 \cdot \overline{(y_1 \oplus y_2)} + x \cdot (y_1 + \overline{y_0 \oplus y_2}). \quad (6)$$

Equations (5) and (6) are computed directly from digital maps of Fig. 1(b) using Table II code words. However, they have the form as in (2) and (3) and can be obtained directly by replacing (4) in (2) or (3).

IV. Delay Cell

The delay cell in Fig. 1(a) is the synchronizing block in the demultiplexer that controls how long the FSM stays in one state. When the first data transition initiates the demultiplexer, FSM binary outputs, i.e., y_0, y_1, y_2 in a 1:2 demultiplexer, are fed back to the input of FSM every T_b and generate next state and next outputs based on (1), (5), and (6). Ideally, the delay must be equal to the input bit period T_b. Fig. 2(a) demonstrates how the demultiplexer operates for a sample input data. When one output is following the input, the other output is holding its own previous value.

If the delay cell has a delay T_b', different from T_b, the outputs might experience glitches. However, any input transition will immediately correct those glitches and avoid any unwanted output transition. Fig. 2(b) illustrates an example where $T_b' < T_b$. As can be seen, out_1 in the second bit period is holding its previous value, "1". After T_b' new values for y_0, y_1, and y_2 are ready at the FSM input while the correct input, x, corresponding to out_1 has not arrived yet. out_1 starts to follow the incorrect x. However, after $\Delta t = T_b - T_b'$, the next data transition arrives and $\mathrm{out}_1, \mathrm{out}_2, y_0, y_1$, and y_2 are immediately corrected

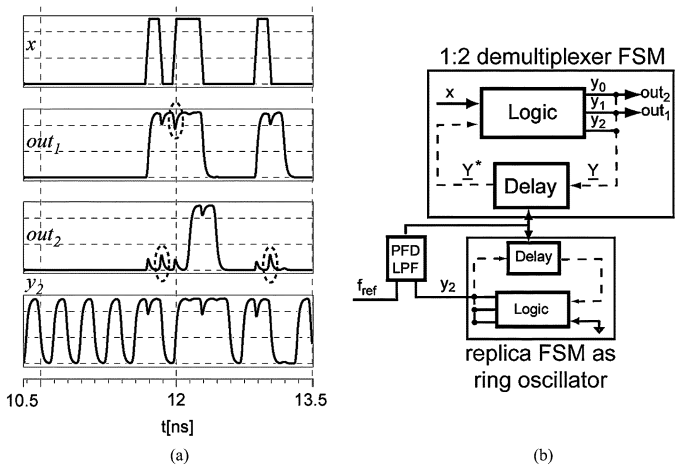

Fig. 3. (a) Outputs of the 1: 2 Demultiplexer when $T_b' < T_b$ simulated with HSPICE (b) Demultiplexer with delay control loop.

because they relate to x with a combinational logic relation such as (5) or (6). Although glitches are observed at the outputs as a consequence, delay mismatch is corrected every cycle and does not accumulate when data transitions occur.

A behavioral simulation of the demultiplexer in HSPICE confirms the above argument. The T_b' is 125 ps while input data is 7.5 Gb/s. In Fig. 3(a), the marked bumps on out_1 and out_2 correspond to the same glitches discussed in Fig. 2(b). As predicted, the arriving data transition immediately corrects for such errors. The other weaker bumps in the outputs are related to hazards that occur when two or more terms in the SOP of output function are changing simultaneously, while the overall output function remains at the same logic level.

The delay mismatch bounds the maximum number of consecutive identical bits (CIB) at the input. If delay mismatch is Δt, as in Fig. 2(b), number of CIBs should be less then $n = T_b/\Delta t$. Otherwise, the FSM will swap the outputs and the $(n+1)$ th bit will be resolved at the incorrect output.

Δt can be made very small by using a delay control circuit that forces Δt to zero. A ring oscillator is formed by closing a positive feedback loop around a replica of the delay cells. The period of oscillation equals twice the delay of the delay cells, $2T_b'$. A PLL locks the frequency of the ring oscillator to an accurate reference clock by tuning the replica delay cell. The same control voltage is used to adjust the delays in the FSM. In practice, the logic circuits in the FSM have propagation delays that contribute to the total delay around the feedback loop of the FSM. Furthermore, process variations could significantly impact the propagation delay of the gates. Therefore, a delay control loop that adjusts the delay cells alone would be inconsequential. The replica ring oscillator must include all the blocks that contribute to the delay. For the 1:2 demultiplexer, it can be shown that in the absence of input transitions, f_2 from (1) is simplified to

$$y_2^* = \overline{y_2}. \quad (7)$$

Equation (7) shows the y_2 output inverts for every period delay around the FSM loop, T_b'. In other words, the y_2 output oscillates with the period of $2T_b'$ when the input is constant. In fact, y_2 acts as the internal timer of the FSM in the absence of input

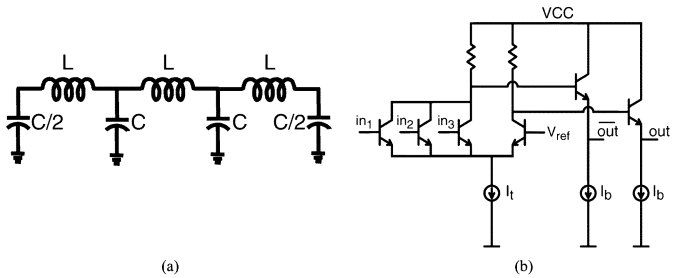

Fig. 4. (a) Three-section *LC* ladder delay cell. (b) Three-input ECL OR gate.

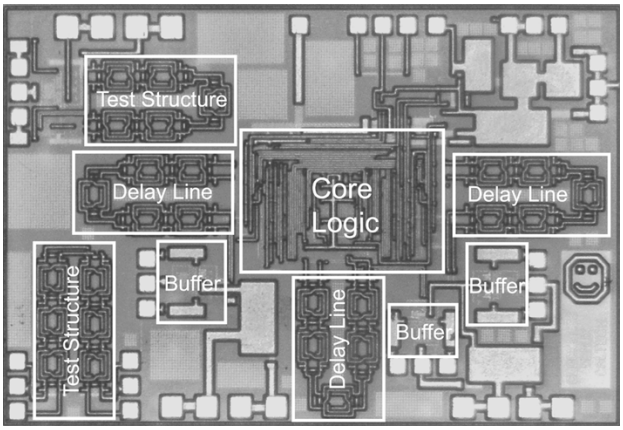

Fig. 5. Die microphotograph of the 1: 2 demultiplexer with three 5-section differential *LC* delay line.

transitions. The y_2 output in Fig. 3(a) demonstrates this. When input is zero, y_2 oscillates for four cycles. As soon as the data transition arrives in the fifth cycle, the y_2 phase is aligned and the oscillation stops. If there were no additional input transitions y_2 would oscillate again with corrected phase. Now, the replica of the 1:2 demultiplexer with y_2 as the output forms the ring oscillator in the delay control loop. This ring oscillator includes all the blocks that contribute to the delay. Fig. 3(b) illustrates the architecture. The same architecture could be used to design a variable bit rate demultiplexer by adjusting the delay around the loop based on the input bit rate.

The delay block can be implemented using active [6] or passive [7] delay elements. If delay control loop is not used, passive delay cells based on inductor–capacitor (*LC*) ladder structures can be used, as shown in Fig. 4(a). The *LC* delays have potentially very low sensitivity to process technology variations as shown in [7]. The delay is determined by the value of the passive components from $T_D = k\sqrt{LC}$ where L and C are the inductance and capacitance, respectively, and k is the number of sections in the ladder.

V. Prototype Measurement Results

An integrated 1:2 demultiplexer based on the instantaneous clockless architecture is fabricated in SiGe BiCMOS process technology. SOP logic functions form the FSM as described in the previous section. The logic functions are realized by emitter coupled logic (ECL) gates. Fig. 4(b) shows a 3-input *OR* gate. When any of the three inputs is at high logic level, all the tail current, I_t, runs in the left branch of the emitter coupled stage,

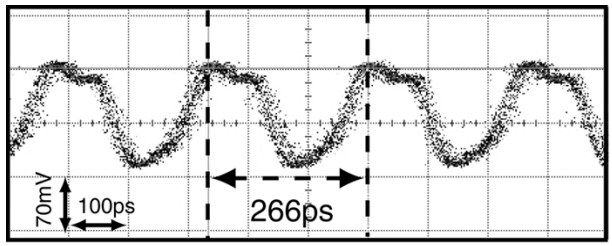

Fig. 6. The y_2 output in the oscillator mode.

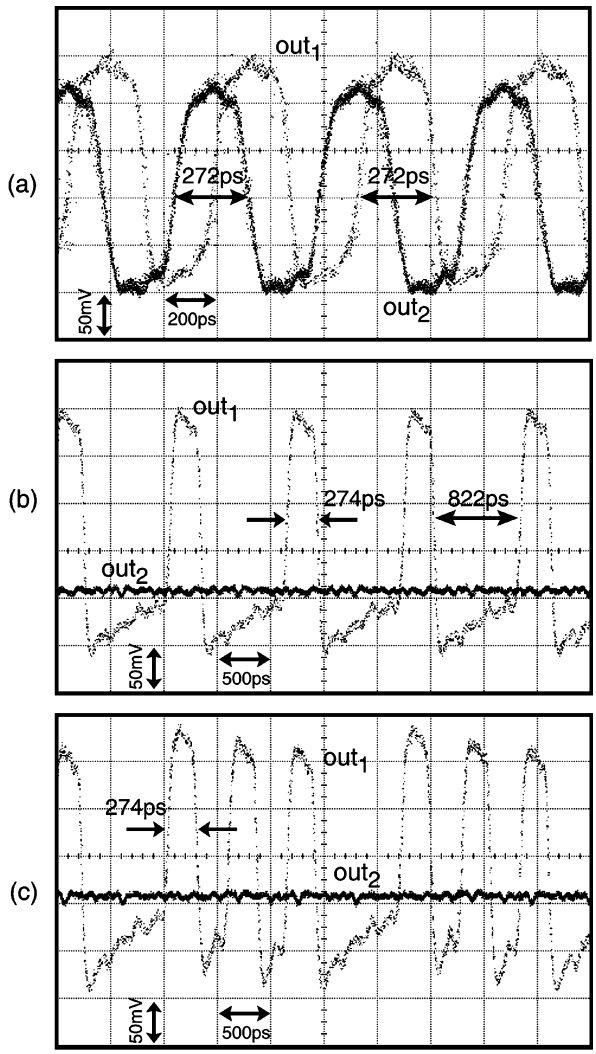

Fig. 7. Demultiplexer outputs out_1 and out_2 for 3 input sequences (a) 1100 (b) 10 000 000 (c) 1 000 000 010 001 000.

which forces the *out* to high logic level. The *AND* gates are implemented using *OR* gates by inverting the inputs and output. A 5-section differential *LC* delay line is implemented for each of the delay cells in the feedback loop of binary functions y_0, y_1, and y_2. The die photograph is shown in Fig. 5. Chip dimensions are 2.5 mm × 1.7 mm. The core logic occupies only 11% of the total die area.

Fig. 6 is the y_2 output of the demultiplexer when the input is a constant "1." As mentioned, y_2 oscillates with a period equal to twice the total delay around the FSM feedback loop, 266 ps. Therefore, the delay (bit period) is 133 ps and demultiplexer works at input bit rate of 7.5 Gb/s. About 55% of the total delay

is generated by the delay lines and the rest is from the ECL gates and interconnect parasitics.

The two outputs of the demultiplexer, out_1 and out_2 are measured for three different input sequences, as shown in Fig. 7. The "sync" signal from the input signal source is used to trigger the sampling oscilloscope for viewing the outputs and justifies that the outputs are synchronized with the input. When the input is a repeating "1100" sequence, the demultiplexed outputs should both be "10" sequence repeating at half the bit rate, i.e., twice the bit period, of the input. In addition, out_2 is one input bit period delayed with respect to out_1. Fig. 7(a) shows these outputs.

A data transition can experience different delays when going through a linear delay cell, e.g., an *LC*-ladder delay cell, based on previous data bits. Delay is defined as the time difference of the threshold crossing times of the data transition, e.g., a "$1 \rightarrow 0$" falling transition, at the input and output. The threshold crossing time is affected by the bits arrived at the delay cell prior to the transition due to the memory of the delay cell [8], [9]. For example, "010" and "110" sequences will have different threshold crossing times at the output of the delay cell. In the latter sequence the residue of the last bit before the falling transition impacts the threshold crossing time. This data dependent delay affects the overall delay in the FSM feedback loop. There are three delay cells for y_0, y_1, and y_2 in the FSM and the input to each depends on the FSM input sequence. Thus, each of the individual delay values may vary based on the input sequence. Furthermore, the sequences to the input of the delay cells are not necessarily the same. We define the FSM loop delay as the average of the three delays, while this average depends on the FSM input sequence. For instance, when the FSM input is a constant "1," the three delay cells for y_0, y_1, and y_2, respectively, see a constant "1," a constant "1," and "10" sequences at their input; whereas for a "1100" at the input the delay cells respectively experience "1100," "1100," and "10" sequences. The inputs to the y_0 and y_1 have changed. Consequently, the average loop delay changes. The two cases are shown in Figs. 6 and 7(a), where average loop delay changes from 133 ps (half of 266 ps) in the former to 136 ps (half of 272 ps) in the latter. One way to avoid data dependent delay values is to use nonlinear (digital) delay cells as opposed to linear *LC*-delay cells.

Fig. 7(b) shows the outputs when input sequence is "10 000 000" that has seven consecutive zeros. The two outputs are respectively "1000" at half bit rate and all-zero. The droop for long sequences of one is due to ac-coupled outputs which will not be present in a dc-coupled version. A longer sequence, i.e., 16 bit, is tested in Fig. 7(c). The input sequence, "1 000 000 010 001 000," results in the outputs "10 001 010" at half the bit rate and all-zero. The demultiplexer is locking to the input phase and correctly demultiplexes to two outputs without using a synchronous clock. The chip is using a 3.3-V power supply and draws 316 mA of current, of which 110 mA is flowing in the output buffers and bias circuits.

VI. Conclusion

An new architecture is introduced that instantaneously recovers and demultiplexes data without explicit clock recovery. The architecture is based on a FSM that assigns input to a proper output and maintains the value of other outputs. State transitions are synchronized with the arrival of input data transitions. Binary logic functions map the current state along with the input bit to the next state. Analog delay cells with bit period delay feedback the value of the binary functions to the input and synchronize FSM with the input data. An architecture is introduced for tuning the overall loop delay. A prototype 1:2 demultiplexer is demonstrated with integrated passive *LC* delay lines and operates at 7.5 Gb/s without using a clock signal.

Acknowledgment

The authors thank J. Buckwalter for his feedback on the manuscript, H. Hashemi and A. Komijani for their contributions to the layout, and S. Mandegaran for helping with Fig. 4(b).

References

[1] L. M. Devito, "A versatile clock recovery architecture and monolithic implementation," in *Monolithic Phase-Locked Loops and Clock Recovery Circuits*, B. Razavi, Ed. New York: IEEE Press, 1996, pp. 405–420.

[2] A. E. Dunlop, W. C. Fischer, M. Banu, and T. Gabara, "150/30 Mb/s CMOS nonoversampled clock and data recovery circuits with instantaneous locking and jitter rejection," in *Dig. Tech. Papers Int. Solid-State Circuits Conf. ISSCC'95*, vol. 38, Feb. 1995, pp. 44–45.

[3] M. Nakamura, N. Ishihara, and Y. Akazawa, "A 156 Mbps CMOS clock recovery circuit for burst-mode transmission," in *Dig. Tech. Papers Symp. VLSI Circuits*, Jun. 1996, pp. 122–123.

[4] S. Kobayashi and M. Hashimoto, "A multibitrate burst-mode CD circuit with bit-rate discrimination function from 52 to 1244 Mb/s," *IEEE Photon. Technol. Lett.*, vol. 13, no. 11, pp. 1221–1223, Nov. 2001.

[5] J. F. Wakerly, *Digital Design: Principles and Practices*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1994.

[6] J. Buckwalter and A. Hajimiri, "An active analog delay and the delay reference loop," in *Dig. IEEE RFIC Symp.*, Jun. 2004, pp. 17–20.

[7] B. Analui and A. Hajimiri, "Statistical analysis of integrated passive delay lines," in *Proc. IEEE Custom Integrated Circuits Conf. (CICC'03)*, Sep. 2003, pp. 107–110.

[8] B. Analui, J. Buckwalter, and A. Hajimiri, "Data dependent jitter in serial communications," *IEEE Trans. Microw. Theory Tech.*, to be published.

[9] J. Buckwalter, B. Analui, and A. Hajimiri, "Predicting data dependent jitter," *IEEE Trans. Circuits Syst. II: Expr. Briefs*, vol. 51, no. 9, pp. 453–457, Sep. 2004.