Self-Timed Circuits Using DCVSL Semi-Bundled Delay Wrappers

Jung-Lin Yang and Erik Brunvand

Institute of Electronic Engineering
Southern Taiwan University of Technology, Tainan County 71005, Taiwan (R.O.C.)
School of Computing
University of Utah, Salt Lake City, UT 84112, USA
jyang@mail.stut.edu.tw and elb@cs.utah.edu

ABSTRACT

We present a technique for generating robust self-timed completion signals for general dynamic datapath circuits. The wrapper circuit is based on our previous domino semi-bundled delay (SBD) circuits, but uses DCVSL circuits in the wrapper for higher performance. We describe the basic SBD-DCVSL building blocks in the template with respect to their circuit structures and operational behavior. These DCVSL SBD circuits show better performance, exhibit reduced overhead, and require reduced operating margins for the matched delay compared with the domino version. The DCVSL wrapper can also identify a class of delay faults in the data path.

1. INTRODUCTION

There have been many publications describing different styles of asynchronous controller circuits and the design synthesis tools used to generate them [1], [2], [3], [4]. Some researchers have argued strongly that these controller synthesis tools can also be used for datapath circuits [5]. However, the requirements for the design of the asynchronous controllers and for self-timed functional blocks are quite different. A tool that is suitable for a controller may not be efficient for datapath circuitry. Also, controller synthesis tools generally have limits on the size and complexity of their target circuits that limits their usefulness when building practical datapath circuits [6].

One area where datapath circuits differ substantially from self-timed control circuits is in their hazard behavior. Control signals in a self-timed system must be hazard-free at all times so that glitches won't be interpreted as controlling signals. Functional blocks typically are allowed to be hazardous while they are settling to their final result. Removing all hazards in the datapath circuits by using controller synthesis tools would result in significant overhead.

2. BACKGROUND

Self-timed components are defined as circuits that generate their own completion signals. In general, self-timed datapath components generate a completion signal in one of two ways: using a matched bundling delay, or using a multiple-wires-per-bit scheme that allows the data itself to encode completion.

A matched bundling delay of a self-timed function block is just a separate delay circuit that models the time a function takes to produce its outputs [7]. The challenge in this approach is to build a delay that will not only model the delay of the function but will also track the delay difference caused by temperature and supply voltage variations. Simple bundling data circuits can be made using inverter chains with a predicted safety margin. Another design alternative is to use multiple bundled delays and to determine on-the-fly which delay is the best match given the specific inputs [8], [9].

Dual-rail signaling is the most common practice for multiple-wires-per-bit encoding schemes [10]. In dual-rail signaling, control of each bit is encoded by two separate wires. The major drawbacks of dual-rail circuits include the increased number of wires, the necessity of designing new datapath circuits that use a multiple-wire protocol instead of using standard datapath library circuits, and the time required to detect completion on all bits of a large data bundle. These circuits are robust and reliable but slower and consume more power than an equivalent bundled circuit.

3. SELF-TIMED WRAPPER

Our self-timed wrappers allow dynamic circuits to be used in a self-timed environment where completion signals must be generated. We take advantage of two properties of the dynamic circuits. First, they have a reset phase where all outputs are reset to a known value (typically a precharge phase). Secondly, the output transitions from the reset state are monotonic. Based on these properties our wrappers can monitor the output signals by detecting the completion of the function when a transition occurs. If no output transition is seen, or the permitted time period for the bundling delay has expired, then we can assume that no transition will take place and the logic has reached its final value.

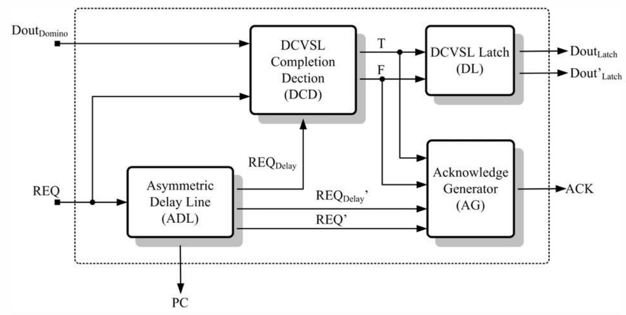

Figure 1. SBD-DCVSL Self-timed Wrapper

2.1 SBD-DCVSL Implementation

The SBD wrapper circuit consists of four major sub-blocks (Figure 1). They are DCVSL completion detection (DCD), asymmetric delay-line (ADL), differential latch (DL), and acknowledge generator (AG). Each block in the wrapper performs similar tasks to its corresponding block in the SBD-Domino wrapper [12], except that no precharge controller is needed for the wrapper, and the differential latch is more like a 6-transistor RAM cell rather than a complicated edge-triggered D-FF. Thus, the SBD-DCVSL is much smaller than its SBD-Domino counterpart in terms of transistor count.

There is also an internal protocol difference. The data precharge is started when the *REQ*↓ arrives, so this precharging operation is no longer in parallel with the handshaking overhead. In return, however, we gain a delay fault recovery technique that allows flexibility in the safety margin added to the matched delay line.

2.1.1 Asymmetric Delay-Line (ADL)

The ADL block function is to postpone the rising edge of the request signal for the matched (worst-case) delay of the functional block. The ideal ADL should propagate the falling edge to its output (REQ_{Delay}) without any additional delays. Thus, a simple inverter chain is not suitable. The ADL block is implemented using a delay element described in [12]. The *REQ*↑ will be delayed for the elapsed time of the matched delay(s), but the *REQ*↓ is propagated to the REQ_{Delay} immediately.

The REQ_{Delay} acts the same way as in the domino wrapper [12]: it models the worst-case delay of all possible input combinations. If the evaluation rail of the DCVSL circuit does not change when the REQ_{Delay}↑ arrives, the wrapper will assume the evaluated output is the complemented value of the evaluation rail. We can see how this works when we introduce the DCD block in the later section.

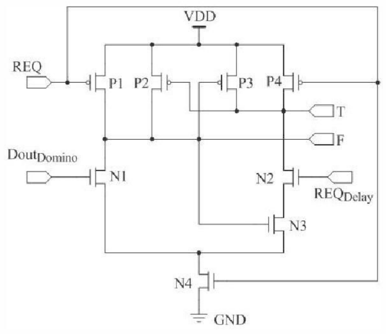

Figure 2. DCVSL Completion Detection (DCD)

2.1.2 DCVSL Completion Detection (DCD)

The DCD block (Figure 2) is a modified DCVSL template structure. When the *REQ*↑ arrives, both the *T* and the *F* outputs are pulled high, and the DCVSL gate holds its result until the next evaluation result is ready. As in all DCVSL circuits, the transistors *N4*, *P1*, *P2*, *P3*, and *P4* are used to control precharging and evaluating phases. We add *N1* as a sensor transistor to evaluate the result from the dynamic functional block. The *N2* transistor is the same as *N1*, except it turns on by the delayed *REQ* signal (REQ_{Delay}↑). For the evaluation rail, when *N1* is on it means that we have an early completed result. For the worst-case matched delay rail *N2* will be on when the delay expires. Thus, we may abort the functional block when the evaluated rail detects early completion. *N3* is connected to *F* so it can ignore the worst-case assumption when the true early completion takes place.

Two other notes on the DCD block are: N1 can be replaced with the last stage of the functional block regardless of whether the wrapper structure is connected or not, and this DCD design provides a technique to monitor delay faults. If the matched delay time is not long enough, the output T will generate a short pulse during the evaluation period (REQ=high). This property is used for designing simple delay fault self-testing circuitry.

2.1.3 Differential Latch (DL)

The differential latch (DL) is implemented as a 6-transistor RAM cell and acts as a SR flip flop. The DL block accepts three input combinations. When both the *T* and *F* are precharged to high, the DL holds its current value. When either *T* or *F* has fallen, the back-to-back inverter pair may set or reset the stored value. The *T* and *F* cannot be pulled down at the same time; otherwise the latch is unpredictable in the stored values. In our DCVSL wrapper, this last situation cannot occur.

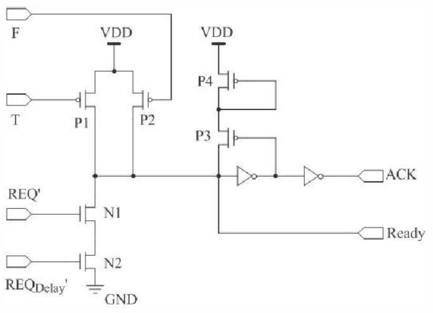

Figure 3. Acknowledge Generator (AG)

2.1.4 Acknowledge Generator (AG)

The last stage of our SBD-DCVSL wrapper is the acknowledge generator (AG) as shown in Figure 3. After *REQ*↓ arrives, both the *T* and *F* are pulled up, and the *REQ'* and the *REQ_{Delay}'* goes high at the same time. *ACK*↑ will then arrive after two inverter delays. The inverters are used to match the DL's setup/hold time and as a buffer to the *ACK* output. The *REQ'* is uesd to lock the pull down network during the evaluation phase. The *REQ_{Delay}'* is used to extend the *ACK*=high period to make sure all functional blocks have sufficient precharge time. When the environment sees *ACK*↑, it means that the result is latched at the DL block. Then, *ACK*↓ signals that the SBD component is ready for the next request.

3. RESULT AND ANALYSIS

The proposed DCVSL SDB wrapper template has a number of advantages over the described SBD domino circuits in the previous section [12], which includes better delay characteristics for reduction of internal handshaking overhead over the same input pattern, better average evaluation delay for bundled data structure, timing constraint for obtaining shortest elapsed time between subsequent requests, and the ability to sense a type of delay faults. In addition, the template is useful for general DCVSL circuits by leveraging the advantages of using just one rail of the normally dual-rail DCVSL.

3.1 Internal Handshaking Overhead

The proposed circuit architecture is shown in Figure 2. This architecture has a very small reaction delay in the DCD block. The SPICE simulation shows only about 120ps ~ 160ps for *Din*↑ triggering *T*↓ under the TSMC 0.25um/2.5V technology (see Table 1). And, the *REQ_{Delay}*↑ to *F*↓ overhead can be overlapped with the safety margin of the matched delay line. In comparison with SBD-Domino wrappers whose reaction delay time is between 450ps ~ 500ps by using the same technology. And the data shows that our new design suffers much less internal handshaking delay. However, the SBD-Domino wrapper does initiate precharge cycle immediately after the functional block's result has been latched.

TABLE 1. COMPARESION OF TRADITIONAL BUNDLED DATA WITH OUR SBD WRAPPERS

	Internal Handshaking	Additional Pre-charge Cycle Time	Average Evaluation Delay	Power (Wrapper)
Traditional	N/A	T_{pc}	*Equation(1)*	N/A
SBD-Domino	450ps ~ 500ps	T_{pc} - max(300ps,ACK↑ →REQ↓)	*Equation(2)*	~ 80 uw
SBD-DCVSL	120ps ~ 160ps	T_{pc}	≦*Equation(2)*	~ 60 uw
T_{pc}: minimum pre-charge cycle time				

We use the same extensible self-timed adder design in [12] for evaluating the performance gain provided by our latest SBD wrapper. Simulation result confirms that the reduced handshaking overhead of the SBD-DCVSL helps improve the ADD_3 (1220ps ~ 1360ps, 25% improvement) and INC_3 (760ps ~ 1350ps, 10% improvement) modules.

3.2 Average Evaluation Delay

Equation (1) is the average delay when the bundling matched delay and dynamic functional blocks are used. This simple bundled data structure has to match the worst-case transition time with a reasonable safety margin. There is no data dependence under such simple structure, therefore, the average evaluation delay equals the worst-case delay.

$$T_{WC} = \max(\tau_{0\rightarrow 1}) + \tau_{safety} \tag{1}$$

$$T_{SBD}=P_{0\rightarrow 1}\times[\mathrm{avg}(\tau_{0\rightarrow 1})+\tau_{wrapper}]+(1-P_{0\rightarrow 1})\times T_{WC} \tag{2}$$

Equation (2) is applicable for both SBD-Domino and SBD-DCVSL self-timed wrappers. The idea of the technique proposed is to replace one of the two rails with our wrapper and delay line. Thus, we have to separate the delay analysis into two parts. First, when the precharged output of the functional block has dissipated, the average evaluation delay equals its evaluate-to-discharge average case delay. If the value of the output is the same as the precharged output, the average evaluation delay is the same formula as the conventional bundled data structure. The distributions of these two delay factors determine the choice of using on-set or off-set Boolean equations to implement the dynamic functional blocks. Actually, equation (2) is a little pessimistic with respect to the SBD-DCVSL wrapper as will be seen in section 3.4.

3.3 Timing Constraint

The DCVSL wrapper uses the REQ signal as the precharge/evaluation signal for the datapath without any additional circuits except some driving buffers. This makes the SBD-DCVSL wrapper simpler than the SBD-Domino wrapper. However, we need to consider timing constraints for the SBD-DCVSL wrapper and the external controller. For a slow environment, both the wrapper and controller need no timing constraint at all. The "slow" implies that no separation period between two subsequent request times is less than the minimal precharging time. If we do not know such timing information, we have to delay the ACK↓ to precharge both the function block and the wrapper itself.

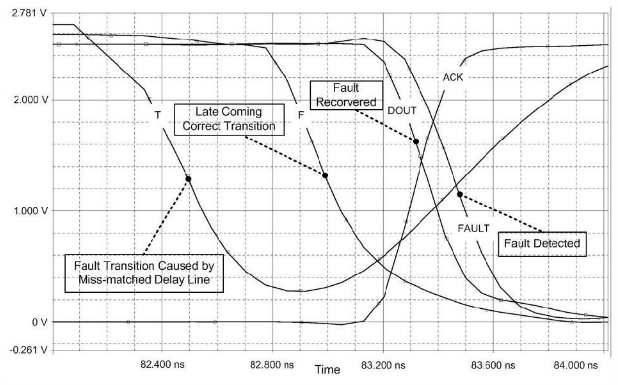

Figure 4. 500ps Delay Fault Recovery Waveform

3.4 Delay Fault Recovery

Finally, the N3 transistor of the DCD block enables a simple delay fault detector that reports a delay fault when the delay line fails to meet the worst-case scenario.

For a rapid data consumer, such as FIFO, this behavior can signal the environment as to the validity of the data. For a controller-based asynchronous environment, the reaction time between 0 to 1ns, a fault can be safely recovered. Figure 4 shows a 500ps delay fault recovery situation. As we observe from the waveform, the output is corrected by the late arriving evaluated result at almost the same time as the acknowledgement signal rises. We can conclude that as long as the environment reacts to the ACK↑ at more than 1 FO4 delay, the data is already stabilized at the correct value. This delay can be included in the safety margin of our matched delay.

3.5 Single-Rail General DCVSL

Both DCD and ADL blocks are needed when our wrapper is used as a generic single-rail DCVSL template. With sufficiently long matched delays, the whole circuit behaves exactly like a DCVSL circuit. In most circuits, the functional blocks are much larger than the matched delay so using the matched delay for one rail of the DCVSL saves power and area. The trade-off is slightly reduced performance. However, if we take the reduced input load into account for overall input-to-output delay, the decrease in speed will be largely compensated for by the "single-rail" structure. Furthermore, the average power used by the wrapper is around 60uw ~ 80uw (Table 1). Thus, low-power consumption is also an expected outcome.

4. CONCLUSION

A new self-timed SBD DCVSL wrapper is described for adding self-timing to datapath circuits in various dynamic logic families. This wrapper implementation makes high-performance dynamic functional blocks accessible in a self-timed system. The wrapper presents a standard self-timed REQ/ACK protocol at its interface and implements precharge/evaluate sequencing, variable-time completion detection, and possible output latching recognition for the dynamic function blocks. In this enhanced wrapper design, we do not merely simplify the wrapper to improve handshaking overhead but also include a delay fault recovery feature to allow flexibility in adding safety margin to the delay line. The proposed new structure also exhibits low power consumption in the completion detection circuitry and the asymmetric delay components.

5. REFERENCES

[1] S. M. Burns, and A. J. Martin, "Syntax-directed translation of concurrent programs into self-timed circuits," in *J. Allen and F. Leighton, editors, Advanced Research in VLSI*, MIT Press, 1988, pp. 35-50.

[2] C. J. Myers and T. H.-Y Meng, "Synthesis of timed asynchronous circuits," in *IEEE Transactions on VLSI Systems, 1(2)*, June 1993, pp. 106-119.

[3] S. M. Nowick, K. Y. Yun, and D. L. Dill, "Practical asynchronous controller design," in *Proc. International Conf. Computer Design (ICCD),* IEEE Computer Society Press, October 1992, pp. 341-345.

[4] V. Yakovlev, A. Petrov, and L. Rosenblum, "Synthesis of asynchronous control circuits from symbolic signal transition graphs," in *S. Furber and M. Edwards, editors, Asynchronous Design Methodologies, volume A-28 of IFIP Transactions*, Elsevier Science Publishers, 1993, pp. 71-85.

[5] J. Sparso and S. Furber, *Principles of Asynchronous Circuit Design: A Systems Perspective*. MA: Kluwer Academic Publishers, 2001.

[6] E. Brunvand, S. Nowick, and K. Y. Yun, "Practical advances in asynchronous design and in asynchronous/synchronous interfaces," in *Proc. ACM/IEEE Design Automation Conference*, 1999, pp. 104-109.

[7] I. Sutherland, "Micropipelines," in *Communications of the ACM, 32(6)*, June 1989, pp. 720-738.

[8] S. M. Nowick, "Design of a low-latency asynchronous adder using speculative completion," in *IEEE Proceedings, Computers and Digital Techniques, 143(5)*, September 1996, pp. 301-307.

[9] S. M. Nowick, K. Y. Yun, P. A. Beerel, and A. Dooply, "Speculative completion for the design of high-performance asynchronous dynamic Adders," in *International Symposium on Advanced Research in Asynchronous Circuits and Systems*, April 1997.

[10] T. Verhoeff, "Delay-insensitive codes-an overview," *Distributed Computing, 3(1)*, 1988, pp. 1-8.

[11] K. M. Fant and S. A. Brandt, "NULL conventional logic: A complete and consistent logic for asynchronous digital circuit synthesis," in *International Conference on Application-specific Systems, Architectures, and Processors*, 1996, pp. 261-273.

[12] Jung-Lin Yang, Erik Brunvand, "Self-Timed Design with Dynamic Domino Circuits," *IEEE Computer Society Annual Symposium on VLSI* (ISVLSI'03).