Optimization design of a full asynchronous pipeline circuit based on null convention logic*

Guan Xuguang(管旭光)†, Zhou Duan(周端), and Yang Yintang(杨银堂)

(Institute of Microelectronics, Xidian University, Xi'an 710071, China)

Abstract: This paper proposes a new optimization method to improve the performance of a null convention logic asynchronous pipeline. Parallel combinational logic modules in the pipelines can work alternately in null and data cycles by using a parallel processing mode. The complete waiting time for both null and data signals of combinational logic output in previous asynchronous register stage is reduced by decoupling the output from combinational logic modules. Performance penalty brought by null cycle is reduced while the data processing capacity is increased. The novel asynchronous pipeline based on asynchronous full adders with different bit widths as asynchronous combination logic modules is simulated using 0.18-μm CMOS technology. Based on 6 bits asynchronous adder as asynchronous combination logic modules, the simulation result of this new pipeline proposal demonstrates a high throughput up to 72.4% improvement with appropriate power consumption. This indicates the new design proposal is preferable for high-speed asynchronous designs due to its high throughput and delay-insensitivity.

Key words: threshold gate; asynchronous circuit; self-timed circuit; high-speed asynchronous pipeline; parallel processing

DOI: 10.1088/1674-4926/30/7/075010 **EEACC:** 1265B

1. Introduction

With the continuous development of semiconductor technology, billions of transistors can be integrated into a single chip. The conventional globally synchronous design is easily affected by problems, such as clock skew, clock jitter, high clock tree power consumption, and EMI. The emergence of the asynchronous design has greatly relieved these problems. Owing to the clockless nature of the asynchronous design, a series of problems can be avoided. As asynchronous circuits are data driven, the circuit goes into operation when data arrives; otherwise it keeps its present state. So, an asynchronous design has the advantages of low-power, anti-EMI, high-robustness, and reusability[1–3].

Among various asynchronous design methods, the NCL (null convention logic)[4] self-timed design method has the most potential. It works under Seitz's "weak condition" delay-insensitive manner[5]. NCL circuits use a four-phase dual-rail protocol where the control signals are integrated into data signals, which can reduce the complexity of the control system.

This paper analyzes principles and performance restrictions of NCL full asynchronous pipelines, and proposes a new design of the high-speed NCL full asynchronous pipeline. In order to enhance the data processing speed and improve the throughput of the whole pipeline, we adopt the parallel processing mode, which enable the parallel combinational logic to work alternately in the NULL mode and the DATA mode. In addition, we decouple the computed results of the present stage from the completion detection signal of the register in the next stage. A comparison between the new proposed pipeline and the conventional pipeline is made. Both theoretical analysis and simulation results show that the new design proposal brings a great performance improvement.

2. Principles of NCL asynchronous pipelines

NCL asynchronous pipelines work under the four-phase dual-rail encode mode. As is shown in Table 1, one dual-rail signal D is transmitted by two mutual exclusion data lines, D0 and D1. Caution should be paid here that D turns to NULL when D0 and D1 are zero, which is known as the return to zero (RTZ) process. RTZ processes, namely reset processes, are needed after every NCL operation cycle.

Threshold gates are the basic logic units of NCL. The main type of the threshold gate is a THmn gate, where $1 \leqslant m \leqslant n$, as shown in Fig. 1. The THmn gate has n inputs with a

Table 1. Dual-rail encoding.

D	D0	D1
DATA0	1	0
DATA1	0	1
NULL	0	0

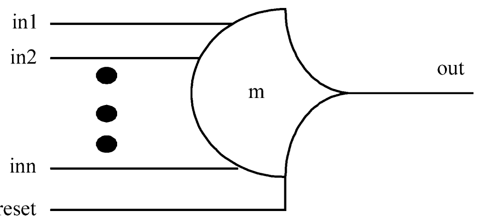

Fig. 1. NCL THmn threshold gate with reset port.

* Project supported by the National Science Fund for Distinguished Young Scholars (No. 60725415) and the National Natural Science Foundation of China (Nos. 60676009, 90407016).

† Corresponding author. Email: guanxuguang_5@126.com

Received 17 November 2008, revised manuscript received 26 February 2009

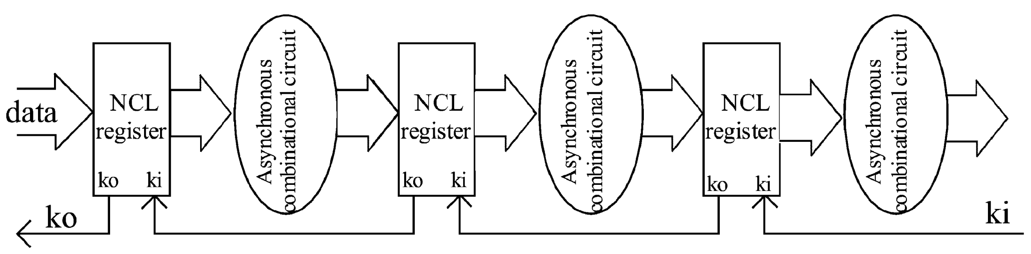

Fig. 2. Structure of the basic NCL pipeline[7].

threshold m. When m out of these n inputs goes high, the output goes high; only in case of all inputs become low, the output becomes low. The hysteresis of the threshold gate guarantees that only after the inputs and the outputs turn to NULL the next set of data can be input into the threshold gate[6].

NCL threshold gates can also have a reset function. The behavior of the NCL threshold gate with a reset port, shown in Fig. 1, can be expressed by

$$Z = (S + (RZ^*)) \times \overline{\text{reset}}\ . \quad (1)$$

S represents the set condition; R stands for the hold condition; Z^* represents the former state of Z; reset is the condition of reset.

NCL systems require at least two delay-insensitive asynchronous registers involved in the input and the output. Data inputs and outputs between registers are controlled by request and acknowledgement signals. Each input request signal of the register comes from the completion detection of the output of the next register. In the beginning, the whole pipeline is initialized to the NULL state with every NCL register requesting data from previous stages, as shown in Fig. 2. To prevent the replacement of present data by the subsequent data, a NULL signal is inserted between two DATA signals; that is, DATA signals and NULL signals are inputted into the asynchronous combinational circuits alternately. An Ack signal is generated by a completion detection circuit, which informs the previous stage that the computation in the present stage has been completed, and it is ready for the next NULL and DATA signals. If the previous stage can not receive the ack completion signal, the previous stage will block the next input signal until the present stage has completed the computation and informed the previous stage through the ack signal that it is ready for new data.

3. Performance analysis

This paper proposes an asynchronous parallel design. Through adding an identical asynchronous combinational logic between two asynchronous registers, the pipelines can work in a parallel mode, as shown in Fig. 3. The black square represents the NCL asynchronous register, while the circle represents asynchronous combinational logic circuits. The arrowhead gives the direction of data flow.

As shown in Fig. 3(a), the data transmissions modes in basic NCL asynchronous pipelines are "empty–full–empty–full". That is, the asynchronous combinational logic in each

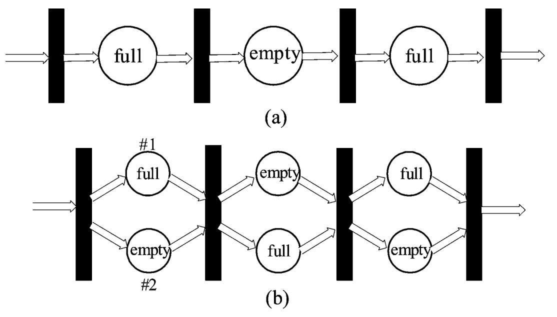

Fig. 3. Comparisons of the two pipelines: (a) Basic NCL transmission graph; (b) Proposed transmission graph in this paper.

stage will go through a NULL period after computation. Every register must wait for the NULL signal to pass through the asynchronous combinational logic and to be successfully received by the next stage register before it can transmit the next DATA signal. In other words, the asynchronous combinational logic will spend half of the time processing NULL signals in a DATA–DATA cycle, which has a negative impact on the throughput of the pipeline. Therefore, the design proposal in Fig. 3(b) uses a parallel processing mode through inserting an identical asynchronous combinational logic between every asynchronous register and by the selection of an asynchronous combinational logic to receive the signal at the input port. In the beginning, the circuits are initialized to a NULL state. When the first DATA reaches the asynchronous register, it will be selected into the asynchronous combinational logic #1, and will inform the present stage register that the data has been successfully received. Then, the present stage register transmits a req signal to the previous stage with a NULL signal at the output. Once the completion detection circuits detect that the asynchronous combination logic have completed the computation, the next stage asynchronous register will receive the computed data. Meanwhile, a NULL signal will be transmitted to the asynchronous combinational circuit from the present stage register, and a completion signal is send to the register in the present stage. The present stage register will get a DATA signal from the previous stage and will wait for the completion of the NULL signal computation. After the computation of the NULL signal has completed, circuit #2 is selected to receive the next DATA signal with the same working procedure of circuit #1. The scheme that circuit #1 and #2 receive DATA/NULL signals alternately has greatly reduced the processing time for the conventional one to complete the DATA/NULL cycle. The whole asynchronous pipelines work in the "full–semi-empty–full–semi-empty" mode with an enhanced throughput.

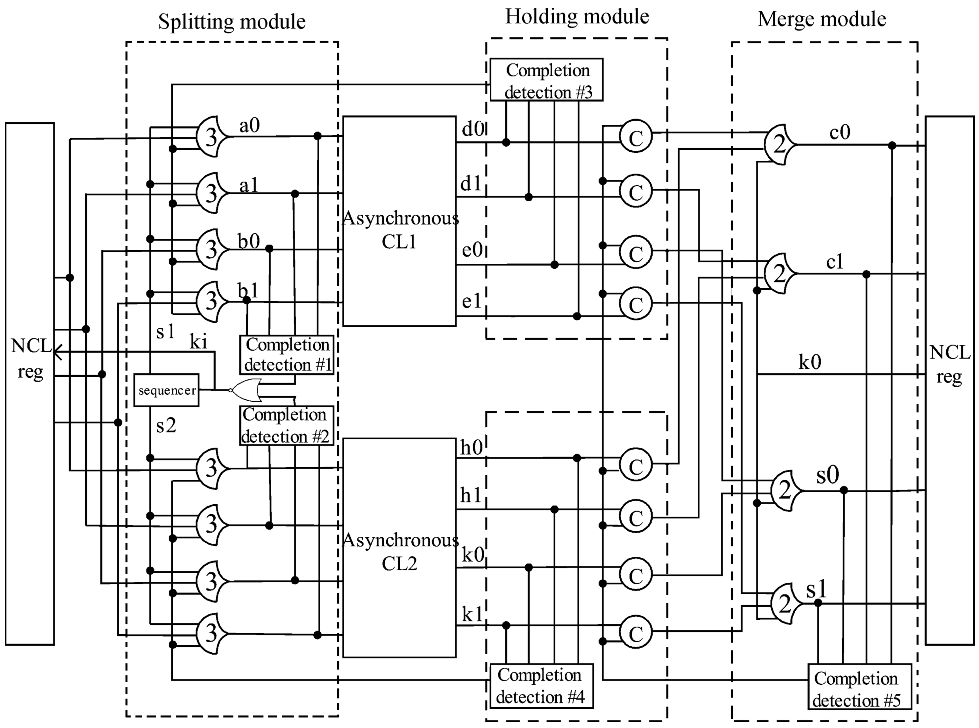

Fig. 4. Structure of the optimized NCL asynchronous pipeline circuit.

4. Circuit implementations

The proposed NCL asynchronous pipeline circuit is shown in Fig. 4. CL1 and CL2 are identical NCL asynchronous logic modules with completion detection circuits at inputs and outputs. Input completion detection is used to control the sequencer to select between CL1 and CL2, while output completion is used to control and hold the inputs of the combinational logic CL1 and CL2. The input completion detection of the next stage register is used to control the output of the present stage. For the circuit in Fig. 4, the outputs of the combinational circuits and the completion detection of the next stage register are decoupled so that it is not necessary to wait for the ack signal coming from the next stage to start the next DATA/NULL cycle, which is different from the scheme in Ref. [9]. Moreover, the output selection is entirely controlled by the completion detection circuit of the next stage, which can reduce extra performance losses due to the sequencer.

The operation procedure of the NCL asynchronous pipeline in Fig. 4 is as follows: In the beginning, all modules are initialized to the NULL state. Owing to the fact that all of the inputs of CL1 are zero, the input and output completion detections of CL1 are valid, which will make the output of the completion detection ki high, and s1 is selected by the sequencer. If DATA is loaded into the NCL register, the th33 gate of CL1 will reach its threshold; so, data can pass through a0, a1, b0, and b1 into module CL1 and begin to compute. If the input completion detection of CL1 has detected the validity of the inputs, the output of the completion detection ki becomes zero, and s1 goes low by the sequencer. The left NCL register will load the NULL signal after receiving the low ki, but the NULL signal can not be transmitted through the th33 gate until receiving the completion signal of the CL1. Thus,

Table 2. Operating process of the sequencer.

Period	ki	s1	s2
#1	1	1	0
#2	0	0	0
#3	1	0	1
#4	0	0	0

it ensures that the previous DATA signal can not be overwritten by a NULL signal. When the outputs d0, d1, e0, and e1 of the CL1 become valid, they will go into the output holding circuits constructed by C-elements. The CL1 circuit can begin to compute immediately, rather than waiting for the output of the next register to become NULL, as is done in conventional NCL asynchronous pipelines. That is, the waiting time for the ack signal is reduced, which makes the asynchronous combinational logic go into the next cycle as soon as possible. The output of CL1 will keep in C-elements instead of receiving a NULL from the output of CL1, until the data in d0, d1, e0, and e1 have been transmitted to c0, c1, s0, and s1. Also, the th23 gate is allowable to return to the NULL state, since k0 becomes low after the next asynchronous register has sent out the data successfully. By now, a DATA/NULL cycle has been completed. The working procedure of the CL2 circuit is the same as that of the CL1. The right working order of CL1 and CL2 is determined by the sequencer, and the behavior of the sequencer can be depicted by Table 2.

The implementation circuit of the sequencer is shown in Fig. 5. The sequencer selects the output through the input times of ki. The last input can be preserved in the circuit due to the hysteresis of the threshold gates until the input ki changes, which alternately causes s1 and s2 to go to the high state. The specific working process of the sequencer is depicted

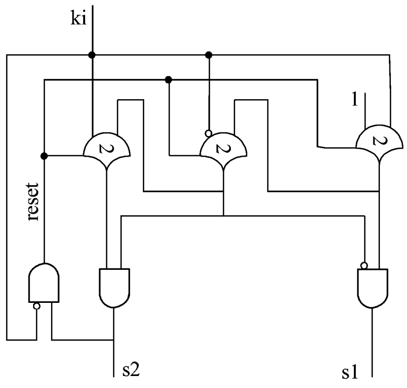

Fig. 5. Implementation of the sequencer circuit.

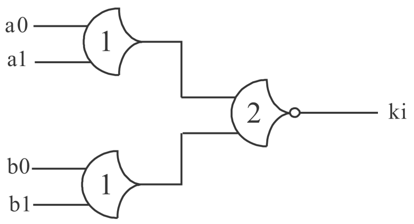

Fig. 6. Completion detection circuits[8].

in Table 2. Ki is controlled by the outputs of the completion detections circuits of CL1 and CL2; s1 goes high when ki becomes high first, while s1 and s2 go low after ki becomes low. When ki becomes high afterwards, s2 goes high. The mutual exclusion behavior of s1 and s2 guarantees that CL1 and CL2 can receive data alternately, which can greatly enhance the throughput of the pipeline.

Figure 6 shows the completion detection circuit with the function of detecting the validness of DATA and NULL signals. The number of th12 gates here should equal to the widths of signal to be detected. In the beginning, a0, a1, b0 and b1 are all low, which indicates requesting DATA signals. Th22 gates will go low after a and b get data, and requesting NULL signals. With the NULL signal arrives, the output of th22 becomes high and requesting DATA signal, which indicates a detection cycle has completed.

But in practical applications, congestions may often happen. So sometimes it is necessary to make some changes to output holding module to prevent data errors. Specific circuits are shown in Fig. 7, and this module can substitute the module with dashed line in Fig. 4.

After the output holding module in Fig. 4 changed to the circuit of Fig. 7, outputs of CL1 and CL2 are controlled by completion detection module #3 and #4 respectively, only after the output of CL1 (d0, d1, e0, e1) changed to zero, namely in the NULL cycle can output of CL2 (h0, h1, k0, k1) begin to go through th33 gate, and vice versa. So the two output holding module can work in a mutual exclusion manner, and congestions in the pipeline can be removed.

5. Simulation results and analysis

A spice simulation is made based on SMIC 0.18-μm CMOS technology with a voltage of 1.8 V. Figure 8 is a part of the simulation waveform, in which a0, a1, b0, and b1 are inputs of the full-adder, and rs is the reset signal. The circuit needs to reset before it can properly work. After the reset, ki is high, which will make the output of the sequencer s1 turn high, and CL1 is selected to process data. When the combinational logic CL1 has completed the computation, a completion detection signal will be sent to the register, and the NULL signal begins to enter CL1. This is the RTZ (return to zero) process. CL2 will be enabled after the ki signal rises again, and the next data can be sent to CL2 from the register, as shown in Fig. 8. The outputs of the sequencer s1 and s2 go high alternately, which causes CL1 and CL2 to process the input data alternately.

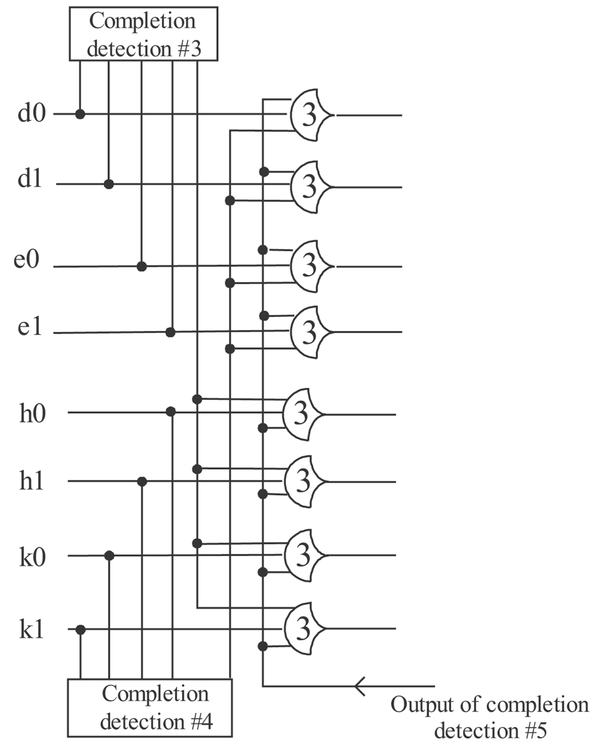

Fig. 7. Congestion immunization module.

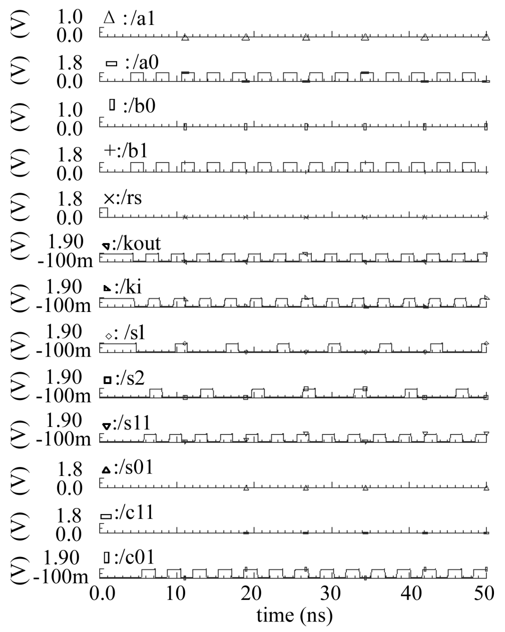

Fig. 8. Simulation waveform of the new proposed pipeline.

Table 3. Performance comparisons of the two pipelines (five stages pipelines).

Asynchronous CL	Conventional pipeline DATA–DATA cycle[10] (ns)	Proposed pipeline DATA–DATA cycle (ns)	Throughput improvement
2 bits full-adder	1.93	1.42	35.9%
3 bits full-adder	2.21	1.54	43.6%
4 bits full-adder	2.64	1.63	62.3%
5 bits full-adder	2.92	1.75	67.1%
6 bits full-adder	3.24	1.88	72.4%

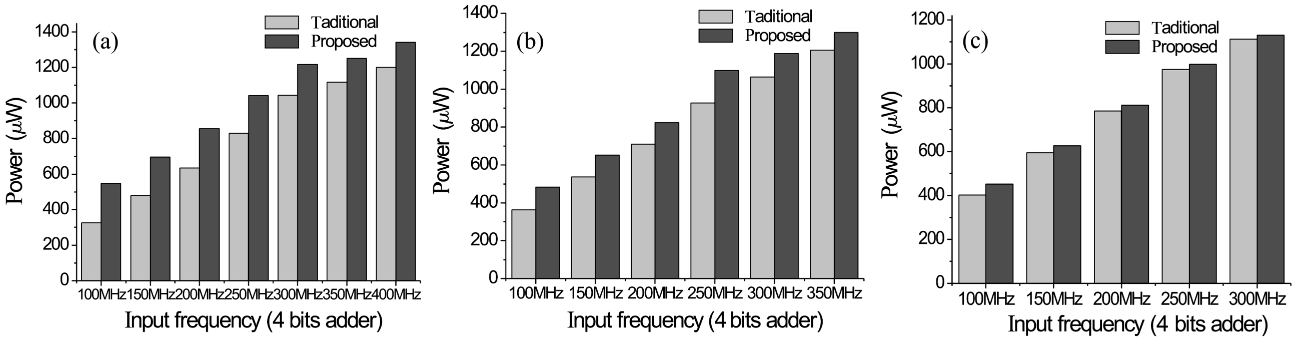

Fig. 9. Power consumption analysis of two pipelines: (a) 4 bits adder as asynchronous CL; (b) 5 bits as asynchronous CL; (c) 6 bits adder as asynchronous CL.

A comparison between conventional NCL asynchronous pipelines and the proposed pipelines based on different widths of the asynchronous full-adder is made, as shown in Table 3. Both of the two pipelines use five stages pipelining.

From Table 3, we can find that the larger asynchronous combinational logic leads to a higher throughput and a performance improvement of the optimized NCL pipelines. Based on a 6 bits full-adder as the asynchronous combinational logic, the throughput is increased by 72.4%. Since the performance of the asynchronous circuits is decided by the average case of the circuits, the throughput here indicates the average throughput, i.e., the throughput arithmetic mean value of all kinds of input combinations, which can be described as

$$\overline{G} = \frac{\sum_{i=1}^{n} G_i}{2^n}, \tag{2}$$

where G_i is the throughput of the pipeline under the given combinational inputs. 2^n is the number of possible combinational inputs of the n-bit pipeline.

This paper applies the optimization method to the one stage of the 10 stages asynchronous pipeline, and tests the average power consumption per stage in the pipeline, as shown in Figs. 9(a), 9(b) and 9(c). Comparisons are made based on 4 bits, 5 bits and 6 bits adder as asynchronous combinational logic module respectively.

It can be found that the power consumptions of two pipelines are getting closer to each other gradually as the input frequency and the complexity in the combinational logic increases. When the 6 bits adder serves as an asynchronous combinational logic module and the input frequency of the pipeline is 300 MHz, the power consumption difference is less than 0.016%. This is because the frequency of the asynchronous combinational logic in the proposed pipeline is one-half of that in a traditional pipeline for the same input frequency. This reflects the power consumption more obviously as the increasing complexity in the asynchronous combinational logic module.

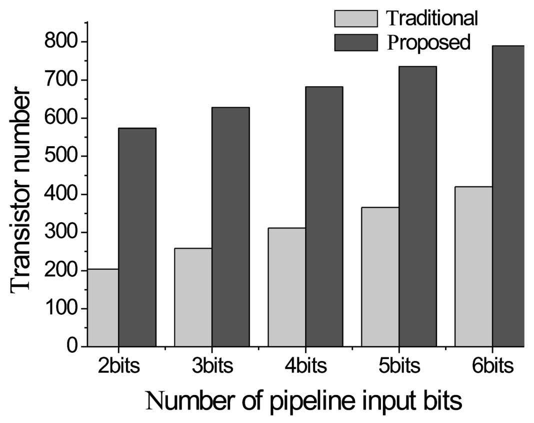

Fig. 10. Area comparisons of two pipelines.

A comparison of the transistor numbers of the two pipelines is presented in Fig. 10.

As can be seen from Fig. 10, owing to the parallel processing mode, the area of the proposed pipeline is about twice than that of the traditional one.

The performance of the proposed pipeline has great advantages as compared to the traditional one under the circumstances of complexity in the combinational logic. So, it is suitable for the slower pipeline stage to enhance the throughput of a whole pipeline in a data path. In asynchronous system designs, a slower pipeline stage often becomes a performance bottleneck of the whole system. Therefore, it is acceptable to increase the area appropriately to get a tremendous performance enhancement with little difference in power consumption when applying it to high speed systems.

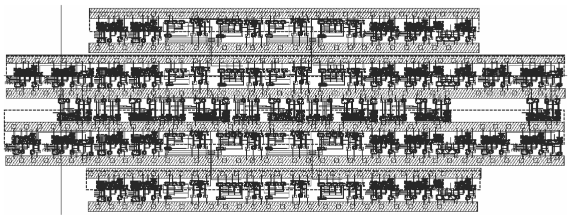

Fig. 11. Layout of the proposed pipeline.

A full-custom layout is implemented using SMIC 0.18-μm 1P6M technology, as shown in Fig. 11. The circuit has successfully passed the post-simulation and the area of the layout is 12783.193 μm^2. The proposed pipeline has already been integrated into our NOC asynchronous routers and has lead to a great performance enhancement with high robustness.

6. Conclusions

This paper proposes an optimized design for NCL asynchronous pipelines. It resolves the performance loss of NCL asynchronous pipelines when processing NULL signals. Two identical asynchronous combinational logic modules are used to realize parallel processing, and the selection of the modules is controlled by completion detection signals. The advantage of this method is that the asynchronous combinational logic can go into the next computation cycle before the output completion signal of the next stage arrives. The data-hold circuits constructed by c-elements ensure that data is stable before being successfully received by the next register, thus avoiding the possibility of data errors. As seen from the spice simulation, the proposed pipeline has a great advantage over the conventional one. Especially under the circumstance of a 6 bits asynchronous full-adder as the asynchronous combinational logic module, the throughput of the pipeline is increased by 72.4%. The proposed pipeline is suitable for high-speed on-chip asynchronous designs.

References

[1] Shan Y, Lin B. Custom networks-on-chip architectures with multicast routing. IEEE Trans VLSI Syst, 2009, 17(3): 342

[2] Geer D. Is it time for clockless chips. Computer, 2005, 38(3): 18

[3] Saneei M, Afzali-Kusha A, Navabi Z. A low-power high-throughput link splitting router for NoCs. Journal of Zhejiang University: Science A, 2008, 9(12): 1708

[4] Fant K M, Brandt S A. NULL convention logic: a complete and consistent logic for asynchronous digital circuit synthesis. International Conference on Application Specific Systems, Architectures, and Processors, 1996: 261

[5] Seitz C L. System timing, in introduction to VLSI systems. Addison-Wesley, 1980: 218

[6] Sankar R, Kadiyala V, Bonam R, et al. Implementation of static and semi-static versions of a 24 + 8 × 8 quad-rail NULL convention multiply and accumulate unit. IEEE Region 5 Technical Conference, 2007: 53

[7] Kuang W, Yuan J S, DeMara R F, et al. Performance analysis and optimization of NCL self-timed rings. IEE Proceedings Circuits, Devices and Systems, 2003: 167

[8] Di J, Yuan J S. Energy-aware design for multi-rail encoding using NCL. IEE Proceedings Circuits, Devices and Systems, 2006: 100

[9] Smith S C. Speedup of NULL convention digital circuits using NULL cycle reduction. Journal of Systems Architecture, 2006, 52(7): 411

[10] Kakarla S, Al-Assadi W K. Testing of asynchronous NULL conventional logic (NCL) circuits. IEEE Region 5 Conference, 2008: 1