An evaluation for the design of asynchronous systems

SUN-YEN TAN [1], WEN-TZENG HUANG [2]

[1] Department of Electronic Engineering
National Taipei University of Technology
No. 1, Sec. 3, Chung-hsiao E. Rd., Taipei,10608, Taiwan, R.O.C.
sytan@ntut.edu.tw

[2] Department of Computer Science and Information Engineering
Mingsin University of Science and Technology
No.1, Xinxing Rd., Xinfeng Hsinchu 30401, Taiwan, R.O.C.
wthuang@must.edu.tw

Abstract: - The asynchronous circuit style is based on micropipelines, a style used to develop asynchronous microprocessors at Manchester University. This paper has presented some engineering work on developing a technique of sharing resources for micropipeline circuits. The work presented in this paper shows a comparison of 2-phase and 4-phase implementations in transistor count, speed, and energy. Though the nature of the work is mainly engineering, there are some significant new insights gained in the course of the work.

In resource sharing the 2-phase implementations have better performance than the four-phase implementations. There is no "return to zero" problem. **Fork** and **join** cost nothing to the two-phase implementations. With some additional buffer stages the 4-phase implementations using the fully decoupled and long hold latch control circuits can also implement resource sharing. However, the four-phase implementations using the simple and semi-decoupled latch control circuits require more buffer stages to avoid deadlock.

Key-Words: - Asynchronous design, Micropipelines, Processor, Sharing resources, Synthesis

1 Introduction

Well structured asynchronous design styles, such as micropipelines, reduce the difficulty. Event-driven logic modules may be designed by electronic experts. Then designers with less experience can easily build micropipelined circuits using such modules. An automatic synthesis tool is available [1][4]. It converts the behavioural VHDL into structural VHDL and Verilog based on micropipelines [1][7][11]. Two-phase and four-phase VHDL models of event-drive logic modules and standard logic function elements were created. In this paper we demonstrate the technique of sharing resources for micropipeline circuits.

Section 2 introduces some asynchronous design techniques. Section 3 briefly describes nicropipelines and introduces 2-phase and 4-phase event-driven Logic modules. The sharing resource design will be presented in Section 4. Section 5 will present experimental results. Section 6 gives an analysis for the synthesized circuits. Finally, Section 7 will give a short conclusion.

2 Asynchronous design

Asynchronous design has potential advantages over synchronous design [8][9][10], such as no clock skew problem, low power, average case performance and good Electro-Magnetic Compatibility (EMC). The benefits may be most apparent in mobile communication applications and other portable systems which use advanced VLSI technologies.

Asynchronous logic circuits have several important advantages over their counterparts in clocked logic. An asynchronous logic function is potentially faster because it works at the average-case delay rather than the worst-case delay. There is no global clock on asynchronous circuits so they will not unnecessarily dissipate power when there is no useful work to do. Asynchronous logic has the potential for low power [5]. Asynchronous logic may be used to implement systems with lower power dissipation.

3 Micropipelines

The design of asynchronous circuits generally follows a modular approach, where a system is designed as an interconnection of modules. In the 1988 Turing

Award Lecture, Sutherland expounded a modular approach to building hardware systems based on data-driven asynchronous self-timed logic elements called micropipelines [6]. In the 4-phase handshaking protocol, only rising transitions or only falling transitions of either control wire have the meaning; they represent request events or acknowledge events. In this signalling scheme, the operating cycle is (1) data available, (2) change request to active state, (3) change acknowledge to active state, (4) return request to inactive state, and (5) return acknowledge to inactive state. If the active state is logic **'1'** the the operating cycle is (1) data available, (2) **request+**, (3) **acknowledge+**, (4) **request-**, and (5) **acknowledge-**. Figure 1 illustrates two kinds of four phase signalling, the **'early'** mode and the **'broad'** mode [2]. The **'early'** mode (Figure 1(a)) uses the rising edge of the **Request line** to indicate **'data available'** and the rising edge of the **Acknowledge line** to indicate **'data latched'**. The falling edges are return to zero actions that carry no meaning. The **'broad'** mode (Figure 1(b)) uses the rising edge of the **Request line** to indicate **'data available'** and the falling edge of the **Acknowledge line** to indicate **'data latched'**. Another possible protocol is **'late'** mode which uses the falling edges as active.

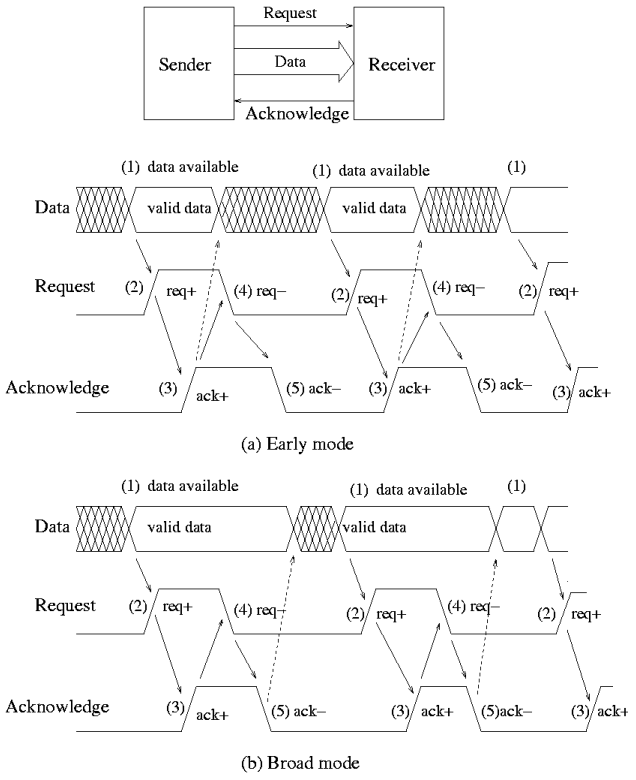


Fig. 1. 4-phase bundled data convention

Various event-driven logic modules for controlling transition signals are shown in Figure 2. They were devised for composing to 2-phase control circuits. Muller C-elements and XOR gates are the same whether they are used in 2- or 4-phase designs. However, 4-phase Toggle, Select, Call and Arbiter modules are different from their 2-phase counterparts. A Toggle is used to alternately deliver events on its input to one of two outputs. In the 2-phase protocol each transition denotes an event. Therefore, the odd number transitions on the input of a Toggle will be sent to the dotted output and the even number transitions on the input of a Toggle will be sent to the non-dotted output. In the 4-phase protocol each event consists of a rising transition and a falling transition. A rising transition and the following falling transition must be sent to the same output. Therefore, the odd number rising and falling transitions on the input of a Toggle will be sent to the dotted output and the even number rising and falling transitions on the input of a Toggle will be sent to the non-dotted output.

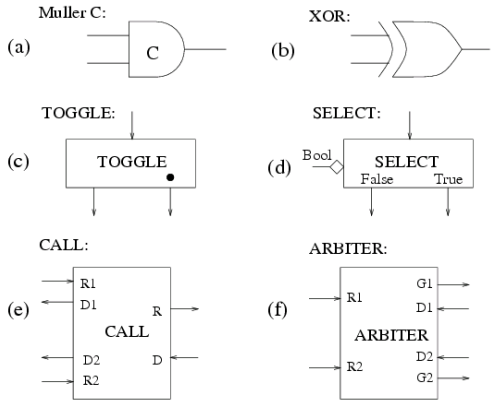


Fig. 2. Various event-driven logic modules.

Furber and Day developed four kinds of 4-phase latch control circuits. They are the simple, semi-decoupled, fully decoupled and long hold 4-phase latch control circuits [3][11]. They use the 4-phase bundled data convention.

4 Sharing resources

Figure 3 shows that the **stg4** stage contains two computations, a 16-bit adder and a 17-bit adder. A resource sharing implementation may be applied to save cost. A 17-bit adder stage can be created and calling this adder stage twice may get the same result. To implement resource sharing a re-partitioning of the stages is required. Latches are put at the input and

output ends of the computations. After putting latches at the input and output ends of the computations inside the **stg4** stage and adding control circuits for the latches some new stages are created as shown in Figure 4. The **stg3** stage forks into the **stg4**, **stg31** and **stg35** stages and the **stg3**, **stg32** and **stg36** stages join into the **stg4** stage.

Some **Arbiter** and **Call** modules are required to build the circuit for connecting different source stages to the resource stage. For example, if there are five source stages Figure 5 shows the **Arbiter** and **Call** circuits which can be used to connect the five source stages to the resource stage. The source stage which is identified as **'1'** has to connect its **Rout** and **Aout** to the **r1** and **d1** inputs which are labelled **1**. The source stage which is identified as **'2'** has to connect its **Rout** and **Aout** to the **r2** and **d2** inputs which are labelled **2**, and so on.

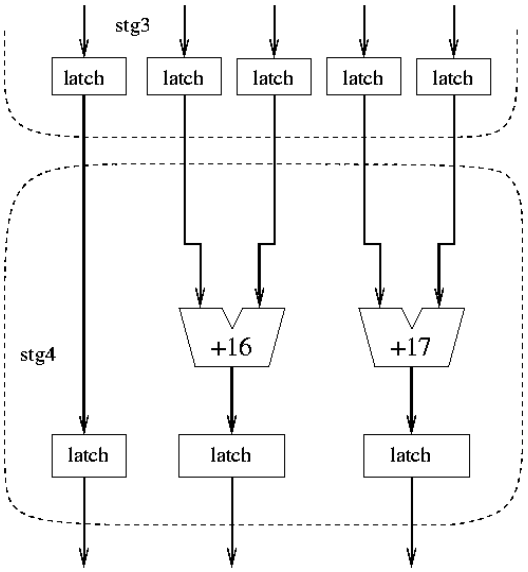


Fig. 3. A stage(stg4) contains a 16-bit adder and a 17-bit adder

Two sets of multiplexer circuits as shown in Figure 5 are required to connect the data from different source stages to the computation device inside the resource stage. The source stage which is identified as **'1'** has to connect two data outputs to the two multiplexers which are labelled **1**. The source stage which is identified as '**4'** has to connect two data outputs to the two multiplexers which are labelled **4**. The signals labelled **s3** are connected together. They are used to control the multiplexers. These connections of the multiplexer controls are for the 4-phase protocol. **Select** and **XOR** circuits are also required to connect the **Rout** and **Aout** of the resource stage to different output stages.

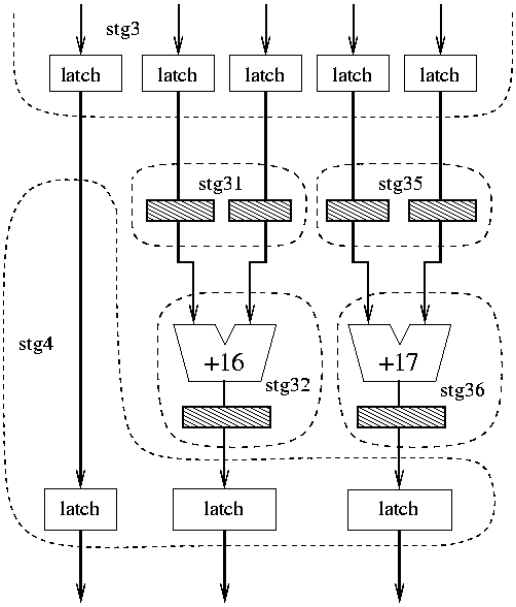


Fig. 4. Some latches are connected at the inputs and outputs of adders.

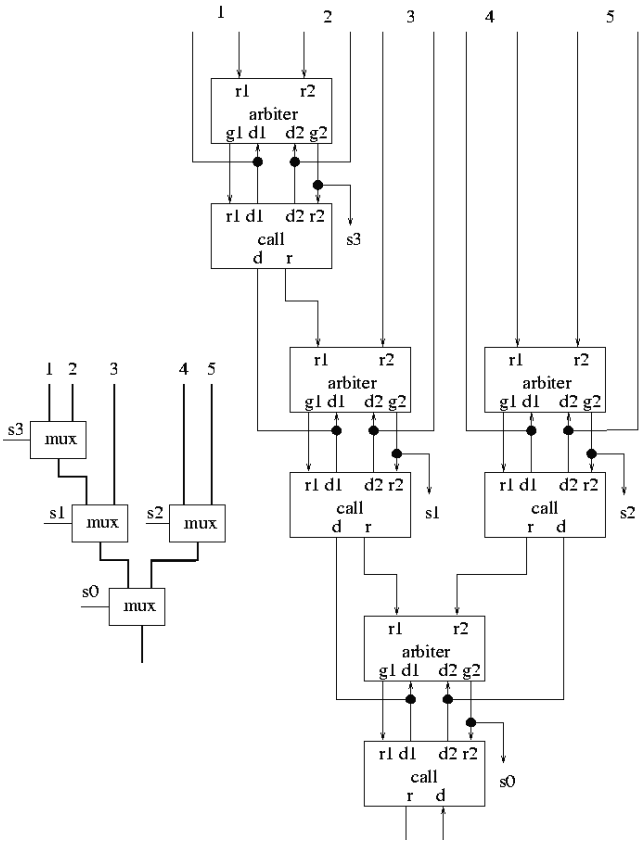


Fig. 5. The Arbiters, Calls and Muxs are connected to control the resource stage receiving requests from different stages.

Figure 6 shows **Select** and **XOR** circuits which can be used to connect a resource stage to five output stages. The output stage which is identified as **'3'** has to connect its **Rin** and **Ain** to the **Select** and **XOR** circuits which are labelled **3**. The inputs labelled **s1**, **s2**, **s3** and **s4** of the latch shown in Figure 6 are generated from the circuits shown in Figure 5. The outputs labelled **b1**, **b2**, **b3** and **b4** of the latch shown in Figure 6 are used to decide where the events should be sent to. They are connected to the inputs labelled **b1**, **b2**, **b3** and **b4** of the **Select** circuits.

Using the above techniques a 17-bit adder resource stage can be created and called by two source stages, **stg31** and **stg35**. The circuit is shown in Figure 7. The resource stage is connected to two output stages, **stg32** and **stg36**. The 2-phase protocol is used in this circuit. The **stg31** stage is connected to the **Arbiter**, **Call** and **Mux** circuits at the position labelled **1**. The **stg35** stage is connected to the **Arbiter**, **Call** and **Mux** circuits at the position labelled **2**. The **stg32** stage is connected to the **Select** and **XOR** circuits at the position labelled **1**. The **stg36** stage is connected to the **Select** and **XOR** circuits at the position labelled **2**.

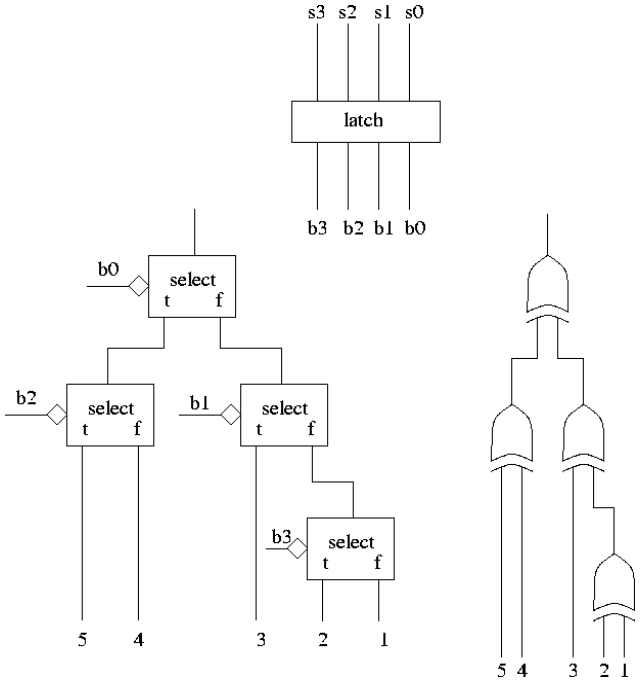


Fig. 6. The selects and XORs are used to deliver the request from the resource stage to the corresponding stage.

Those non-connected inputs of the **MUX**s may need to be connected to logic **'0'** or **'1'** depending on what the computations are. The circuit in Figure 4 may be denoted by Figure 8(a) or 8(b) depending on whether one resource is used or two resources are used. Both circuits which are implemented using a 2-phase protocol run correctly.

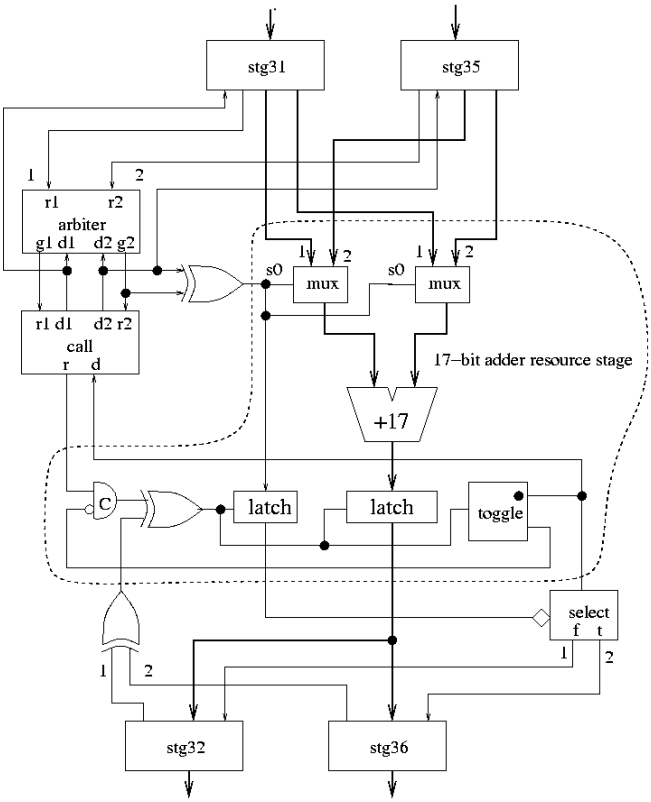


Fig. 7. The stages (**stg31** and **stg35**) use a resource stage (a 17-bit adder) to generate two computations in different time.

However, Figure 8(a) has deadlock if a 4-phase protocol is used to implement the circuit. The 4-phase circuit of Figure 8(b) can run correctly without deadlock. The reason is that the rising transition of the **3** stage is sent to the **4** stage, the **31** stage and the **35** stage at the same time. The acknowledge rising transitions of the **31** stage and the **35** stage are sent out when these two stages hold data. However, the **4** stage is waiting for the rising transitions from the **32** stage and the **36** stage. If the **31** stage first calls the resource stage, the **Rin** of the **38** stage will not return to zero due to the **31** stage can not get the falling transition on its **Rin**. To avoid the deadlock the method shown in Figures 9, 10 and 11 can be applied.

The example shown in Figure 9 illustrates how the steps work. The first step is to create some new latches for those data from the **stg5** stage but not connected to the **stg41** stage. These latches form a new stage called **stg45**. Make the connections

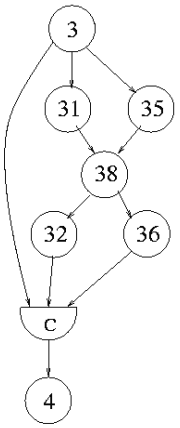
between the **stg5** and the new stage. Remember the destination stage from the **stg5** is the **stg6** stage. After processing the circuit of Figure 9 is changed as shown in Figure 10.

The circuit of Figure 8(a) can use the above method to avoid deadlock. However, the circuit of Figure 10 still contains deadlock. A second step can be followed. The second step is to create some new latches for those data from the **stg45** stage but not connected to the **stg6** stage. These latches form a new stage called **stg46**. Make the connections between **stg45** and the new stage. The final circuit is shown in Figure 11. Now the circuit shown in Figure 11 can run correctly without deadlock. A synthesized processors may contain some 16-bit and 17-bit adders as well as some 16-bit and 17-bit subtractors. One choice is that a 17-bit adder and a 17-bit subtractor process all addition and subtraction operations within different stages. The other is that a 16-bit and a 17-bit adders as well as a 16-bit and 17-bit subtractors process all addition and subtraction operations within different stages.

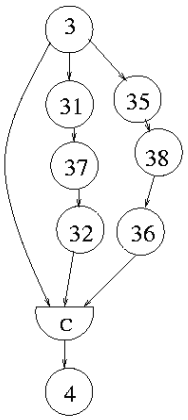
The synthesized 2-phase processor circuit can be converted into resource shared circuits with the exact same size or with a different size. These circuits were simulated correctly using the leapfrog simulator.

The resource shared circuits for the above two cases using 4-phase control circuits have to use the above process to avoid deadlock. An example of the resource shared circuits with common sizes of the synthesized circuits is shown in Figure 12. The **47** stage and the **48** stage are resource stages. They are a 17-bit adder and a 17-bit subtractor. The circuit shown in Figure 12 is after processing. Only synthesized circuits using the fully decoupled and the long hold control circuits were simulated correctly using the leapfrog simulator and PowerMill.

The reason is that the falling transitions of **Rout** in the fully decoupled and the long hold control circuits can be sent out before the falling transition arrives on **Rin**. Therefore, the falling transitions of **Aout** in the fully decoupled and the long hold control circuits can arrive before the falling transition arrives on **Rin**. However, it is necessary to ensure that the falling transitions of **Rout** in the simple and semi-decoupled control circuits are sent out after the falling transition arrives on **Rin**. Deadlocks may happen. For example, as shown in Figure 12, the **11** stage sends rising transitions to the **26** stage, the **28** stage, the **30** stage, the **32** stage, the **34** stage, the **36** stage, the **38** stage and the **40** stage. Then the **26** stage sents the request transition to the **47** stage. After the **47** stage holds data it sends a rising transition to the **27** stage. The **27** stage is one of eight stages which connect to the **12** stage.



(a) 2 sources and 1 resources



(b) 2 sources and 2 resources

Fig. 8. The stages (the **31** stage and the **35** stage) use a resource stage (a 17-bit adder) to generate two computations in different time.

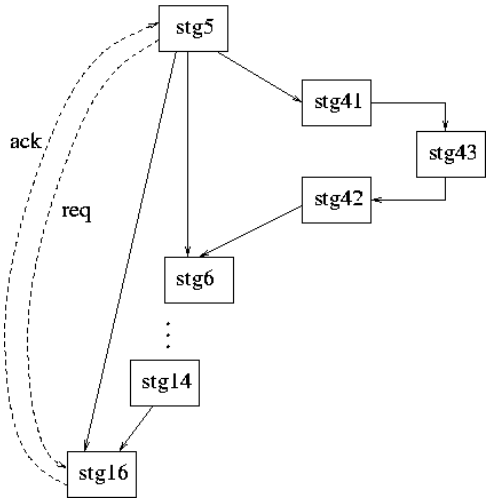


Fig. 9. To avoid deadlock in 4-phase circuits some additional stages are required in the path(1).

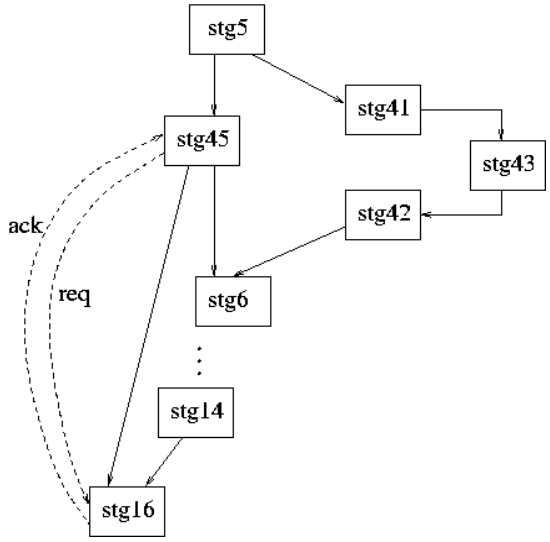


Fig. 10. To avoid deadlock in 4-phase circuits some additional stages are required in the path(2).

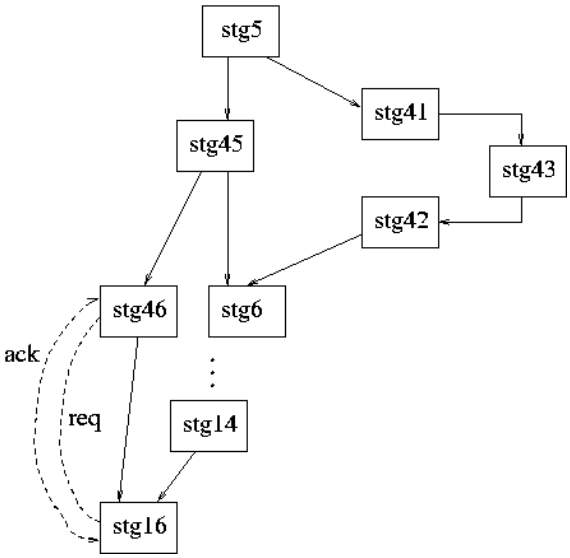


Fig. 11. To avoid deadlock in 4-phase circuits some additional stages are required in the path(3).

It is necessary to wait for these eight stages to send requests to the **12** stage. Then the **12** stage will send the acknowledge to these eight stages. However, four stages will still wait to send requests to the **47** stage. On the other hand, the **29** stage, the **35** stage, the **37** stage, the **39** stage are still waiting for data from the **47** stage. It is then impossible to get the rising transition from the the **12** stage. As shown in Figure 13, the **Rout** of the **27** stage stays at logical **'1'** and the logical **'0'** on the **Rin** of the **27** stage cannot pass the **C**-gate without a rising transition on the **Aout** of the **27** stage.

Eight extra buffer stages are required to connect from the **27** stage, the **29** stage, the **31** stage, the **33** stage, the **35** stage, the **37** stage, the **39** stage and the **41** stage to the **12** stage individually. In total twelve extra buffer stages are required to ensure that the resource shared 4-phase simple and semi-decoupled Stump processors can operate properly.

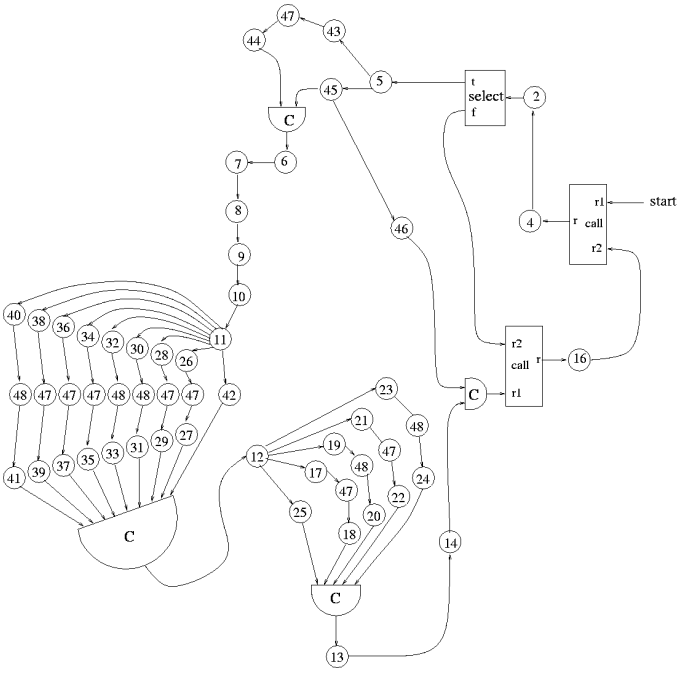


Fig. 12. Resource stages (**47** and **48**) are called from different stages without deadlock.

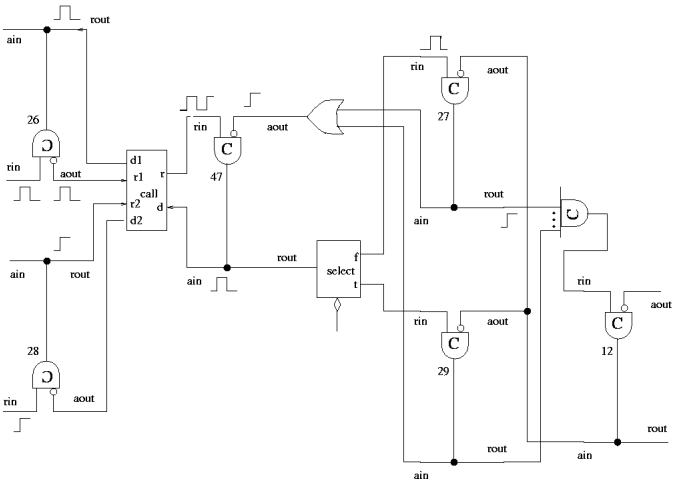


Fig. 13. Four stages of the simple control circuits

5 Experimental results

Table 1 The performance of the resource shared Stump processors

Circuit Name	Transistors (piece)	Run Time (μs)	Through put (KIPS)	Latency (ns)	Energy (fj)
2-phase 4 resources	184934	415.9	577.1	1299.4	1760.59
2-phase 2 resources	183412	536.8	447.1	1678.0	1834.65
4-p Fully 4 resources	193396	467.4	513.5	1460.5	1547.82
4-p Fully 2 resources	191374	617.9	388.4	1931.3	1686.65
4-p Long 4 resources	194586	461.3	520.3	1441.2	1487.89
4-p Long 2 resources	192512	611.1	392.8	1910.2	1626.53

Table 1 shows the performance of the resource shared four-phase fully decoupled and long hold Stump processors.

Circuit	Energy
2-phase 4-resource	1760.59
4-p fully 4-resource	1547.82
4-p long 4-resource	1487.89
2-phase 2-resource	1834.65
4-p fully 2-resource	1686.65
4-p long 2-resource	1626.53

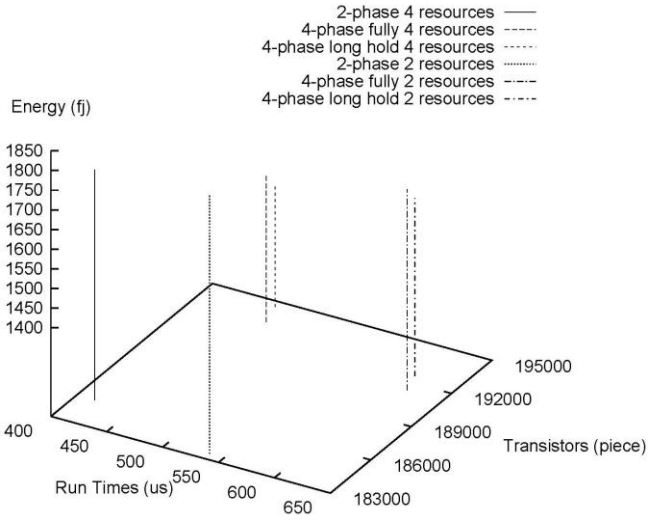


Fig. 14. The performance of the shared resource Stump processors

Resource sharing was described in Section 4. The two-phase, four-phase fully-decoupled and four-phase long-hold Stumps [13] were used to test this technique. The performance of the resource shared circuits are shown in Table 1.

As shown in Figures 14 and 15 the power consumption is high and the run time is increased by 50 percent compared to the original designs. This is just a demonstration of how the method works. If man expensive components are required in a circuit this method can be applied to reduce costs. It is clear that the two-phase control circuit is still fast.

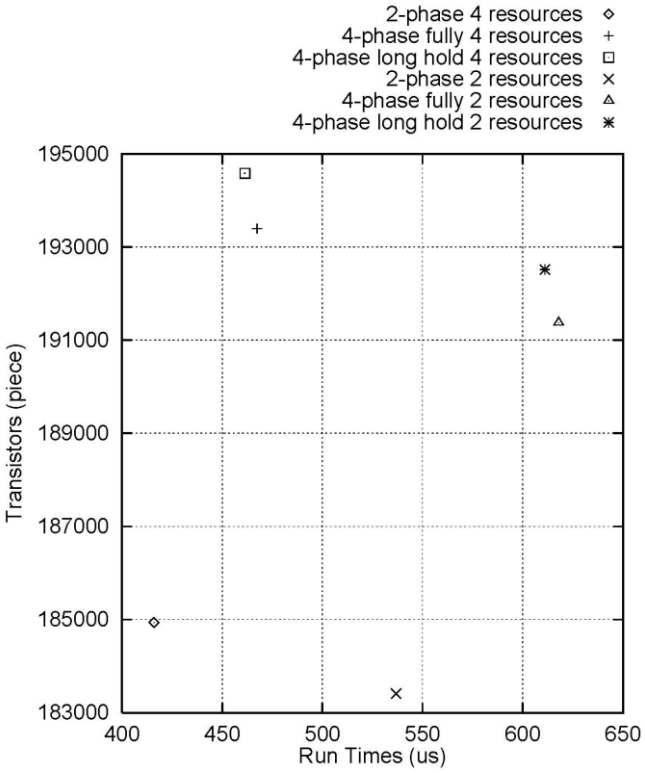


Fig. 15. The performance of the shared resource Stump processors

Asymmetric delay

Figure 16 shows an asymmetric delay. The asymmetric delay was also applied to improved the performance of the circuits. The performance of the synthesized Stump [13] using the asymmetric delay is shown in Table 2.

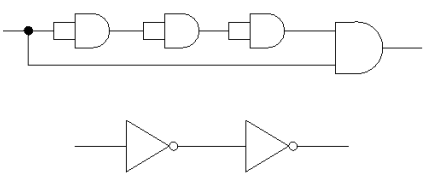


Fig. 16. Delay models

Circuit	4-p simple	4-p semi	4-p fully	4-p long
Energy	1332.53	1520.86	1109.53	1059.34

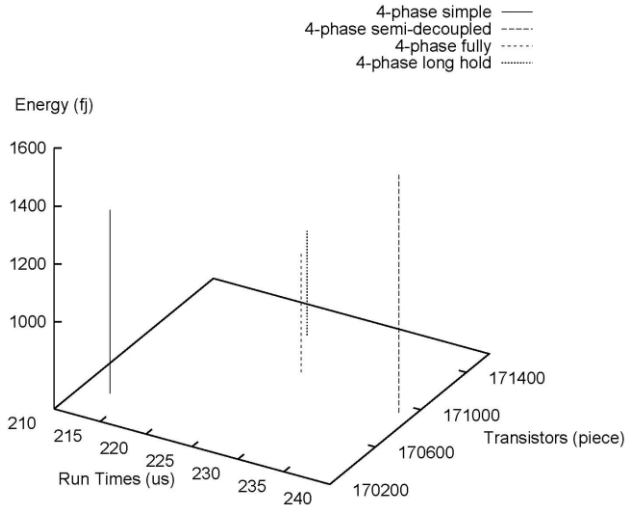


Fig. 17. The performance of the Stump processors using asymmetric delay

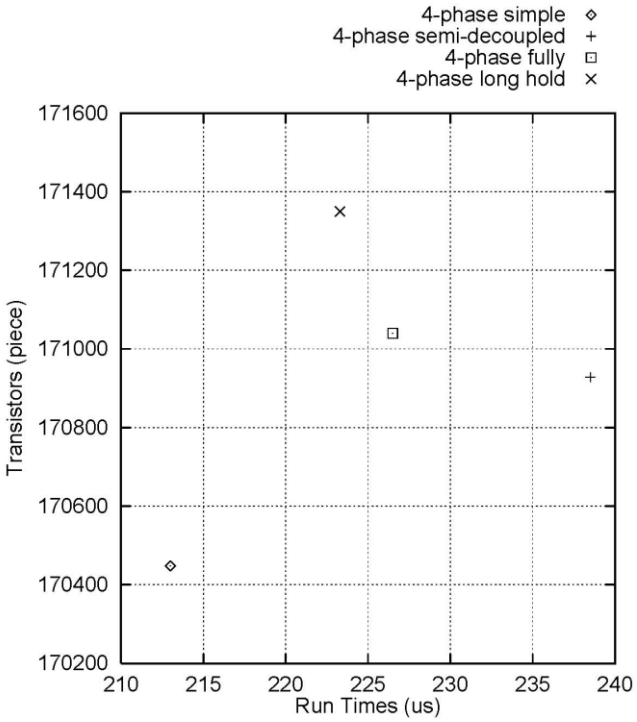


Fig. 18. The performance of the Stump processors using asymmetric delay

Table 2 The performance of the Stump processors with asymmetric delay

Circuit Name	Transistors (piece)	Run Time (μs)	Throughput (KIPS)	Latency (ns)	Energy (fj)
4-p Simple	170448	213.0	1127.0	662.4	1332.53
4-p Semi	170928	238.5	1006.4	740.8	1520.86
4-p Fully	171040	226.5	1059.6	670.2	1109.53
4-p Long	171350	223.3	1074.7	658.4	1059.34

As shown in Figures 17 and 18 the Stump processor using the four-phase simple control circuit is 40 percent faster than the original synthesized circuit. However, it is necessary to deal with asymmetric delays very carefully. If the asymmetric delay is applied it is necessary that no transition can arrive on the input within double the delay time.

The reason is that some unwanted transitions appear on the control signal after the **AND** operation. The four-phase fully decoupled and long hold control circuits can easily meet such problems. The time between the rising transition and the falling transition of the control signal is about 70 ns. If the data path delay is bigger and the asymmetric delay is used the circuit may go wrong.

6 Analysis

Relative to the original synthesized Stump processors, optimization reduces the transistor counts by 19.5 % ~ 21.6 %. The power saved is about 66.1 % ~ 69.8 %. The run time of the two-phase and the simple control Stump processors have an improvement of 29.5 % and 36.6 %. The throughput of the two-phase and the simple control Stump processors have an improvement of 41.9 % and 57.6 %. The latency of the two-phase and the simple control Stump processors is reduced by 29.5 % and 34.6 %.

Figures 19, 20, 21, 22, and 23 show the performance of different synthesized circuits [7][11][13]. They show that the two-phase circuits have good performance.

Figures 24, 25, 26, 27, and 28 show the comparison of the performance of two-phase circuits and the best performance of the four-phase circuits. It is very clear that the two-phase micropipeline circuits offer better performance than the four-phase designs. The author would like to further study techniques for the optimization of the four-phase control circuits.

The experimental results show that the two-phase circuits have good speed performance. This is due to the rising and falling transitions of the 4-phase circuits following the same routes. To improve performance asymmetric delays can be built using the circuit shown in Figure 29.

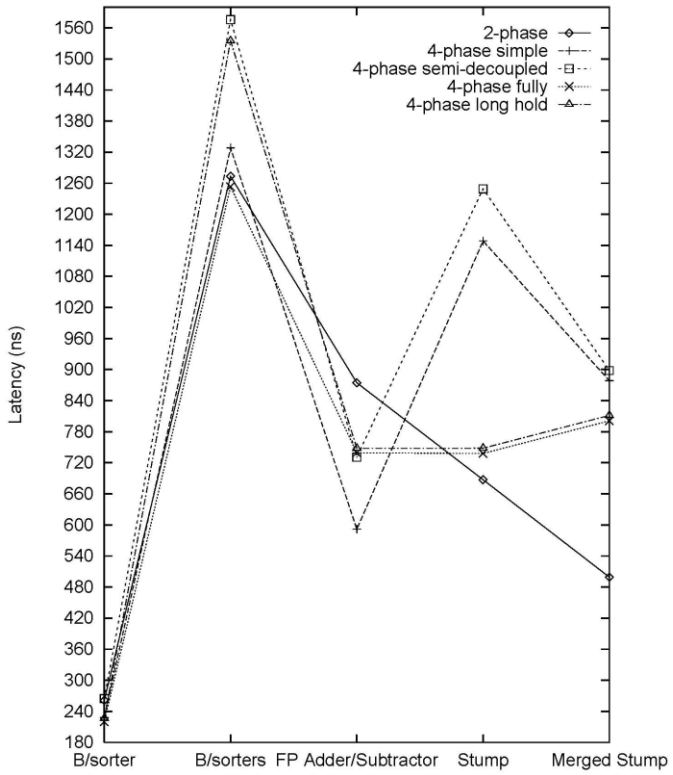


Fig. 19. The latencies of different control circuits

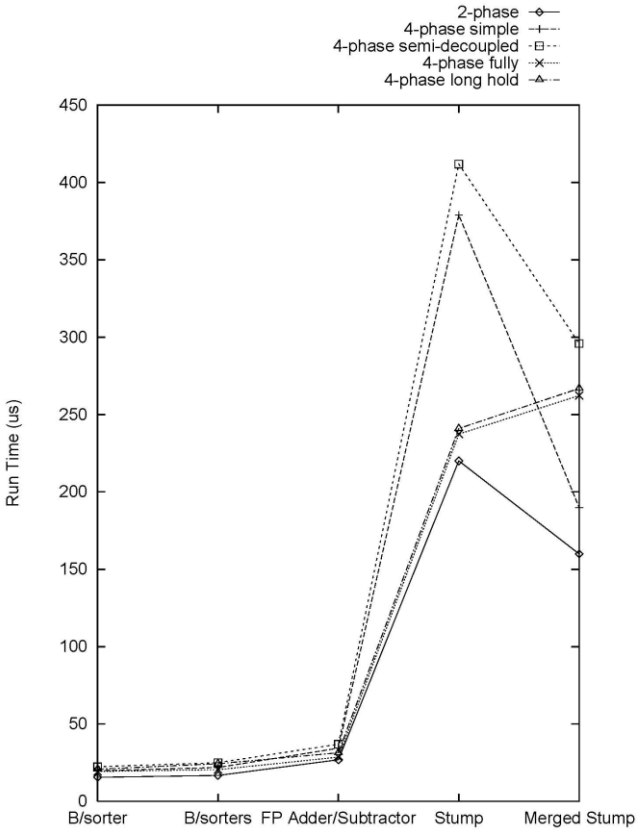


Fig. 21. The run times of different control circuits

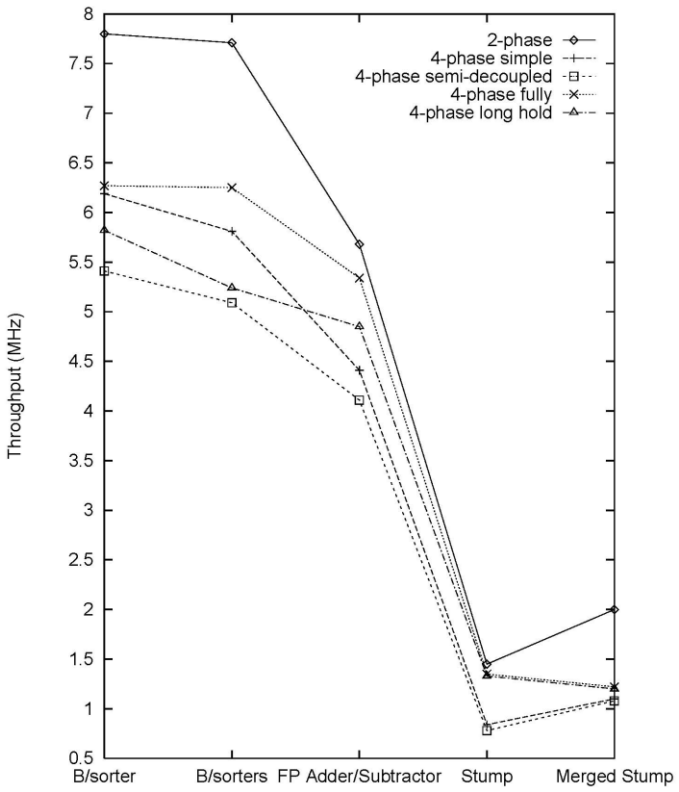


Fig. 20. The throughputs of different control circuits

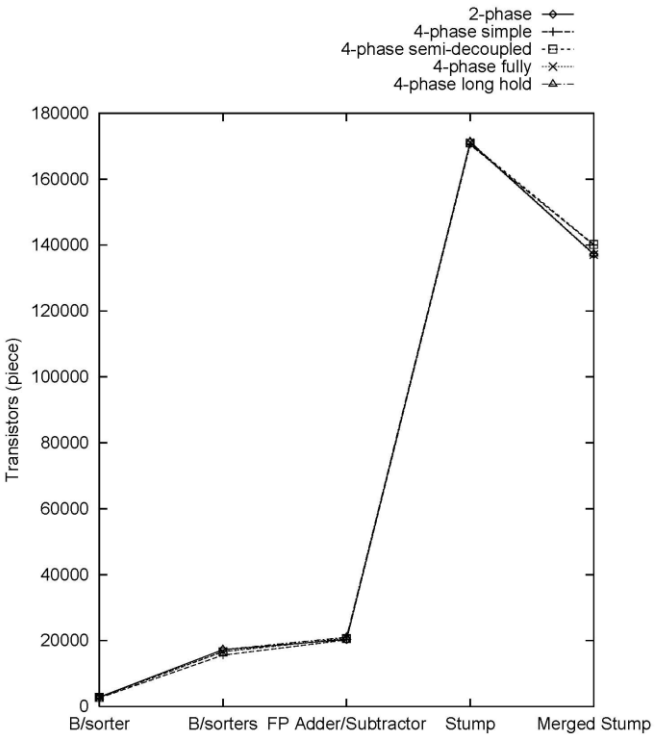


Fig. 22. The transistors of different control circuits

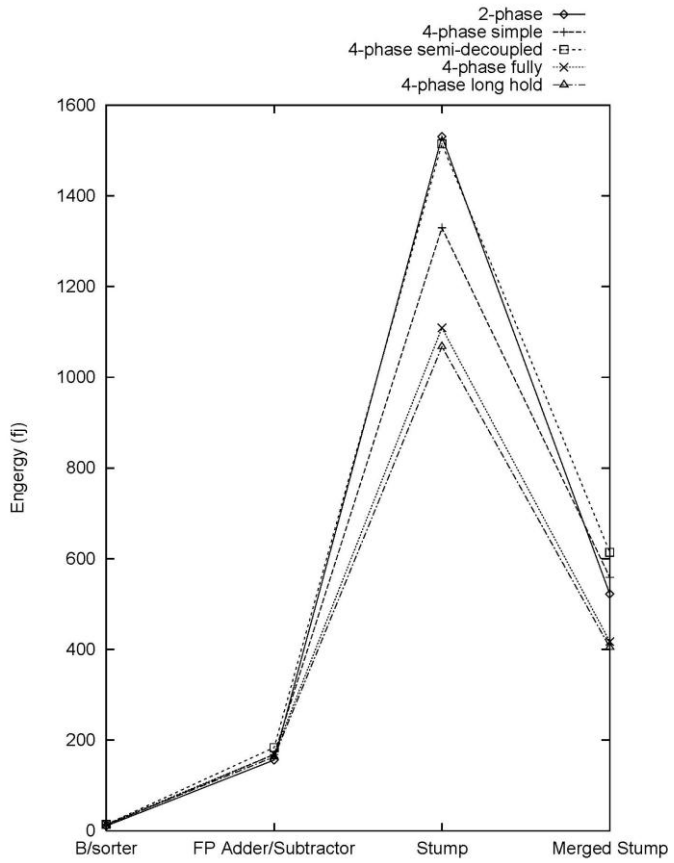


Fig. 23. The energy of different control circuits

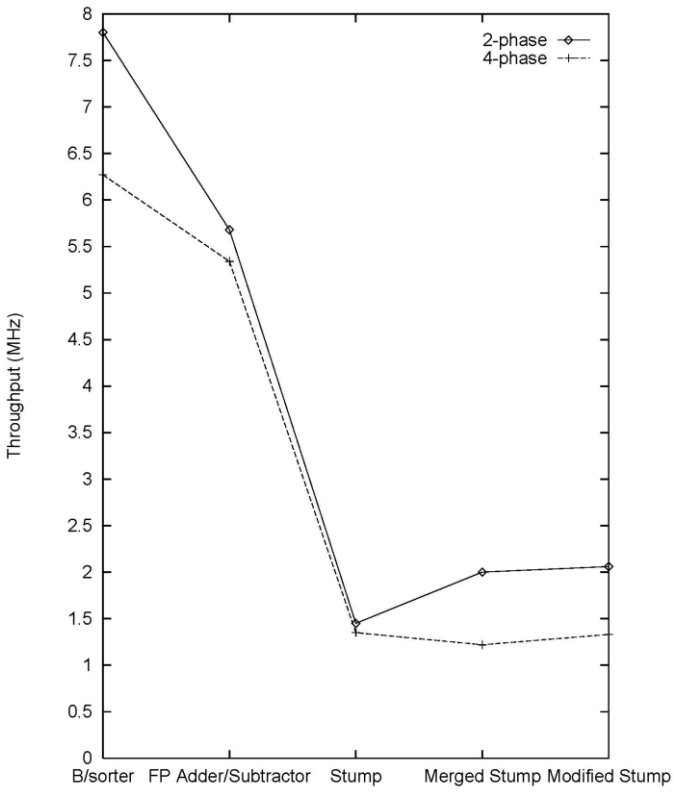


Fig. 25. The throughputs of different control circuits

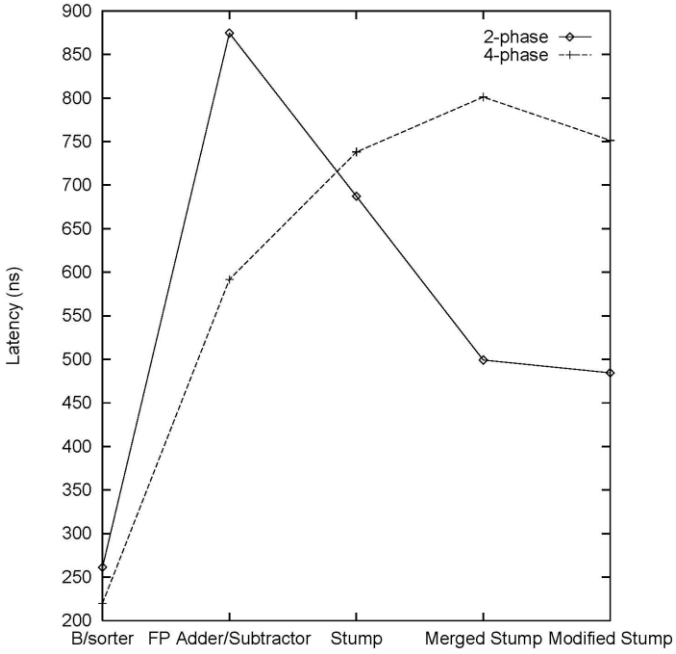


Fig. 24. The latencies of different control circuits

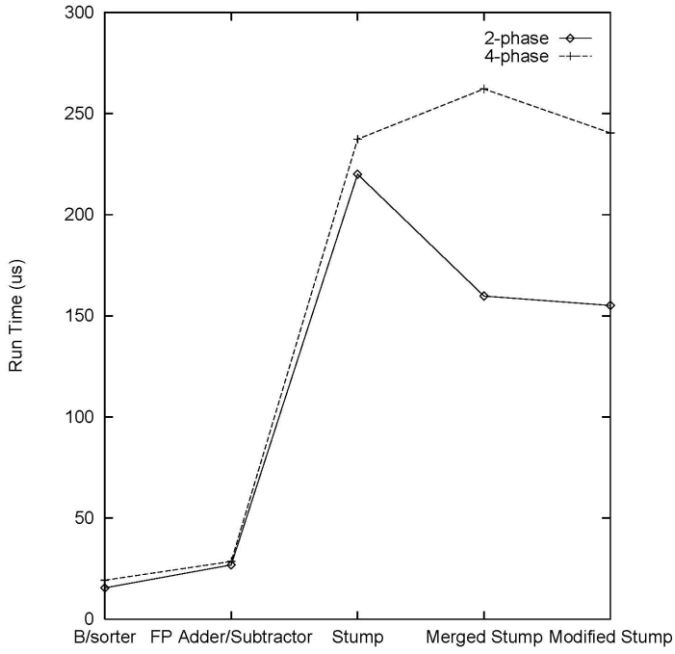


Fig. 26. The run times of different control circuits

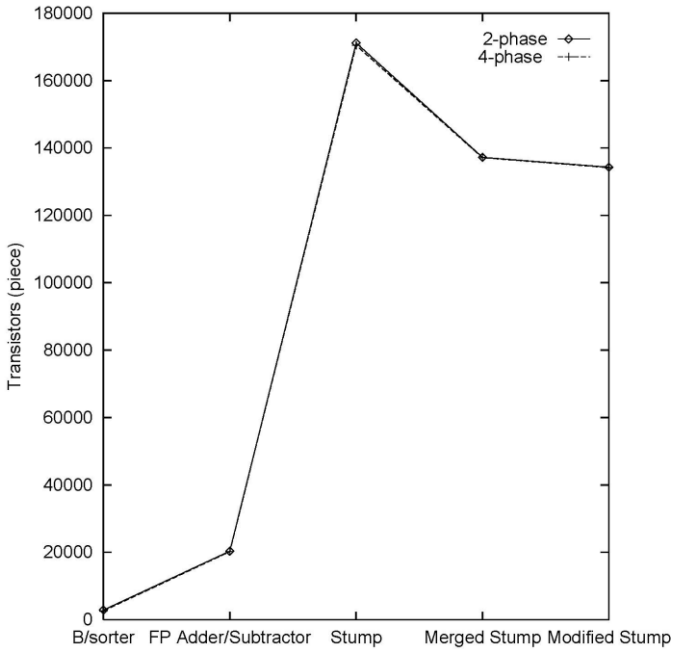


Fig. 27. The transistors of different control circuits

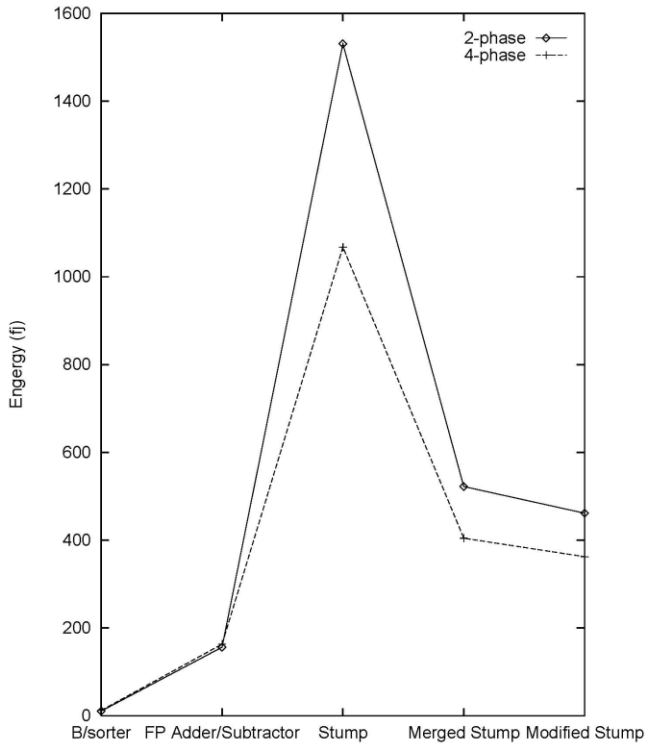


Fig. 28. The energy of different control circuits

Only one instruction flows through the pipeline of the Stump processors [13]. This is why the performance of the experimental results was only 1.55 MIPS. However, this meets the requirement of the behavioural description. Further investigation is required to ensure that multiple instructions are able to flow through the synthesized Stump pipelines.

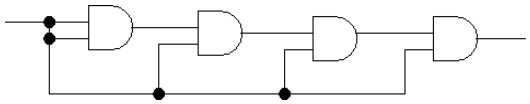


Fig. 29. The asymmetric delay with faster reset

The 4-stage circuit using the four-phase simple and semi-decoupled latch control shown in Figure 30 required an **Arbiter** at the input to the top **Call** module if the **Rout+** of **ss16** is sent out before **d1** becomes logical **'0'** and the **Call** module shown in Figure 31 is applied. Alternatively, deadlock is avoided if the **Call** module [12] shown in Figure 32 is applied.

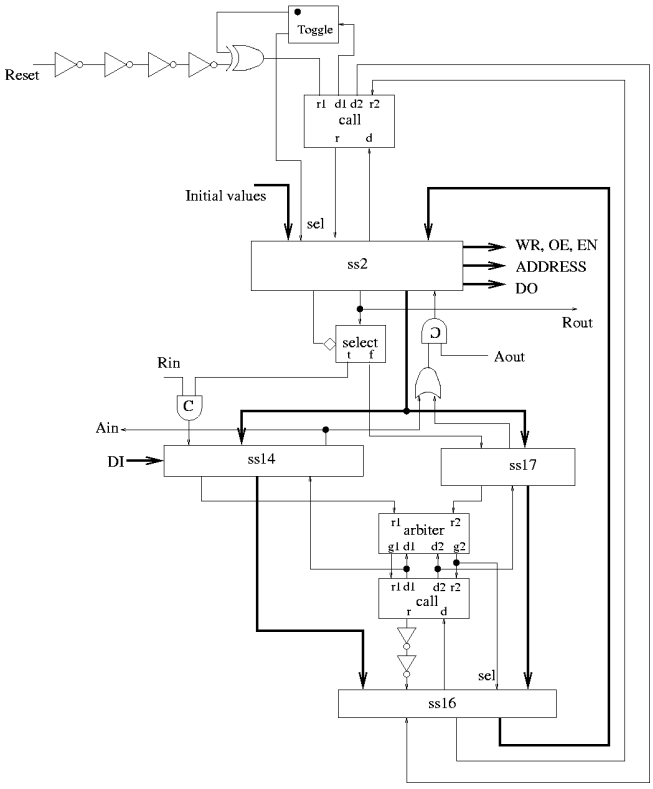


Fig. 30. A four-phase Stump processor

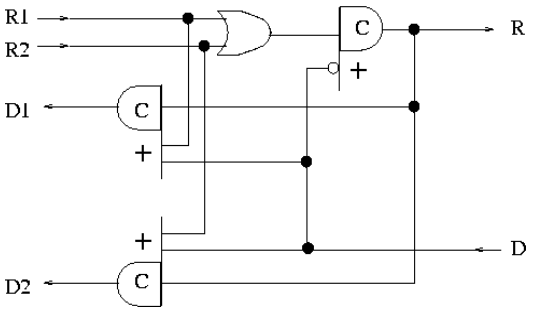


Fig. 31. A four-phase call circuit

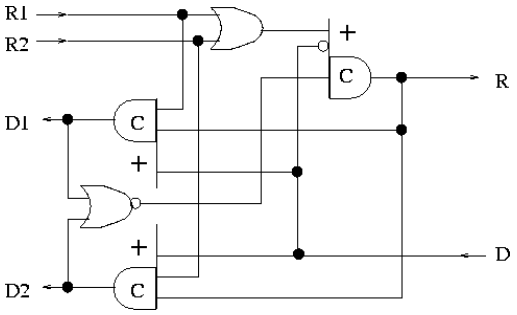


Fig. 32. Another four-phase **Call** module [12]

7 Conclusion

A comparison of two-phase and four-phase micropipeline circuits in VHDL was presented. From the simulations of the various configurations some insight is obtained. It is summarized as follows:

- Two-phase circuits have good performance on speed. When a concurrent reset is not applied on the four-phase circuits and the rising and falling transitions follow the same routes as the two-phase circuits, it is more complex to build the circuits to ensure both the rising and falling transitions can flow through the control path correctly. On the other hand, asymmetric C-gates are useful for building circuits for the selection signals of the multiplexers.
- Four-phase circuits have better performance on power consumption. Four-phase circuits using the long-hold latch control circuit are best.
- When the stages are connected in a feed back loop the number of stages can be 2 for the two-phase circuits and the four-phase circuits using the fully decoupled and long hold latch control circuits. However, the minimum number of stages is 3 for the four-phase circuits using the simple and semi-decoupled latch control circuits.

This paper has presented some engineering work on developing a technique for the construction of micropipeline circuits with sharing resources. The experimental results show that the fastest speed is the synthesized circuit with 2-phase control circuits. The lowest power consumption is the synthesized circuit with the long hold 4-phase latch control circuits. The synthesized circuit with 2-phase control circuits has the lowest the transistor count. The synthesized circuit using the 2-phase control circuit has high throughput as well as low latency.

In resource sharing the 2-phase implementations have better performance than the four-phase implementations. There is no return to zero problem. fork and join cost nothing to the two-phase implementations. Add some addition buffer stages the four-phase implementations using the fully decoupled and long hold latch control circuits can also implement resource sharing. However, the four-phase implementations using the simple and semi-decoupled latch control circuits require more buffer stages to avoid deadlock.

References:

[1] Tan, S.-Y., Furber, S.B., Yen, W.-F., "The Design of an Asynchronous VHDL Synthesizer", *Proceedings of the Design, Automation and Test in Europe Conference 1998 (DATE98),* Paris, Feb. 1998, pp. 44-51.

[2] Furber, S.B., and and Liu, J., "Dynamic Logic in Four-Phase Micropipelines", *Async'96*, Aizu-Wakamatsu, Japan, Mar 18-21 1996.

[3] Furber, S.B., Day, P., "Four-Phase Micropipeline Latch Control Circuits", *IEEE Trans. on VLSI Systems*, vol. 4 no. 2, Jun. 1996 pp. 247-253.

[4] Sacker, M., Brown, A.D., Rushton, A.J., Wilson, P.R., "A Behavioral Synthesis System for Asynchronous Circuits", *IEEE Trans. on VLSI Systems*, vol. 12 no. 9, Sep. 2004, pp. 978-994.

[5] Furber, S.B., "Computing without Clocks: Micropipelining the ARM Processor", in "Asynchronous Digital Circuit Design" edited by G. Birtwistle and A. Davis, Springer Verlag, pp.211-262.

[6] Sutherland, I. E., "Micropipelines", The 1988 Turing Award Lecture, *Communications of the ACM*, Vol. 32, No. 6, January 1989, pp. 720-738.

[7] S.-Y. Tan, W.-T. Huang, "The Design of an Asynchronous Blocksorter", *Proceedings of the 12th International Conference on Networking, VLSI and Signal Processing (ICNVS '10) (WSEAS Cooperating Conference)*, University of Cambridge, UK, 20-22 February 2010, pp. 73-78.

[8] J. Carlsson, K. Palmkvist, and L. Wanhammar, "Synchronous Design Flow for Globally Asynchronous Locally Synchronous Systems", *Proceedings of the 10th WSEAS International Conference on CIRCUITS*, Vouliagmeni, Athens, Greece, July 10-12, 2006, pp. 64-69.

[9]A. N. Ismailoglu, M. Askar, "Verification of Delay Insensitivity in Bit-Level Pipelined Dual-Rail Threshold Logic Adders", *7th WSEAS Int. Conf. on Electronics, Hardware, Wireless and Optical Communications*, Cambridge, UK, February 20-22, 2008

[10]A.Vasilescu, "Algebraic model for the intercommunicating hardware components behaviour", *12th WSEAS International Conference on COMPUTERS*, Heraklion, Greece, July 23-25, 2008, pp. 241-246.

[11] S.-Y. Tan, W.-T. Huang, "A VHDL-based design methodology for asynchronous circuits", *WSEAS TRANSACTIONS on CIRCUITS and SYSTEMS*, Vol. 9, Issue 5, May 2010, pp. 315-324.

[12] J. Liu, "Arithmetic and Control Components for an Asynchronous System", PhD Thesis, Dept. of Computer Science, Univ. of Manchester, 1997.

[13] S.-Y. Tan and W.-T. Huang, The Design of a simple asynchronous processor, *Proceedings of the 12th WSEAS International Conference on MATHEMATICAL METHODS AND COMPUTATIONAL TECHNIQUES IN ELECTRICAL ENGINEERING (MMACTEE '10)*, Timisoara, Romania, October 21-23, 2010, pp. 165-170.

[14] S.-Y. Tan and W.-T. Huang, The Design of sharing resources for asynchronous systems, *Proceedings of the 12th WSEAS International Conference on MATHEMATICAL METHODS AND COMPUTATIONAL TECHNIQUES IN ELECTRICAL ENGINEERING (MMACTEE '10)*, Timisoara, Romania, October 21-23, 2010, pp. 171-176.