The design of sharing resources for 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.

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 had been published [1][7][11]. 2-phase and 4-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 micropipelines and introduces 2-phase and 4-phase event-driven Logic modules. The Stump processor design will be presented in Section 4. Section 5 will present experimental results. Finally, Section 6 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-. Fig. 1 illustrates two kinds of four phase signalling,
the ‘early’ mode and the ‘broad’ mode [2]. The ‘early’ mode (Fig. 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 (Fig. 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.

Various event-driven logic modules for controlling transition signals are shown in Fig. 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.

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

Fig. 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 Fig. 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 Fig. 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.

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

Two sets of multiplexer circuits as shown in Fig. 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. 5 The Arbiters, Calls and Muxs are connected to control the resource stage receiving requests from different stages.

Fig. 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, s4 of the latch shown in Fig. 6 are generated from the circuits shown in Fig. 5. The outputs labelled b1, b2, b3, b4 of the latch shown in Fig. 6 are used to decide where the events should be sent to. They are connected to the inputs labelled b1, b2, b3, 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 Fig. 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 1. 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 MUXs may need to be connected to logic ‘0’ or ‘1’ depending on what the computations are. The circuit in Fig. 4 may be denoted by Fig. 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.

However, Fig. 8(a) has deadlock if a 4-phase protocol is used to implement the circuit. The 4-phase circuit of Fig. 8(b) can run correctly without deadlock. The reason is that the rising transition of the stg3 stage is sent to the stg4 stage, the stg31 stage and the stg35 stage at the same time. The acknowledge rising transitions of the stg31 stage and the stg35 stage are sent out when these two stages hold data. However, the stg4 stage is waiting for the rising transitions from the stg32 stage and the stg36 stage. If the stg31 stage first calls the resource stage, stg38 the Rin of stg38 will not return to zero due to the stg31 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 Fig. 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 stg45 and the new stage. Remember the destination stage from the stg5 is the stg6 stage. After processing the circuit of Fig. 9 is changed as shown in Fig. 10.

The circuit of Fig. 8(a) can use the above method to avoid deadlock. However, the circuit of Fig. 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 Fig. 11. Now the circuit shown in Fig. 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 a 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 Fig. 12. 47 and 48 are resource stages. They are a 17-bit adder and a 17-bit subtractor. The circuit shown in Fig. 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.

**Fig. 8** The stages (\texttt{stg31} and \texttt{stg35}) 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).

The reason is that the falling transitions of \texttt{Rout} in the fully decoupled and the long hold control circuits can be sent out before the falling transition arrives on \texttt{Rin}. Therefore, the falling transitions of \texttt{Aout} in the fully decoupled and the long hold control circuits can arrive before the falling transition arrives on \texttt{Rin}. However, it is necessary to ensure that the falling transitions of \texttt{Rout} in the simple and semi-decoupled control circuits are sent out after the falling transition arrives on \texttt{Rin}. Deadlocks may happen. For example, as shown in Fig. 12, \texttt{stg11} sends rising transitions to \texttt{stg26}, \texttt{stg28}, \texttt{stg30}, \texttt{stg32}, \texttt{stg34}, \texttt{stg36}, \texttt{stg38} and \texttt{stg40}. Then \texttt{stg26} sends the request transition to \texttt{stg47}. After \texttt{stg47} holds data it sends a rising transition to \texttt{stg27}. \texttt{stg27} is one of eight stages which connect to \texttt{stg12}. It is necessary to wait for these eight stages to send requests to \texttt{stg12}. Then \texttt{stg12} will send the acknowledge to these eight stages. However, four stages will still wait to send requests to \texttt{stg47}. On the other hand, \texttt{stg29}, \texttt{stg35}, \texttt{stg37}, \texttt{stg39} are still waiting for data from \texttt{stg47}. It is then impossible to get the rising transition from \texttt{stg12}. As shown in Fig. 13, \texttt{Rout} of \texttt{stg27} stays at logical ‘1’ and the logical ‘0’ on \texttt{Rin} of \texttt{stg27} cannot pass the C-gate without a rising transition on \texttt{Aout} of \texttt{stg27}.
Fig. 12 Resource stages (47 and 48) are called from different stages without deadlock.

Fig. 13 Four stages of the simple control circuits

Eight extra buffer stages are required to connect from stg27, stg29, stg31, stg33, stg35, stg37, stg39 and stg41 to stg12 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.

5 Conclusion
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.

References: