Designing of FIFO for the High Speed Memory Access

Dr. S. Muthukrishnan¹ S. Janani² S. Saravana Kumar³

¹Professor and Head of the Department ²PG Scholar (VLSI Design) ³Senior Design Engineer (VLSI Design)

1, 2 Sri Eshwar College of Engineering, Coimbatore, Tamilnadu, India ³Caliber Embedded Technologies, Coimbatore

Abstract— High speed is one of the most required feature for the up to date electronic systems designed for its high performance and handy applications. First-In First-Out memories (FIFOs) have advanced from objectively simple logic functions to high-speed buffers combine large blocks of Static Random Access Memory. FIFOs are often used to sensibly pass data from one clock control to another asynchronous clock control. Using a FIFO to pass data from one clock domain to another clock domain needs multi-asynchronous clock design techniques. This impression will detail one method that is used to design, synthesize and analyse a safe FIFO connecting different clock domains using Memory counter that are synchronized into a different clock domain before testing for "FIFO full" or "FIFO empty" conditions and Address Calculator that are used to read/write data in user preferred address. The demonstratively evaluated, synthesized counter and address calculator can be included in FIFO.

Key words: FIFO, Asynchronous FIFO, Memory Counter, Address Calculator

I. INTRODUCTION

Memory is the fundamental unit for storage resolution, where we require to customize the FIFO for reduced propagation delay and to increase efficiency. The data storage in FIFO will be in SRAM, Flip-flops or Latches. The SRAM acts as primary cache, and is based on the application. In every digital components, there exit the exchange of data between Printed Circuit Boards (PCB). Intermediate storage or buffering always is crucial when data attain at the receiving PCB at a high speed, but are processed gradually or unevenly. Buffers here analysed with our everyday life. This tailback works with the standard of first approach, earliest served.

All the kinds of FIFOs described in FIFO types can be implement in different hardware architectures. The creation of conventional FIFOs has constantly been developed. primarily, FIFOs worked by the fall-through principle. Nowadays, FIFOs are almost always based on an SRAM, which twisted a considerable increase in the number of data words stored, despite the faster speed. All probable hardware architectures also are found in software FIFOs. Buffers acts as the interface between components that work at different speeds or unevenly. In electronic devices, speed determines the data rate, the data rate of the A/D converter is inhibited by a quartz crystal. The diverse data rates are waged by buffering. A FIFO is a form of buffer, where the data which is written into a buffer, first comes out. There are other kind of buffers like the LIFO (Last In First Out), often called a stack memory, and the mutual memory. The buffer can be chosen base on the application. The choice amonga software and a hardware solution depends on the application and the features desired.

Every memory in which the data word that is written in first also comes out first when the memory is read is a first-in first-out memory. Figure.1. show the common flow of input and output in FIFO. There are three kinds FIFO which can be used according to the applications and their needs. Three varieties of FIFO are Shift record, Exclusive read/write FIFO, simultaneous read/write FIFO. The shift register is not usually referred to as a FIFO, although it is first-in first-out in character. Consequently, this application report focuses exclusively on FIFOs that handgrip variable-length data.

Fig.1. First In First out Flow

There are three varieties FIFO which can be used according to the applications and their needs. Three kind of FIFO are Shift record, Exclusive read/write FIFO, Concurrent read/write FIFO.

- **Shift register** – FIFO with an invariable integer of stored data words and, thus, the necessary synchronism between the read and the write operations because a data word must be read every time one is written.

- **Exclusive read/write FIFO** – FIFO with a variable number of stored data words and, because of the internal structure, the required synchronism between the read and the write operations.

- **Concurrent read/write FIFO** – FIFO with a variable number of stored data words and possible asynchronous between the interpret and the write operation.

The shift register is not usually referred to as a FIFO, even although it is first-in first-out in character. Consequently, this application report focuses exclusively on FIFOs that handle variable-length data.

There are timing relationships between the write clock and the read clock. For occurrence, overlapping of the read and the write clocks could be prohibited. To permit use of such FIFOs between two system that vocation asynchronously to one another, an external circuit is required for synchronization. But this harmonization circuit usually considerably reduces the data rate. In concurrent read/write FIFOs, there is no confidence between the writing and reading of data. Immediate writing and reading are possible in overlapping fashion or successively. This means that two system with diverse frequencies can be connected.
Designing of FIFO for the High Speed Memory Access

(IJSRD/Vol. 3/Issue 02/2015/321)

All rights reserved by www.ijsrd.com

Fig. 2: FIFO Pointer

Concurrent read/write FIFOs, which force into two group:
- Synchronous FIFOs
- Asynchronous FIFOs

II. ASYNCHRONOUS FIFO POINTERS

In order to know FIFO design, one desires to understand in what way the FIFO pointers job. There are mainly two pointers.
- Write Pointer
- Read Pointer

The write pointer continuously points to the next word to be written; for that motive, on retime, both pointers are rest to zero, which also occurs to be the next FIFO word location to be written. Figure 2 gives the write and read operations using the write and read pointers, on a FIFO write operation, the recollection location that is piecing to by the write pointer is printed, and then the write pointer is incremented to indicate to the following location to be written.

In that order, the read pointer always points to the present FIFO word to be read. Again on reorganize, both pointers are locate to zero, the FIFO is empty and the read pointer is indicating to invalid data (because the FIFO is unfilled and the empty flag is affirmed). As quickly as the first data word is printed to the FIFO, the write pointer increments, the empty flag is blank, and the read pointer that is still address the filling of the first FIFO memory word, immediately drives that first applicable word onto the FIFO data production port, to be read by the receiver sense.

The truth that the read pointer is always pointing to the next FIFO word to be read means that the receiver logic does not have to use two clock periods to read the data word. If the receiver first had to rise the read pointerprevious to reading a FIFO data word, the receiver would clock formerly to output the data express from the FIFO, and clock a additional time to capture the data word into the receiver. That would be uselesslyincompetent.

Figure 3 shows that the control lines WRITE CLOCK and FULL are used to write data. When a data phrase is to be written into an asynchronous FIFO, it is earliest required to ensure whether there is space obtainable in the FIFO. This is done by inquiring the FULL status line. If free space is point to, the data phrase is applied to the data inputs and written into the FIFO by a clock edge on the WRITE CLOCK input.

Thus, synchronous FIFOs are unified easily into ordinary processor architectures, offering complete synchronism of the FULL and EMPTY status signals with the particular free-running clock.

The need for huge data transfer from/to mass storage in multiprocessing and data communication fields has become more perilous in recent years. Data transfer hurry is becoming a bottleneck in these systems and the necessity for large data buffers is widely predictable. Mainframe systems have started to implement expanded storage or semiconductor disks within the system to improve
its speed recital. Since the system simplicity is an essential factor in the expanded storage, FIFO memory is thought to be one of the best configurations. Thus it attain the synchronization.

In a digital signal-processor (DSP) system, there is often the need for high-speed data acquisition. Connecting fast data converters to a DSP can be a demanding task. For example, if a 200-MHz DSP must service a 50-MHz analog-to-digital converter (ADC), the DSP must read one value every four clock cycles. This high data rate cannot be achieved if interrupts are used to read data from the converter. Interrupt latencies prevent the DSP from reacting fast enough.

Only a shortest memory access (DMA) controller can handle such data rates. But a fast data converter, connected directly to the DSP, ties up the whole I/O bandwidth of the DSP. A FIFO is an ideal solution to cushion some of the data. When a whole block of data has been sampled, the data can be transferred in a read rupture from the FIFO to the DSP.

IV. CIRCULAR FIFO
To oath the drawback of a initial FIFOs, the manner should no longer shift the data words through all memory places. The trouble is resolved by a circular memory with two pointers. In a circular FIFO commencement, the memory address of the received data is in the write pointer. The address of the initial data word in the FIFO that is to be read out is in the read pointer. After reset, both pointers stipulate the same memory place.

After each write process, the write pointer is set to the next memory location. The investigation of a data word sets the read pointer to the following data word that is to be read out. The read pointer regularly follows the write indicator. When the read pointer ranges the write indicator, the FIFO is empty. If the write pointer hooks up with the read pointer, the FIFO is full. Figure 5, prove the principle of a circular FIFO with two pointers.

![Circular FIFO with two pointers](image)

**Fig. 5:** Circular FIFO with two pointers

The FIFO mean in this concept guarantees that the empty flag will be shaped in the read-clock domain to safeguard that the empty flag is noticed directly when the FIFO buffer is empty and likewise full flag will be shaped in the write-clock domain to insure that the full flag is detected immediately when the FIFO buffer is full. Now, in this FIFO design a spot counter will be take care of FIFO full and empty settings. This status counter will be incremented on each write and will be decremented on every read operation. When the status counter ranges the most FIFO depth it will declare FIFO full signal and when its value is zero it will declare FIFO empty signal. This status counter also assurance that there is no loss of data i.e. it ensures the “write before read conditions” and “read sooner than wire state”. On retune the status counter is fixed to zero value and empty signal will be asserted.

The write allow and read enable will be conditional on write request and read request and empty and full bit.

Write enable == write req. and ~ (not of) full
Read enable == read req. and ~ (not of) empty

V. MEMORY COUNTER
To augment the performance and speed of FIFO, counter can be inbounded effectively using the following counter structure, shown in Figure 6. Memory is the process in which in sequence is determined, put in storage, and recovered. In computing, memory denotes to the corporal devices used to store programs or data on a temporary or permanent basis for use in computer or digital electronic devices. Memory is an significant category of Smart Property. Circuits be analog design that must be prudently designed. Separately from storage, counter is a device which stores the number of times a detailed event or process has occurred, often in connection to a clock signal.

It checks whether the queue is unfilled, occupied, increment or decrement. Counter is of many categories such as binary/non-binary and asynchronous categories. There is propagation delay when data is transmit from one device to one more due to capacitance of the device. This delay can be condensed by Decade counter, Modulus–six counter etc., Additional to reduce the delay transmission from one domain to added in memory, memory answer structure is used.

![Block diagram of Memory Counter](image)

**Fig. 6:** Block diagram of Memory Counter

Memory counter clarify the solution for high speed counter. The common draw near is to add clock and return options for the normal operation of the counter. The indicator in counter can be incremented when the data is written in stack of memory and it get decremented when data is read from stack. The counter register is mould by
adding D flip-flop of 8-bit, it hinders passage of 8-bit data into one of the comparators. There are two comparators, one has a repeated value of “zero” and the other has constant value of “log(width)”, these constant values are associated with the data values from counter register. It afterwards intimate the Full and Empty circumstances of array of address. When the comparator stretches the output of “Empty”, automatically Read/Write(controller) bit from controller reach the multiplexer. Based on this input, counter acquire incremented, thus the writing operation exists. In differing, counter gets decremented throughout read operation. NAND gate becomes the input from memory array. This capitals that the counter described in this area is used to make the memory access very quickly and it helps to reduce the delay propagation of data.

The purpose of the answer block is to count the number of elements stored in the queue and to keep path of change. It may be reset when necessary.

<table>
<thead>
<tr>
<th>Element</th>
<th>Functional description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counter register</td>
<td>Register Size : LOG[Width]. Counts number of elements in queue</td>
</tr>
<tr>
<td>Small Alu</td>
<td>Add or subtract left foot from right foot , with the read/write ctrl.</td>
</tr>
<tr>
<td>Equalizers</td>
<td>1 if equal , 0 otherwise</td>
</tr>
</tbody>
</table>

Table 1: Counter: Sub-Blocks

VI. ADDRESS CALCULATOR – PIPELINED STRUCTURES

The ALU is a primary building block of the central processing unit of a computer, and even the simplest microprocessors encompass one for purposes such as maintain timers. The processors originate inside modern CPUs and Graphics Processing Units (GPUs) put up very powerful and very complex ALUs; a single component may contain a number of ALUs. An ALU must process numbers using the same formats as the rest of the digital circuit. The format of modern processors is roughly always the two’s complement binary number depiction. Early computers used a wide diversity of number systems, counting ones’ complement, two’s complement, sign-magnitude format, and even true decimal systems, with various representation of the digits.

The ones’ complement and two’s complement number systems sanction for subtraction to be skillful by adding the negative of a number in a very easy way which negates the need for specialized circuits to do subtraction; however, calculating the unconstructive in two’s complement requires adding a one to the low order bit and propagating the carry. Then an outside control unit tells the ALU what operation to perform on that data, and then the ALU stores its outcome into an output record. The control unit is responsible for moving the handle data between these registers, ALU and memory. Engineers can plan an arithmetic logic unit to calculate most operations. The added composite the operation, the more exclusive the ALU is, the more space it uses in the processor, and the more control it scatter. Therefore, engineers cooperation. They make the ALU imposing enough to make the processor fast, however not so complex as to become prohibitive.

The address calculator accomplishes the pointer to the top and the bottom of the queue including updating them when required.

ASSUMPTIONS: Handler will not try to write when Is_full is true (“1”) and will not try to read when Is_empty is true (“1”).

IMPORTANT: It is necessary to reset the unit to initialize its operation in order to initialize the address pointers exposed in Figure.7.

Table 2: Sub-Units of Address calculator

<table>
<thead>
<tr>
<th>Sub-Units</th>
<th>Functional description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU</td>
<td>Receives an input of 2 addresses of log[width] of head and tail, as well 3 bits for the Alu_ctrl. It outputs the next address as requested, plus an (Is_full) indicator.</td>
</tr>
<tr>
<td>ENCODER+STARTER</td>
<td>It receives a reset signal and it initializes the head address as 1 and tail address as 0, otherwise it outputs the Alu output to the requested register.</td>
</tr>
<tr>
<td>REGISTERS (head_addr and tail_addr)</td>
<td>The size of the registers is determined by LOG[Width] , and each keeps the address of the current head and tail addresses.</td>
</tr>
<tr>
<td>Mux 2 to 1</td>
<td>Receives as inputs the outputs of the two registers and outputs the requested address (which determined by the ctrl bit).</td>
</tr>
</tbody>
</table>

Fig. 7: Address Calculator
VII. RESULTS

Fig. 8: Simulation result of Memory Counter
Figure 8. shows the output of Memory Counter, whose memory will be count depending on the Full/Empty condition. If WRITE operation is performed, Counter gets Incremented while when READ operation is perform, it gets Decremented.

Fig. 9: Address Calculator
Figure 9. shows the output of Address Calculator, where the data can be written/read in any address. When the input is 0000, Reset is activated. When the input is 010 and 011, the data will be read from the desired address. When the input is 100 and 001, the data can be written and the address pointer can be increased.

VIII. CONCLUSION

Asynchronous FIFO design requires careful design and concentration from pointer creating techniques to full and empty generation. Finding errors typically requires simulation of a gate-level FIFO design. A confined FIFO can be interpret by utilising Memory-Counter and Address Calculator for the high speed memory access and efficiency can be augmented. Since the efficiency is increased, delay in READ/WRITE operation is decreased. They improves the ability to store the data in secure method and can be accessed frequently in very fast approach. The analyse and synthesized output are shown, simultaneously this can be new in FIFO.

In future work the Controller and Memory Array will be designed for the better efficiency of controlling the circuit through the progression of READ/WRITE data hooked on the memory. By using this Controller and Memory Array, the scope of FIFO custom can be enhanced in some areas such as disk scheduling algorithm in-order to determine the order of service disk input and output requests. In connections and network, bridge, routers and switches utilize FIFO memory to hold data packets that are in forward to a given destination. Future Electronics has a complete selection of FIFOs from quite a few manufacturers that can be used as an asynchronous FIFO recollection chip, FIFO director or for a FIFO IC.

REFERENCES