Multiple FIFO Configuration in ispLSI®6192

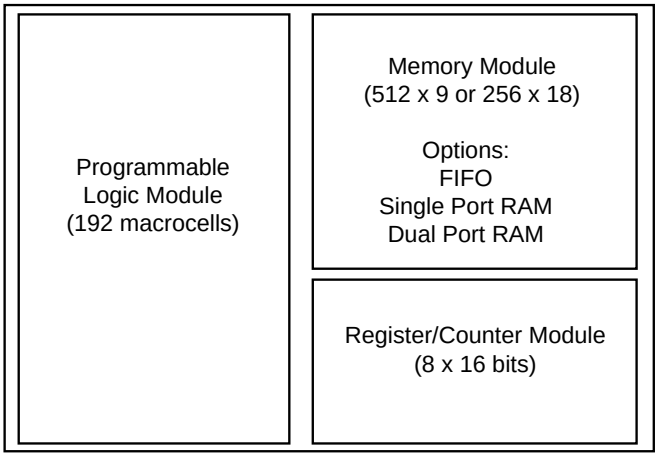
Introduction

In various data communications applications, it is often necessary to transmit and receive large blocks of data at high data rates between two systems. The size of the block can vary from several Kbytes to Mbytes depending on the application. In telecommunication systems, very often data consists of multiple low-speed channels multiplexed into a single high-speed data channel. Each channel may represent voice, data and/or video information from a single subscriber and 24 or more channels may be multiplexed into a single trunk line for long distance transmission. At the receiving end, the system must separate the single multiplexed data channel into multiple subscriber channels. In order to optimize system performance, often times it is desirable to buffer the data into a high-speed memory device, such as a RAM (Random Access Memory) or a FIFO (First In First Out). Individual channel data may be temporarily stored into a FIFO and retrieved by the host processor at appropriate times. This allows the processor to read a block of data at a time from each channel, thus lowering the overhead associated with memory read/write cycles. For a 24 channel system, it is desirable to have 24 FIFOs to buffer data for each data communication channel.

FIFOs are generally available as discrete memory devices. The control logic necessary to implement multiple FIFOs can be realized with PLDs, FPGAs or ASICs. Typically a system with multiple FIFOs requires logic to control full and empty flags from different FIFOs and logic to manipulate read and write signals to different FIFOs. If a system requires 24 FIFOs, the logic necessary to implement the control logic can be quite large. In addition, the designer must consider propagation delay of the control signals from the host processor through the PLD and to the individual FIFOs. Delay through the PLD, including on and off chip delays, must be well understood in order to meet setup and hold time requirements of the FIFOs. Added to the complexity is the requirement for today's high-speed data rates, which makes understanding and minimizing on and off chip delays through the PLD paramount.

Implementing memory functions, such as FIFOs, in general purpose PLDs is not the most efficient use of programmable logic real estate. A better solution is to use a PLD with dedicated memory functions, such as the Lattice ispLSI 6192. Figure 1 shows a high-level functional block diagram of the ispLSI 6192.

Figure 1. ispLSI 6192 Functional Block Diagram



The ispLSI 6192 contains general purpose programmable logic, a flexible 4000-bit memory block, and register/counter block. The memory block can be configured as a single-port RAM, a dual-port RAM, or a FIFO. As a single FIFO, total memory size can be configured as 512 X 9 bits or as 256 X 18 bits. The necessary control logic for a single FIFO is already built into the block.

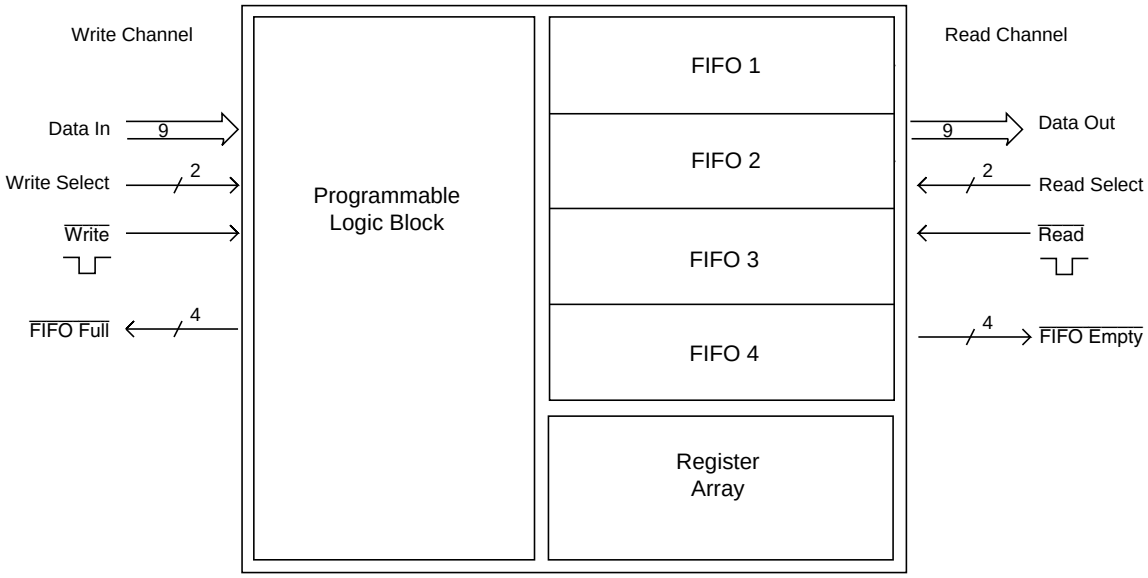
Flexible FIFO Configuration

In addition to the single FIFO, the ispLSI 6192 can also be configured as a bi-directional FIFO or as multiple FIFOs. As a bi-directional FIFO, the memory block is divided into two separate FIFOs, each FIFO being 256 X 9 bits. One FIFO is used for transmit and the other is used for receive functions. Since the FIFOs operate independently, simultaneous read and write operations can occur on the receive and transmit FIFOs. The control logic necessary to support the bi-directional function, such as address counters and FIFO flag logic and registers, can be implemented in the programmable logic block of the ispLSI 6192.

In some applications, it may be necessary to have multiple bi-directional FIFOs. The 4000-bit memory block can be configured as several independent bi-directional FIFOs. For example there can be two FIFOs for the transmit function and two FIFOs for the receive function, each FIFO being 128 X 9 bits. The control logic functionality will increase as you double the size of address counters, FIFO flag logic and registers.

This application note describes how the ispLSI 6192 can be configured into four uni-directional FIFOs. The dia-

Figure 2. Functional Diagram



gram shown in Figure 2 is for four independent FIFOs, each FIFO being 128 bytes deep X 9 bits wide. Read and write operations can occur simultaneously in this configuration, by simply sending the appropriate two-bit address to select one of the four target FIFOs along with the proper write and read signals. The FIFO control logic internal to the ispLSI 6192 generates the proper memory address for read and write operations, and sets the full and empty flags to be read by the external host system.

FIFO Operation

To write to the FIFO, the user (Write Channel) must provide the appropriate two-bit select, data, and a write pulse to the device. The two-bit address selects one of the four FIFOs to be written to. The select line and data bus must be valid while the write pulse is active. Data is written into the FIFO at the end of the write pulse, provided that the FIFO is not full.

To read from the FIFO, the user (Read Channel) must provide the appropriate two-bit select, and a read pulse to the device. The two-bit address selects one of the four FIFOs to be read from. The select line must be valid while the read pulse is active. Data is read from the FIFO at the end of the read pulse, provided that the FIFO is not empty. The data bus is always actively driven by the ispLSI 6192 and valid data is present on the bus at the end of the read cycle.

Internal to the programmable logic block, there are address registers that hold the address pointers for the next read and next write operations. For a given FIFO, the read and write address registers (each seven bits wide) are initialized to '0's upon power up. When data is written to the FIFO, the write register content is incremented to '1', setting the write address pointer for the next write operation to '1'. Subsequent write operations increment the address pointer such that the write register always holds the address of the next write operation.

Similarly, when data is read from the FIFO, the read register content is incremented to '1', setting the read address pointer for the next read operation to '1'. Subsequent read operations increment the address pointer such that the read register always holds the address of the next read operation.

When the write and read addresses become equal, the FIFO FULL flag is set when the last operation was a write; the FIFO EMPTY flag is set when the last operation was a read. When the FIFO is FULL, write operations are not allowed; write pulses are ignored internally to the logic block and are not passed on to the memory block. The write address pointer is not incremented. A read operation must occur to clear the FULL flag (set to '1'), allowing the write operation to take place.

Similarly when the FIFO is EMPTY, read operations are not allowed; read pulses are ignored internal to the logic block and are not passed on to the memory block. A write operation will clear the EMPTY flag and at that point, subsequent read operations are allowed.

The address bus to the memory module is nine bits wide. The two-bit select line forms the upper two bits of the address bus and the seven-bit address generated from the logic block forms the lower seven bits of the address bus.

FIFO Memory Configuration

In order to configure the ispLSI 6192 into multiple FIFOs, the Dual Port RAM version of the ispLSI 6192 is used. The memory block is partitioned into four equal blocks, each block being a FIFO that is 128 X 9 bits.

With this configuration, Port A is designated the read channel and Port B is the write channel. Since Port A and Port B are bi-directional, the read and write data buses can be interchanged, if so desired. The data bus for Port A connects directly to the memory block through the I/O cells and the data bus for Port B connects to the memory block through the I/O cells and generic routing pool (GRP). In this application note, Port A is the read channel and Port B is the write channel.

The address bus to the memory block is nine bits wide (512 bytes). The read select lines form the upper two bits of the read address bus, while the address pointer generated inside the programmable logic block form the lower seven bits of the read address bus. The read address bus is routed out of ispLSI 6192 and routed back into the memory block through the external I/O cells.

The write select lines form the upper two bits of the write address bus, while the address pointer generated inside the programmable logic block form the lower seven bits of the write address bus. The write address bus is routed directly to the memory block through the GRP.

The memory read and write signals are generated internally in the programmable logic block and are routed to the memory block. These signals are generated from the external nWRITE and nREAD signals. The read signal is routed out of the ispLSI 6192 and routed back into the memory block through the nCSA (channel select A) input. The write signal is routed directly to the memory block through the nCSB (channel select B) input.

FIFO Control Logic

The block diagram of the control logic is implemented in the programmable logic block as shown in Figure 3. A

Figure 3. FIFO Control Logic

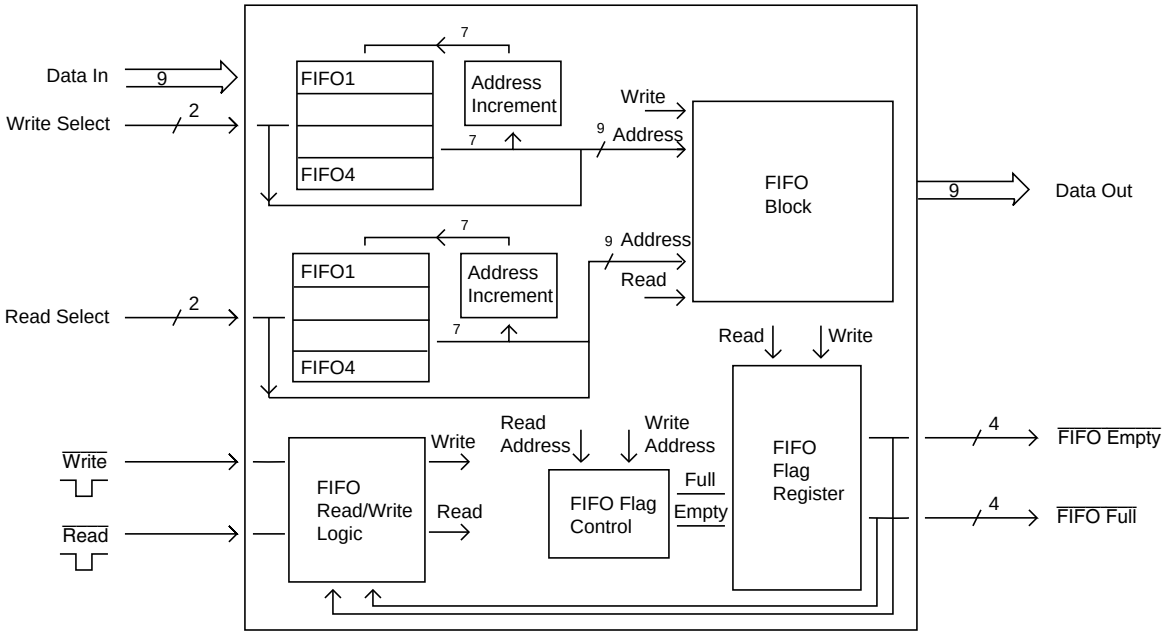
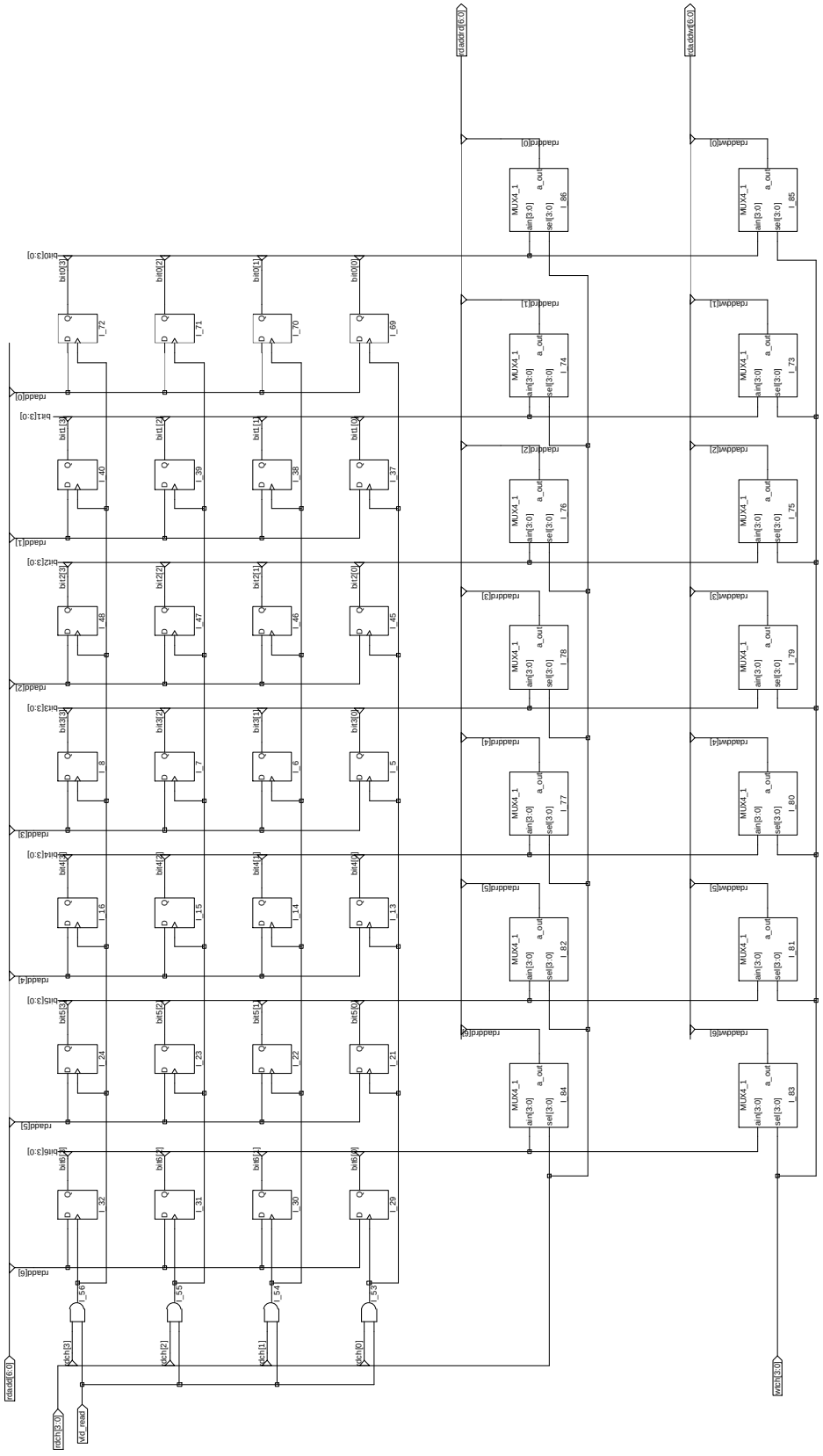


Figure 4. Read Address Register



detailed description of the logic operation is provided below.

Read / Write Address Registers

The read and write address registers store the addresses for the next FIFO read and FIFO write operations respectively. Each FIFO is 128 bytes deep, and consequently each Read Address Register contains seven storage elements and each Write Address Register contains seven storage elements. A total of 56 Flip-Flops are required for the four FIFOs being implemented . Figure 4 shows the schematic for the read address register, where 28 D Flip-Flops are used.

The next read address is loaded into the selected register (selected by RDCH[3:0]) on the rising edge of VLD-READ (valid read), which is derived from the nREAD signal. The read address for the current operation is retrieved by the RDCH[3:0] signal, which selects one of the four FIFOs being read. The WTCH[3:0] signal selects the read address for one of the four FIFOs during a write operation, so that the read address can be compared to the write address.

Read / Write Address Counters

The Read Address Counter is a basic up counter, minus the storage element. It takes the current read address from the read address register, increments the count and outputs the next read address. During a valid read operation, this next read address is loaded into the read address register. See Figure 5 for the implementation of the Read Address Counter.

The Write Address Counter is implemented similarly to the Read Address Counter.

FIFO Flag Registers

Figure 6 shows the block diagram of the FIFO Flag Registers. The three main blocks are EMPTYREG, FULLREG, and MUX4_1.

Figure 5. Read Address Counter

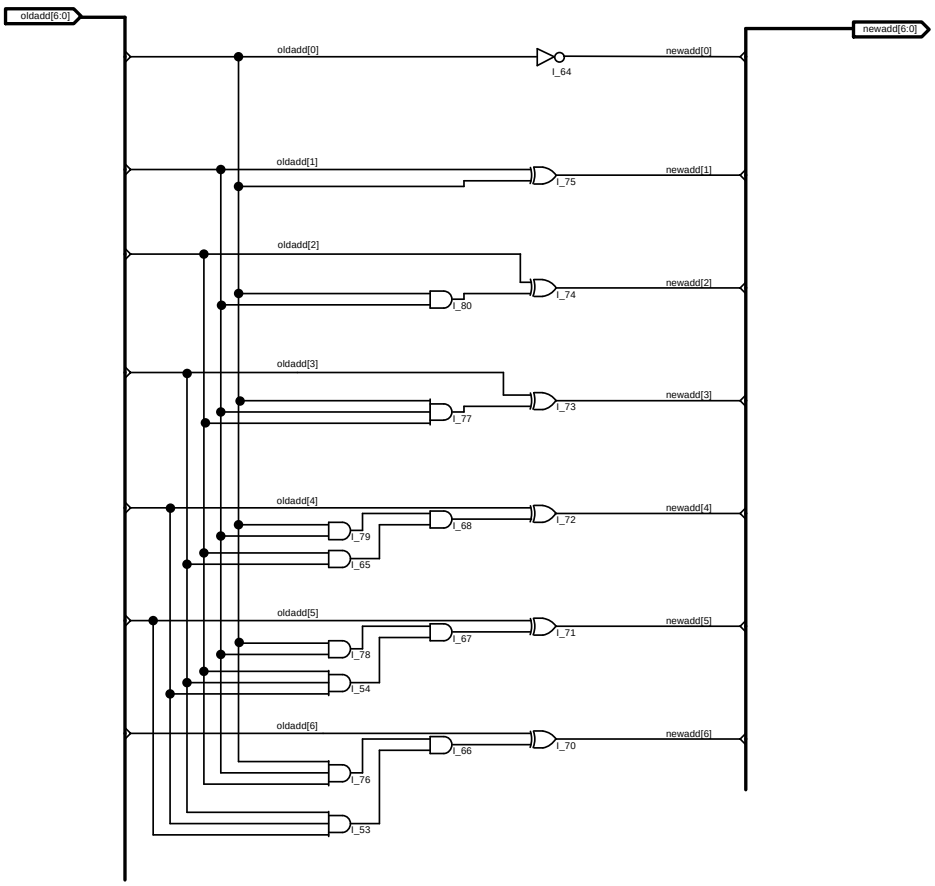


Figure 6. FIFO Flag Registers Block Diagram

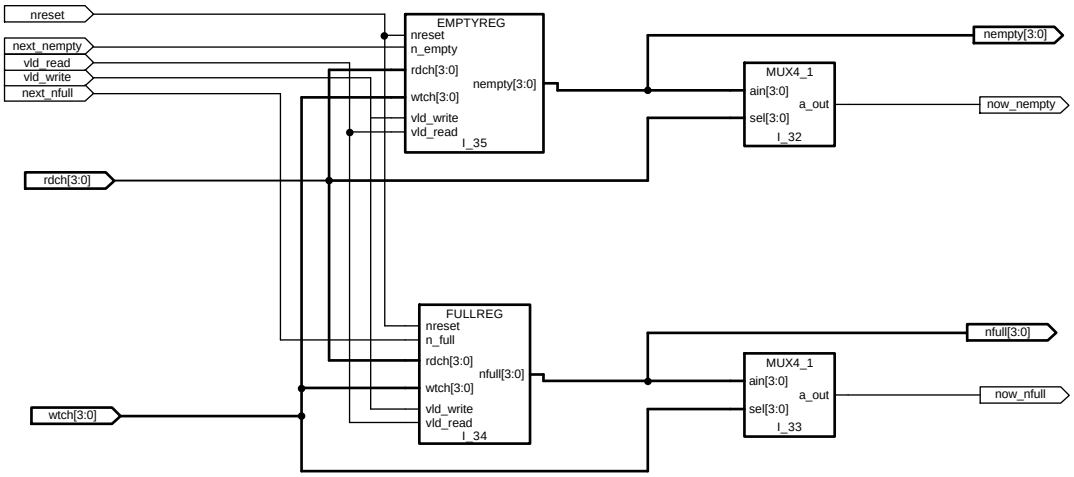
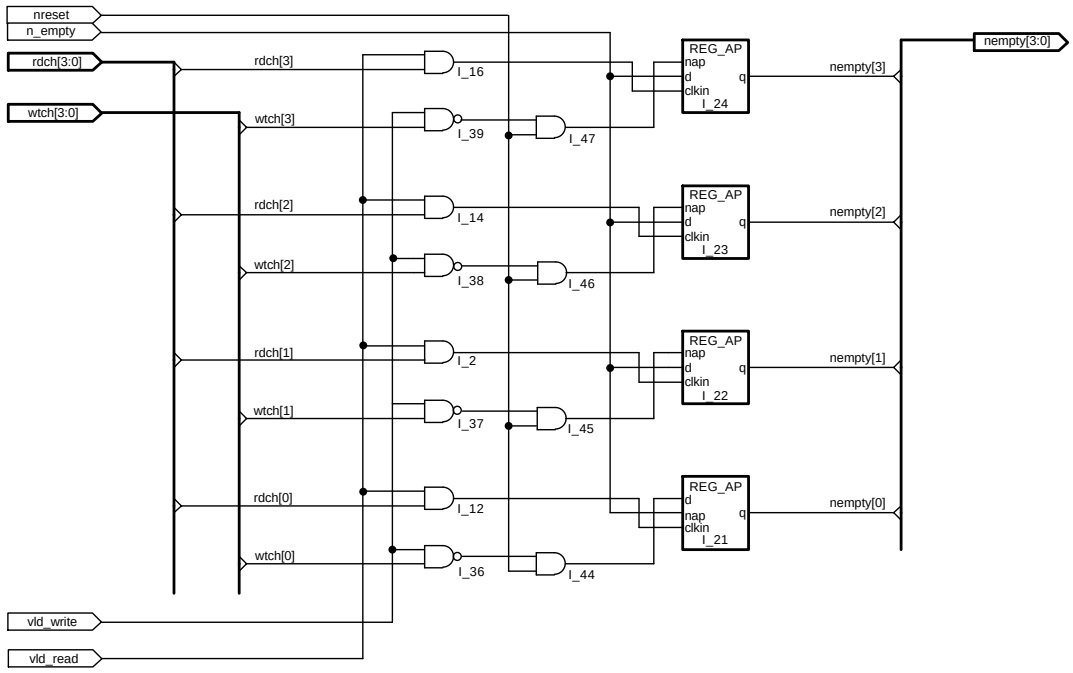


Figure 7. EMPTY Flag Registers



The function of EMPTYREG is as follows: The EMPTYREG stores the current EMPTY flag of the four FIFOs. The RDCH[3:0] selects one of the four EMPTY registers to be written to. The NEXT_NEMPTY signal is the value to be loaded into the selected register, at the rising edge of VLD_READ. WTCH[3:0] clears one of the four selected EMPTY registers during a valid write operation.

The function of FULL REG is as follows: The FULLREG stores the current FULL flag of the four FIFOs. The WTCH[3:0] selects one of the four FULL registers to be written to. The NEXT_FULL signal is the value to be loaded into the selected register, at the rising edge of VLD_WRITE. RDCH[3:0] clears one of the four FULL registers during a read operation.

The MUX4_1 selects one of the four FIFOs Flags to be used in the FIFO Read / Write Logic.

Figure 7 shows the detailed implementation of the EMPTY Register. The new value of EMPTY (n_EMPTY) is clocked into the selected register (selected by RDCH[3:0])on the rising edge of VLD_READ. The selected register is cleared (to a '1' value) on the rising edge of the VLD_WRITE. A valid write operation will clear the EMPTY flag set, allowing further read operations to take place. The REG_AP is a D-Flip Flop with asynchronous preset to allow for the asynchronous operation of VLD_WRITE to clear the EMPTY flag. A similar implementation is required for the FULL Register.

Figure 8. REG_AP Implementation

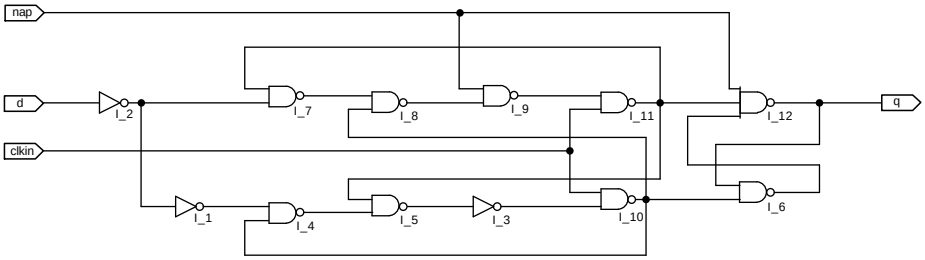


Figure 9. FIFO Flag Control Logic

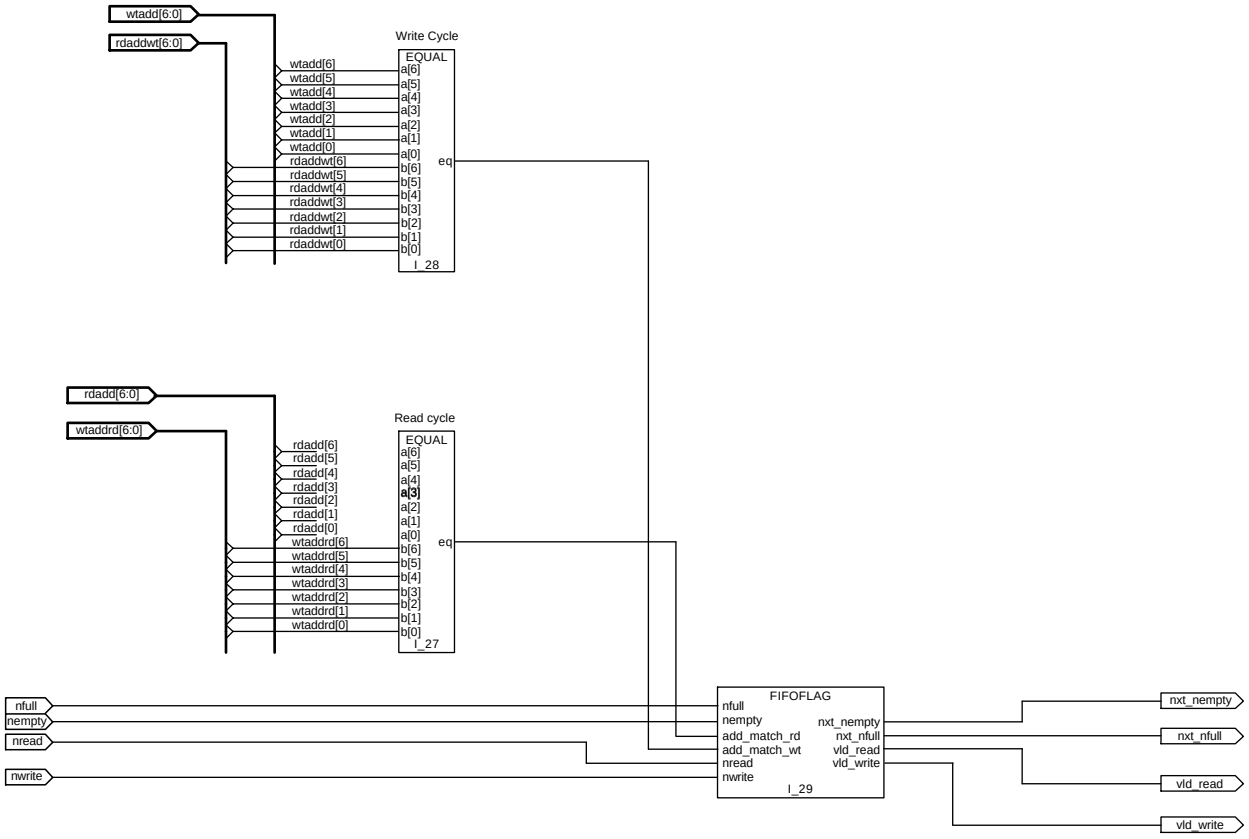


Figure 8 shows the detailed implementation of the D-Flip Flop with asynchronous preset.

FIFO Flag Control

Figure 9 shows the block diagram of the FIFO Flag Control Logic. During the read operation, the current write address (wtaddrd[6:0]) is compared to the next read address (rdadd[6:0]). The next read address is the current read address incremented by a count of 1. When the two addresses are equal, it indicates that the FIFO is empty. This results as the NXT_NEMPTY signal being asserted low. The value of NXT_NEMPTY will be stored into the FIFO Register at the end of the read operation.

Similarly during the write operation, the current read address (rdaddwt[6:0]) is compared to the next write address (wtadd[6:0]). The next write address is the current write address incremented by a count of 1. When the two addresses are equal, it indicates that the FIFO is full. This results as the NXT_NFULL signal being asserted low. The value of NXT_NFULL will be stored into the FIFO Register at the end of the write operation.

The EQUAL block in Figure 9 is a seven-bit equality comparator. EQ is asserted only when every bit of bus a[6:0] is equal to every bit of bus b[6:0].

A read operation is valid only when the FIFO is not empty. This is done by comparing the nEMPTY signal (from the selected FIFO EMPTY Register) during a read operation. A VLD_READ signal is asserted indicating that the current read operation is valid. Similarly, a write operation is valid only when the FIFO is not full and this is indicated by the VLD_WRITE signal being asserted.

Table 1 shows the ABEL file necessary to implement the FIFOFLAG block.

FIFO Read / Write Logic

The FIFO Read and FIFO Write Logic generate the VLD_READ and VLD_WRITE signals to indicate that the current read and write operations are valid. If the nEMPTY flag for the selected FIFO is high when nREAD is asserted, indicating that the FIFO is not empty, the read operation is valid. A VLD_READ signal is asserted which does several things; (a) loads the new read address into the read address register, (b) loads the new EMPTY value into the FIFO flag register, and (c) sends a MEMORY_READ signal to the Memory block so that data can be read from the FIFO.

Similarly, if the nFULL flag for the selected FIFO is high when nWRITE is asserted, indicating that the FIFO is not full, the write operation is valid. A VLD_WRITE signal is asserted which does several things; (a) loads the new write address into the write address register, (b) loads the new FULL value into the FIFO flag register, and (c) sends a MEMORY_WRITE signal to the Memory block so that data can be written to the FIFO.

Figure 10 shows the top level schematic of the FIFO Control Logic.

Table 1 - FIFOFLAG ABEL File

```
MODULE fifoflag
nfull              pin;
nempty             pin;
add_match_wt       pin;
add_match_rd       pin;
nxt_nempty         pin;
nxt_nfull          pin;
nread              pin;
nwrite             pin;
vld_read           pin;
vld_write          pin;

equations
nxt_nempty  =      !add_match_rd;
nxt_nfull   =      !add_match_wt;
vld_write   =      nfull & !nwrite;
vld_read    =      nempty & !nread;


END
```

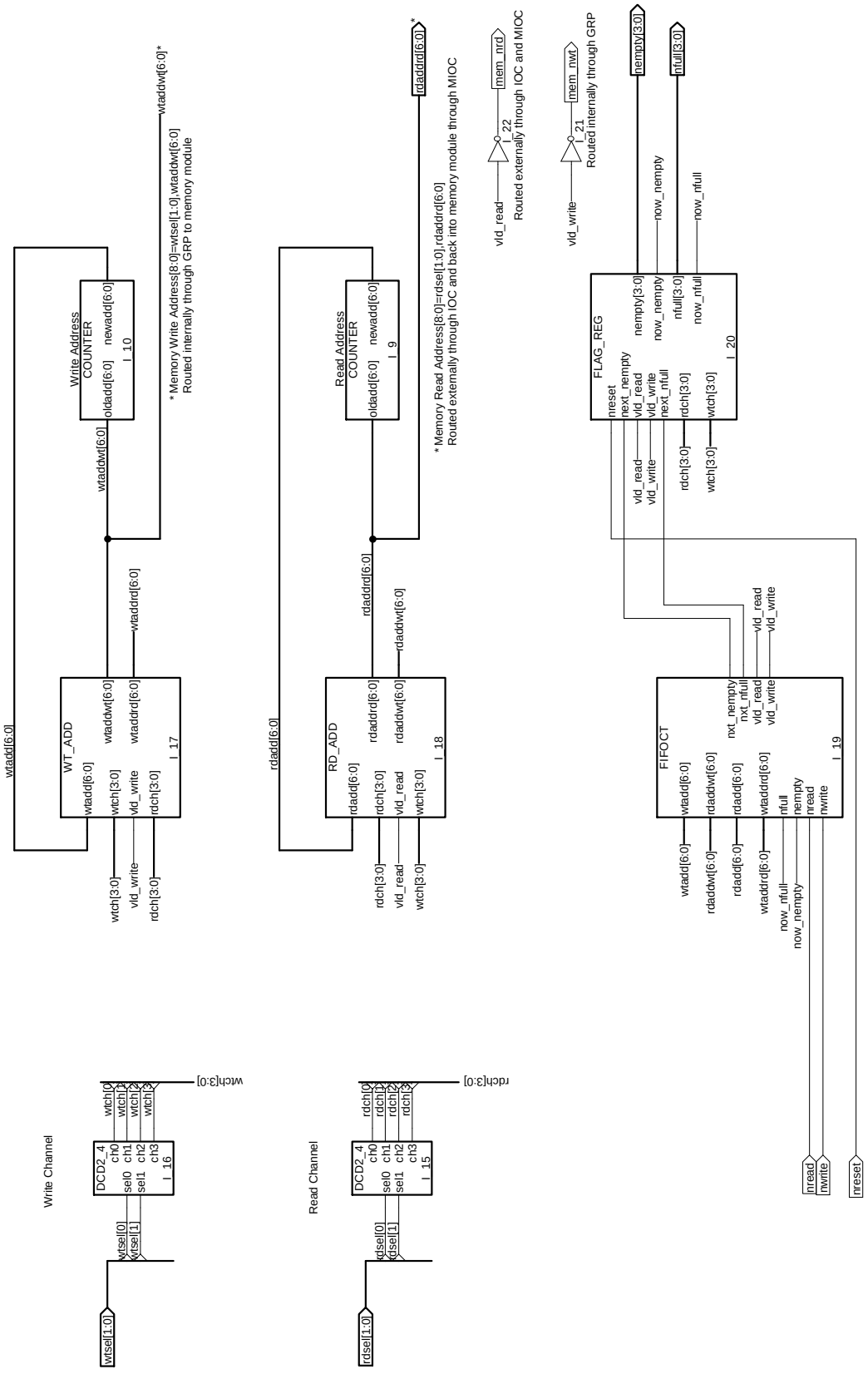
Implementation

A total of 35 GLBs on the ispLSI 6192 were used to customize the FIFO control design in Synario. This is an ideal way to make use of the programmable capability to customize functional modules.

Summary

This application note describes how the ispLSI 6192 CPLD can be configured to implement four independent uni-directional FIFOs. Having the PLD block integrated with the FIFO/Memory block allows the designer to customize the operation and configuration of the FIFO. This revolutionary PLD architecture with fixed functional blocks combine the best of the CPLD and FPGA features.

Figure 10. Top Level Schematic



Copyright © 1996 Lattice Semiconductor Corporation.

E^{2}CMOS, GAL, ispGAL, ispLSI, pLSI, pDS, Silicon Forest, UltraMOS, Lattice Logo, L with Lattice Semiconductor Corp. and L (Stylized) are registered trademarks of Lattice Semiconductor Corporation (LSC). The LSC Logo, Generic Array Logic, In-System Programmability, In-System Programmable, ISP, ispATE, ispCODE, ispDOWNLOAD, ispGDS, ispStarter, ispSTREAM, ispTEST, ispTURBO, Latch-Lock, pDS+, RFT, Total ISP and Twin GLB are trademarks of Lattice Semiconductor Corporation. ISP is a service mark of Lattice Semiconductor Corporation. All brand names or product names mentioned are trademarks or registered trademarks of their respective holders.

Lattice Semiconductor Corporation (LSC) products are made under one or more of the following U.S. and international patents: 4,761,768 US, 4,766,569 US, 4,833,646 US, 4,852,044 US, 4,855,954 US, 4,879,688 US, 4,887,239 US, 4,896,296 US, 5,130,574 US, 5,138,198 US, 5,162,679 US, 5,191,243 US, 5,204,556 US, 5,231,315 US, 5,231,316 US, 5,237,218 US, 5,245,226 US, 5,251,169 US, 5,272,666 US, 5,281,906 US, 5,295,095 US, 5,329,179 US, 5,331,590 US, 5,336,951 US, 5,353,246 US, 5,357,156 US, 5,359,573 US, 5,394,033 US, 5,394,037 US, 5,404,055 US, 5,418,390 US, 5,493,205 US, 0194091 EP, 0196771B1 EP, 0267271 EP, 0196771 UK, 0194091 GB, 0196771 WG, P3686070.0-08 WG. LSC does not represent that products described herein are free from patent infringement or from any third-party right.

The specifications and information herein are subject to change without notice. Lattice Semiconductor Corporation (LSC) reserves the right to discontinue any product or service without notice and assumes no obligation to correct any errors contained herein or to advise any user of this document of any correction if such be made. LSC recommends its customers obtain the latest version of the relevant information to establish, before ordering, that the information being relied upon is current.

LSC warrants performance of its products to current and applicable specifications in accordance with LSC's standard warranty. Testing and other quality control procedures are performed to the extent LSC deems necessary. Specific testing of all parameters of each product is not necessarily performed, unless mandated by government requirements.

LSC assumes no liability for applications assistance, customer's product design, software performance, or infringements of patents or services arising from the use of the products and services described herein.

LSC products are not authorized for use in life-support applications, devices or systems. Inclusion of LSC products in such applications is prohibited.

LATTICE SEMICONDUCTOR CORPORATION
5555 Northeast Moore Court
Hillsboro, Oregon 97124 U.S.A.
Tel.: (503) 681-0118
FAX: (503) 681-3037
http://www.latticesemi.com

November 1996