Simulation of 8257 Direct Memory Access Controller (DMA) Using VHDL

Vinod Kataria, Rajesh Kr. Saini

Reader, SKIT, Jaipur, Rajasthan, India

Abstract— *The 8257 is a 4-channel direct memory access (DMA) controller. It is specifically designed to simplify the transfer of data at high speeds for the microcomputer systems. Its primary function is to generate, upon a peripheral request, a sequential memory address which will allow the peripheral to read or write data directly to or from memory. Acquisition of the system bus in accomplished via the CPU's holds function or via cycle stealing function. It maintains the DMA cycle count for each channel and outputs a control signal for simplified sectored data transfers.8257 significantly simplifies the transfer of data at high speed between peripherals and memories. This work has been taken up to optimize and reduce area of DMA controller.*

Keywords— DMA controller, 8257, CPU, cycle stealing

I. INTRODUCTION

The 8257 DMA Controller is a peripheral chip originally developed for the Intel 8085 microprocessor, and as such is a member of a large array of such chips, known as the MCS-85 Family. This chip was later also used with the Intel 8086 and its descendants. However, most often the functionality the 8257 offered is now not implemented with the 8257 chip itself anymore, but is embedded in microcontrollers and other VLSI chips. The 8257 chip itself is still in production.

Architecture and functional description of 8257 the DMA Controller:

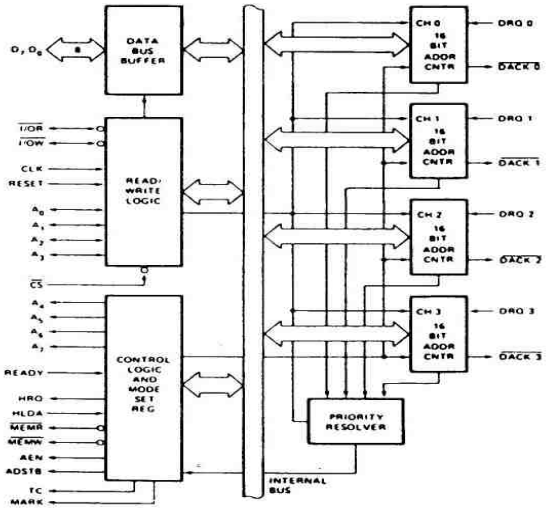

Fig. 1 Architectural representation of 8257

The 8257 is a programmable. Direct Memory Access (DMA) device which, when coupled with a single Intel 8212 I/O port device, provides a complete four-channel DMA controller for use in Intel microcomputer systems. After being initialized by software, the 8257 can transfer a block of data, containing up to 16.384 bytes. between memory and a peripheral device directly,

Working:

Upon receiving a DMA transfer request from an enabled peripheral, the 8257:

1. Acquires control of the system bus.
2. Acknowledges that requesting peripheral which is connected to the highest priority channel.
3. Outputs the least significant eight bits of the memory address onto system address lines A0-A7, outputs the most significant eight bits of the memory address to the 8212 I/O port via the data bus (the 8212 places these address bits on lines A8-A15), and
4. Generates the appropriate memory and I/O read/ write control signals that cause the peripheral to receive or

deposit a data byte directly from or to the addressed location in memory.

The 8257 will retain control of the system bus and repeat the transfer sequence. as long as a peripheral maintains its DMA request. Thus, the 8257 can transfer a block of data to/from a high speed peripheral (e.g.. a sector of data on a floppy disk) in a single "burst". When the specified number of data bytes have been transferred, the 8257 activates its Terminal Count (TC) output, informing the CPU that the operation is complete.

The 8257 offers three different modes of operation: (1) DMA read, which causes data to be transferred from memory to a peripheral: (2) DMA write, which causes data to be transferred from a peripheral to memory: and (3) DMA verify, which does not actually involve the transfer of data When an 8257 channel is in the DMA verify mode, it will respond the same as described for transfer operations, except that no memory or I/O read/write control signals will be generated. thus preventing the transfer of data. The 8257. however, will gain control of the system bus and will acknowledge the peripheral's DMA request for each DMA cycle The peripheral can use these acknowledge signals to enable an internal access of each byte of a data block in order to execute some verification procedure, such as the accumulation of a CRC (Cyclic Redundancy Code) check word For example, a block of DMA verify cycles might follow a block of DMA read cycles (memory to peripheral) to allow the peripheral to verify its newly acquired data

II. DMA OPERATION

Single byte transfer:

single byte transfer is initiated by the 110 device raising the DRO line of one channel of the 8257. If the channel is enabled, the 8257 wilt output a HRO to the CPU. The 8257 now waits until a HLDA is received insuring that the system bus is free for its use. Once HIDA is received the DACK line for the requesting channel is activated (LOW). The DACK line acts as a chip select for the requesting 110 device. The 8257 then generates the

read and write commands and byte transfer occurs between the selected 110 device and memory. After the transfer Is complete, the DACK line is set HIGH and the HRO line Is set LOW to Indicate to the CPU that the bus is now free for use. DRO must remain HIGH until DACK is issued to be recognized and must go LOW before S4 of the transfer sequence to prevent another transfer from occurring. (See timing diagram.)

Consecutive Transfers :

If more than one channel requests service simultaneously y, the transfer will occur In the same way a burst does. No overhead is incurred by switching from one channel to another. In each S4 the DRO lines are sampled and the highest priority request is recognized during the next transfer. A burst mode transfer in a lower priority channel will be overridden by a higher priority request. Once the high priority transfer has completed control will return to the lower priority channel if its DRO is still active. No extra cycles are needed to execute this sequence and the HRO line remains active until all DRO lines go LOW.

Control Override.

The continuous DMA transfer mode described above can be interrupted by an external device by lowering the HIDA line. After each DMA transfer the 8257 samples the HLDA line to insure that it Is still active, lilt is not active, the 8257 completes the current transfer, releases the HRQ line (LOW) and returns to the idle state. If DRO lines are still active the 8257 will raise the HRO line In the third cycle and proceed normally .

The 8257 has a Ready input similar to the 8080A and the 8085k The Ready line is sampled in State 3. If Ready is LOW the 8257 enters a wait state. Ready is sampled during every wait state. When Ready returns HIGH the 8257 proceeds to State 4 to complete the transfer. Ready is used to interface memory or 110 devices that cannot meet the bus set up times required by the 8257.

The 8257 uses four clock cycles to transfer a byte of data. No cycles are lost in the master to master transfer maximizing bus efficiency. A 2MHz clock input will allow the 8257 to transfer at a rate of 500K bytes/second.

Memory Mapped 110 Configurations

The 8257 can be connected to the system bus as a memory device instead of as an I/O device for memory mapped **I/O** configurations by connecting the system memory control lines to the 8257's I/O control lines and the system i/O control tines to the 825Ts memory control tines.

This configuration permits use of the 8080s considerably larger repertoire of memory instructions when reading or loading the 8257s registers Note that with this connection, the programming of the Read (bit 15) and Write (bit 14) bits in the terminal count register will have a different meaning .

Observation :

1. On giving the clock cycle all signal are presented on the screen. We sees the following observations:

- All the signals are initialized.
- All the terminal counter registers are set to "00000000000000".
- The intermediate signal a,b,c,d are set to "0000".
- Intermediate integers e, f, g, h are set to 0.
- We use the fixed priority in this project so the priority signal is set to 0 for fix.
- All other signals are uninitialized with U.

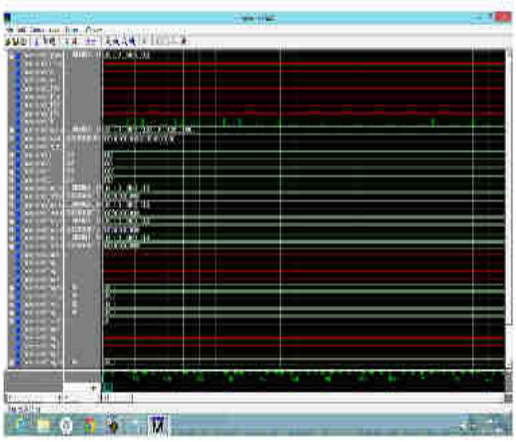

Fig. 2 Signals after giving clock

2. Connecting the peripheral to the channel DMA sends the Hold request to the CPU. On connecting the peripheral to Channel 0 and channel 1. We observe the following observations:

- Input from the peripheral is 1100 that is assigned to the temo.
- This value of temo. is assigned to the DRQ as 1100.
- But channel 0 has highest priority. The intermediate signal DRQ line will 1000.
- The HRQ line signal is generated to the corresponding this DRQ signal will be 1000. Now this signal goes to CPU for holding request.

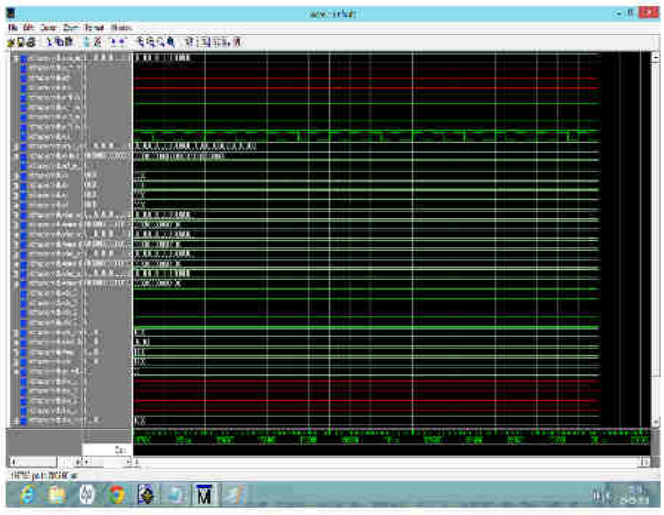

Fig. 3 signals after requesting peripheral

3. DRQ line goes to CPU we observe following observations:

- If CPU sees its data bus free it sends Hold acknowledge to DMA.
- So the MSB of CPU memory register is set to "00000000".
- CPU generates the hold acknowledge (HLDA line) 1000. So the channel 0 is selected for data transfer.
- DMA acquires the control signal.

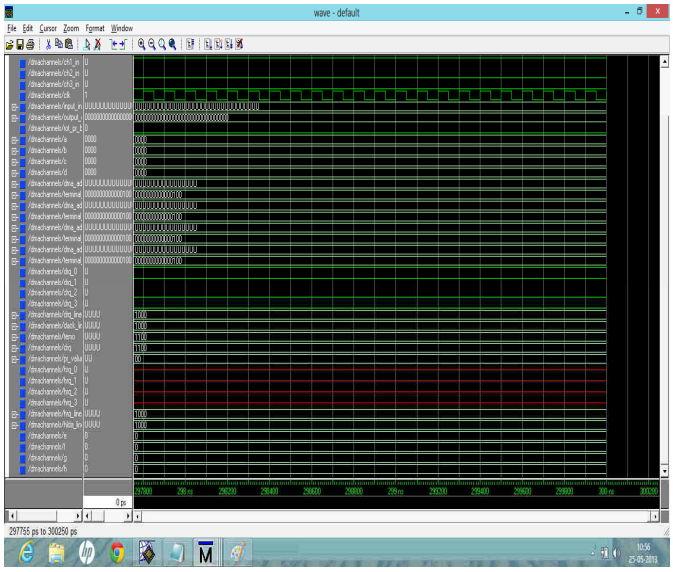

Fig.4 Signals after HLDA

4. DMA generates the Read and Write signal and data are transfer between peripheral to CPU memory. DMA comes

to master mode. Data to be transferred is 80C0E0F0. It will generate the IOR read signal. We observe the following observations:

- When IOR=1 and IOW=0 then byte C0 read from peripheral.
- When IOR=0 and IOW=0 then byte C0 is written to CPU memory register.
- When IOR=1 and IOW=1 then byte C0 transferred to DMA register from CPU memory.
- When IOR=0 and IOW=1 then byte C0 transferred to peripheral.

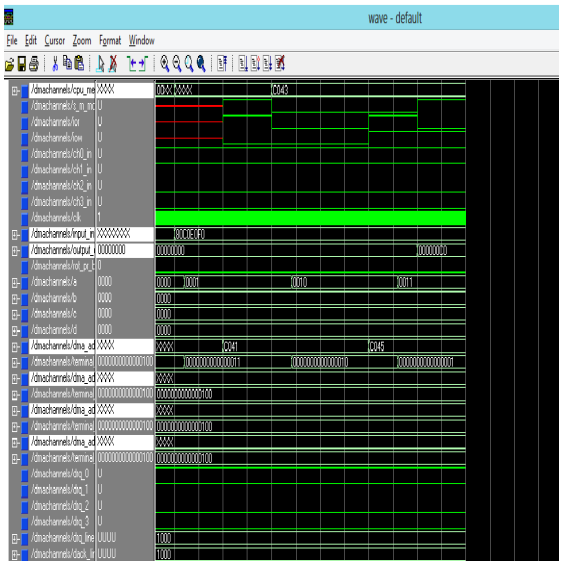

Fig.4 Byte transferred to memory and Peripheral

The observation shown above completes one cycle of byte transfer from memory to peripheral . Similar observations can be taken for byte transfer from peripheral to memory.
Based on the above observations the following results were obtained.

III. RESULT

From above observation we got a byte transferred from peripheral to memory at the specified location.

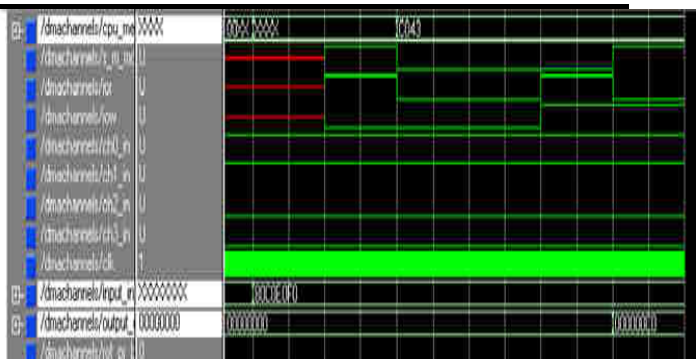

Fig. 5 Result

Above observed screen indicates the byte transfer.

IV. CONCLUSION

In view of the above results of the DMA source code, we can conclude that-

- The system simulated by us functions as 8257 IC chip functions.
- Based on results, we conclude that the simulated program implemented by us in VHDL is 8257 IC.
- During the simulation the byte is transferred to CPU memory successfully.
- We know that as the application of microprocessors in our technology intensive environment is increasing, so we have to develop practical skills in the areas of computer applications, networking, and interfacing. This project 8257 DMA using VHDL is based upon simulated test bench waveform analysis. After doing this project we are able to provide manufacturing details to fabricate the chip.

REFERENCES

[1] "The Practical Xilinx designer lab book", Writer: David Van den Bout Published by Xilinx Corporation.
[2] "A VHDL Primer". Writer: J. BhaskarPublished by Pearson Education
[3] "Programmable DMA Controller 8257". Published by Intel.