PROJECT 2

DATA MEMORY ACCESS (8237A)

Faculty of Engineering - Ain Shams University - Computer department

SUBMITTED BY:
Group(3):
Pierre Nabil (16E0056)
Gerges Michael (1600446)
John Bahaa (1600459)
Ibrahim Shoukry (1600011)
David John(1600531)
Adham Noor el Wafaa (1600226)
Ziad Tarek (1600606)

Subject
Computer Organization II

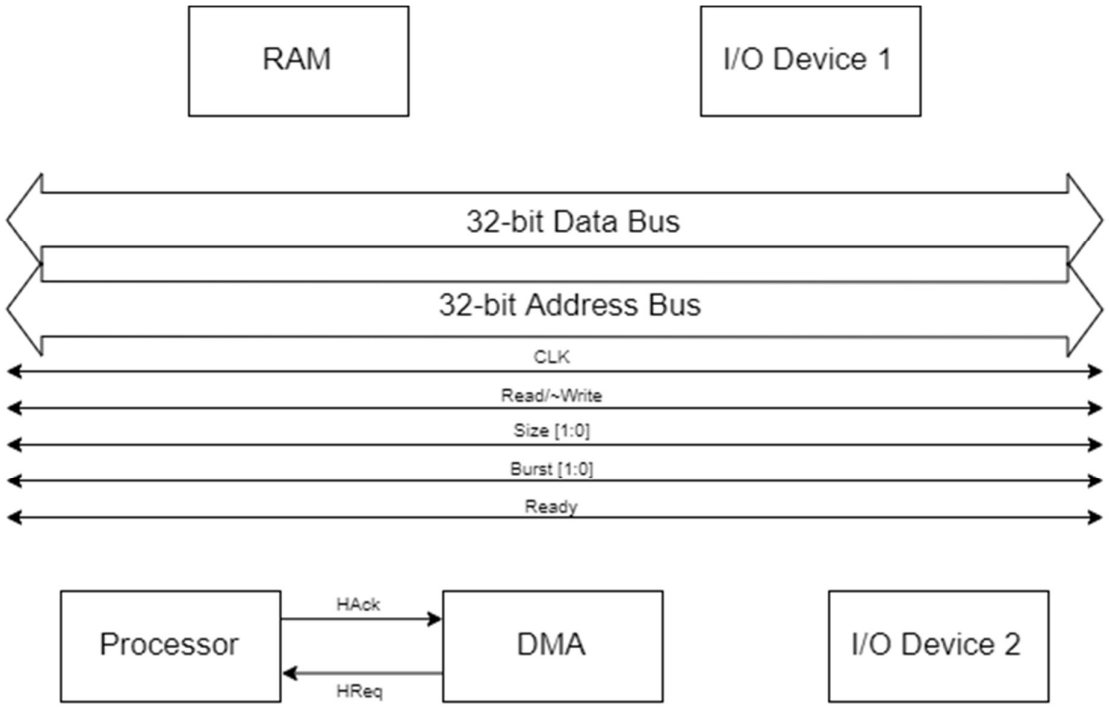
Contents

Introduction: 2
External Connections: 2
Supported Bus: 2
Internal Connections: 3
DMA Pins: 3
Internal Registers: 4
Command / Transaction Register: 4
Mask Register: 4
Request Register: 4
Mode Register: 5
Status Register: 5
Temporary FIFO Buffer: 5
Base/ Current Address/Word Count Registers: 5
Programming the DMA: 6
Design Choices: 7
Disclaimer Note: 8

Introduction:

In this project, we decided to create an 8237A Direct Memory Access IC using Verilog. We decided to create a 32-bit version that works with a Bus similar to the AHB Bus using the AHB protocol. However, this choice made it that many of the included feature of the original 8237A are obsolete! So we decided to alter some of the features and enhance other features to create a better DMA.

External Connections:

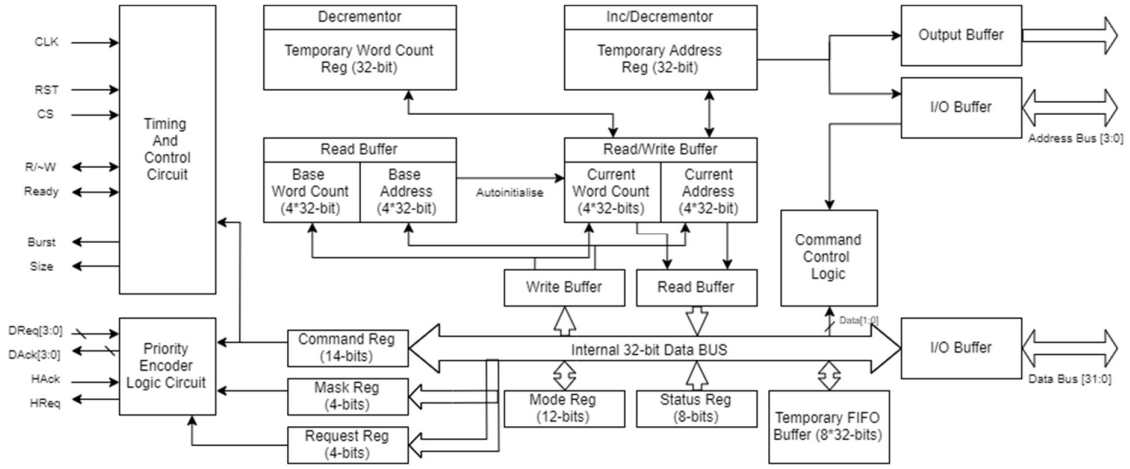


Supported Bus:

The Bus is exactly the same as the simplified AHB bus that we take I the syllabus but has a sing 32-bit Data bus (used both for read and write operations) & a single Read/Write line. The bus supports sending data of size Byte (8-bit), Half Word (16-bit) & Word (32-bit). It also supports Bursts of 1, 2, 4 & 8-beat Data. It has the following Signals:

Signal Name	***Size***	***Description***
CLK	1-bit	The standard Clock Signal
Address	32-bit	Uses the 32-bit address space required
Data	32-bit	Used for both read and write instructions
Read/~Write	1-bit	Defines the operation (1=Read, 0=Write)
Size	2-bit	Defines the type of Data to be used (0=Byte, 1=Half Word, 2&3=Word)
Burst	2-bit	Defines how many bursts of data are used in a single operation (0=1beat, 1=2beats, 2=4beats, 3=8beats)
Ready	1-bit	Used by the slave to tell the Master that It's Ready to Read Data or Write Data to the Data Bus.

Internal Connections:



DMA Pins:

Pin Name	***Size***	***Description***
CLK	1-bit	**Clock** Input
CS	1-bit	**Chip Select:** It's an Active High Input used when the DMA is called by the processor to be programmed. It's connected to an external decoder which is connected to the Most Significant Bits of the Address Bus
RST	1-bit	**Reset:** It's an Active High Input that resets the contents of all the Registers in the DMA
RDY	1-bit	**Ready:** It's an Active High Input sent by the slave to tell the DMA that it's ready to read or write
HReq	1-bit	**Hold Request:** It's an active High output sent to the Processor to tell it that the DMA wants the Bus
HAck	1-bit	**Hold Acknowledge:** It's an Active High Input sent by the processor to tell the DMA to take over the BUS. It changes the DMA from the Idle Mode (Programming) to the Active Mode (Memory Access)
DReq	4-bits	**Device Request:** This Signal is used only for the Cascade Mode of the Device. Up to 4 other DMAs can send their HReq thorough those pins and the DMA will Send the request through Its Own HReq. Note that the DReq pins are unrelated to the Channels Programmed to the DMA! This means that the "master" DMA can still hold 4 Channels to Access their Memories.
DAck	4-bits	**Device Acknowledge:** It's how the DMA can give the HAck back to the Requesting DMAs. Note that the 'master" DMA only sends the DAck to the other DMAs after it is done with Its own Data Transfer First. The DMA chooses which DMA to give DAck to based on the Priority Setting Given during the programming of the DMA (Fixed Priority or Rotating Priority)
R/~W	1-bit	**Read Write:** It's an InOut signal that tells the bus whether the master wants to read or write to the slave.
Data	32-bits	**Data:** Connected to the Data Bus of the System
MSA	28-bits	**Most Significant Address:** is an Output Signal to call the Slaves During the Active Mode.
LSA	4-bits	**Least Significant Address:** is an InOut Signal used to select which internal Register to write to during the Idle mode and used to write the address during the Active mode.

Pin5	1-bit	**Pin5:** "This pin should always be at logic HIGH level. An internal pull-up resistor will establish a logic high when the pin is left floating. It is recommended however, that PIN5 be connected to VCC." This pin exists only for meme purposes!! XD

Internal Registers:

Command / Transaction Register:

It's the Register responsible for programming the whole DMA. It tells the DMA the source and destination for 2 Transactions. Transaction 1 is always done first. It also stores the size of the data that is to be transferred. A lot of options for this problem were discussed during the design of this DMA, and we found the 2 Transactions is optimal!

Bit No:	**Description:**
1:0	Transaction 1 beat size
3:2	Transaction 1 Source Channel No.
5:4	Transaction 1 Destination Channel No.
7:6	Transaction 2 beat size
9:8	Transaction 2 Source Channel No.
11:10	Transaction 2 Destination Channel No.
12	Controller Enable (1 = Tells the DMA to stop sending HReq until this Bit is Reset to 0)
13	Rotating Polarity/ Fixed Polarity for other DMAs (1=Rotating, 0=Fixed)

Mask Register:

It's a Register that Masks the Request of DMAs connected via DReq.

Bit No:	**Description:**
0	Mask DReq for the DMA connected at DReq[0]
1	Mask DReq for the DMA connected at DReq[1]
2	Mask DReq for the DMA connected at DReq[2]
3	Mask DReq for the DMA connected at DReq[3]

Request Register:

It's a Write Register where the Processor can give a forced DReq for a DMA connected via DReq.

Bit No:	**Description:**
0	Assume DReq for the DMA connected at DReq[0]
1	Assume DReq for the DMA connected at DReq[1]
2	Assume DReq for the DMA connected at DReq[2]
3	Assume DReq for the DMA connected at DReq[3]

Mode Register:

It's a Register that Selects the mode of Each Channel of the DMA.

Bit No:	Description:
0	Autoinitialize Channel 0 Current Registers
1	Channel 0 Address Decrement or Increment (0 = Increment, 1 = Decrement)
2	Channel 0 Address Hold
3	Autoinitialize Channel 1 Current Registers
4	Channel 1 Address Decrement or Increment (0 = Increment, 1 = Decrement)
5	Channel 1 Address Hold
6	Autoinitialize Channel 2 Current Registers
7	Channel 2 Address Decrement or Increment (0 = Increment, 1 = Decrement)
8	Channel 2 Address Hold
9	Autoinitialize Channel 3 Current Registers
10	Channel 3 Address Decrement or Increment (0 = Increment, 1 = Decrement)
11	Channel 3 Address Hold

Status Register:

It's a Read Only Register used to state the TC (Terminated Counter) of each Channel and the request of the 4 paired DMA Devices connected via DReq & DAck.

Bit No:	Description:
0	Channel 0 has reached Terminating Counter
1	Channel 1 has reached Terminating Counter
2	Channel 2 has reached Terminating Counter
3	Channel 3 has reached Terminating Counter
4	DMA 0 Request
5	DMA 1 Request
6	DMA 2 Request
7	DMA 3 Request

Temporary FIFO Buffer:

It's a First In First Out Register used to store data that is being sent from one channel to another. We designed the system to have a maximum of 8-beats sent at a time so no more 8 word Registers were required.

Base/ Current Address/Word Count Registers:

They are exactly similar to the 8237A DMA, except that they are 32-bits wide instead of 16-bits! They are used to keep track of the Addresses of the 4 Channels and their respective Word Count (Words to be Sent/Received).

Programming the DMA:

The DMA is programmed by giving it the required instructions in the form of Address/ReadWrite combinations. The combinations are given in the table below.

Most of the instructions are straightforward and don't need explanation.

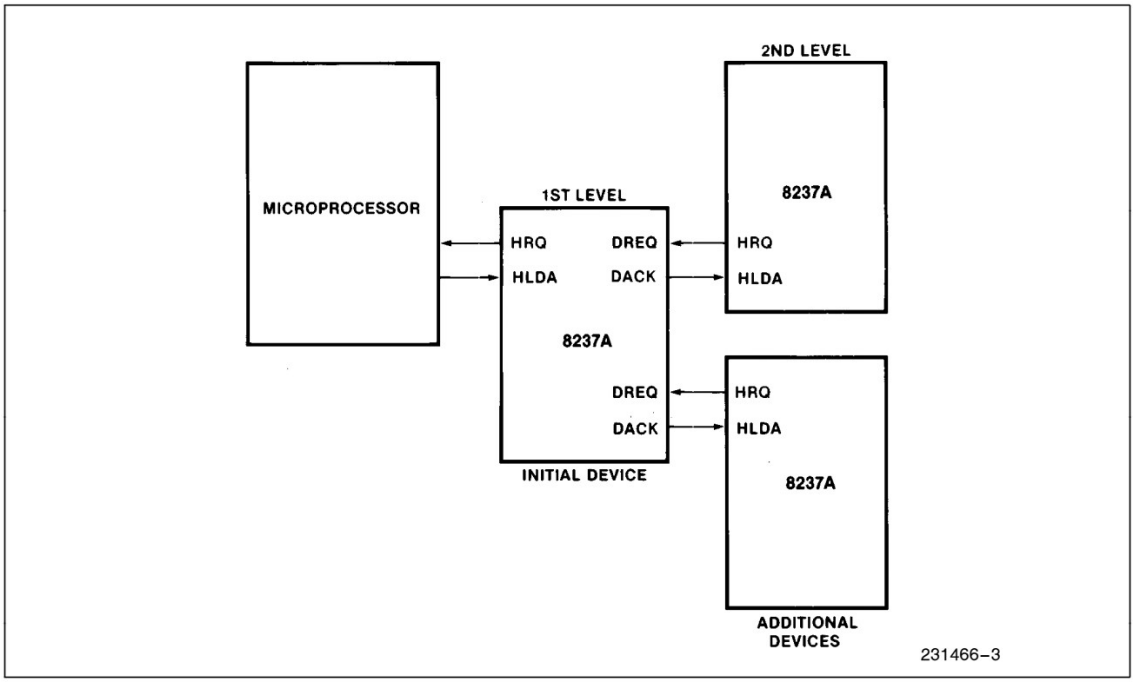
- Master Clear works as a software RST Pin.
- Read Last Temporary Register removes the last element from the Temporary Register. You have been warned!!!
- For writing Mode and Mask registers there are 2 options:
  1. You can write for the whole register in one shot using the Write All instruction. This works similar to any other Write instruction.
  2. You can write the Register for 1 Channel only. This is done by using the 2 least significant data pins to specify the channel, The use the next least significant bit (for Mask) or 3-bits (for Mode) to write the required data for the given Channel.

LS Address	R/~W	Operation
0000	0	Channel 0 Base & Current Address Write
0000	1	Channel 0 Current Address Read
0001	0	Channel 0 Base & Current Word Count Write
0001	1	Channel 0 Current Word Count Read
0010	0	Channel 1 Base & Current Address Write
0010	1	Channel 1 Current Address Read
0011	0	Channel 1 Base & Current Word Count Write
0011	1	Channel 1 Current Word Count Read
0100	0	Channel 2 Base & Current Address Write
0100	1	Channel 2 Current Address Read
0101	0	Channel 2 Base & Current Word Count Write
0101	1	Channel 2 Current Word Count Read
0110	0	Channel 3 Base & Current Address Write
0110	1	Channel 3 Current Address Read
0111	0	Channel 3 Base & Current Word Count Write
0111	1	Channel 3 Current Word Count Read
1000	0	Write to Command Register
1000	1	Read Status Register
1001	0	Write Request Register
1001	1	XX Illegal
1010	0	Write Single Mask Register
1010	1	XX Illegal
1011	0	Write Single Mode Register
1011	1	XX Illegal
1100	0	Write All Mode Register
1100	1	XX Illegal
1101	0	Master Clear
1101	1	Read Last Temporary Register
1110	0	Clear Mask Register
1110	1	XX Illegal
1111	0	Write All Mask Register
1111	1	XX Illegal

Design Choices:

We decided to remove some “options” from the DMA for multiple reasons and these will be stated below each with its own reasoning:

1. We removed the Address Enable and Address Strobe Pins since these were used to output 16-bit addresses from the 8-bit Address Pins and 8-bit Data Pins. This was done using an internal FlipFlop and an external 8-bit Latch. This option was removed since we were already planning to increase the no. of pins due to the 32-bit address space which was required. So we created all 32 Address pins and 32-Data pins. However, this increases the Pin number to a total of 80 pins.
2. We chose to change the 4 Read Write pins (MEMR, MEMW, IOR, IOW) to a single Read/Write pin which is more efficient for the pseudo-AHB bus that we designed.
3. We chose to remove the EOP (End Of Process) pin because it seemed to have no use in the case of the an AHB bus! It might have had a purpose for the ISA bus it was designed for but it has no purpose for our Bus.
4. The DReq and DAck signals were used to cascade DMAs together. This is used instead of the usual use of the DReq signals which we neglected because it didn’t fit with the purpose of having a DMA in the first place as it works more as an interrupt than a Request Signal… Therefore we left all DReq functionality for a PPI IC and/or an Interrupt Controller to make the design slightly easier and to make the functionality more focused.
5. The Cascade Mode wors differently than the 8237A in that the 8237A was inefficient with using its cascade mode for many reasons. By using the DReq signals to cascade the DMAs and making the DReq signals independent of the Memory Channels, we managed to make each DMA can cascade with 4 other DMAs while still keeping its original functionality by still being able to hold data for 4 separate Channels for Memory Access. This means that no channels are wasted in the cascading process! This also means that if we wanted to save 8 Channels, this would only require 2 DMAs instead of the Classic 8!!



6. We Chose to use a FIFO Temporary Buffer for moving Data instead of the single 1 Byte Register in the 8237A. This is to make Data Transfer faster as we can now send 1, 2, 4 or 8 beats of data in 1 bus operation. This reduces the overhead of the transfer speed and makes the transfer faster in most cases. However, since the bus only supports exactly 1, 2, 4 or 8 beats per operation, if we wanted to transfer 7 beats for example this would require 3 operations (4+2+1) even though it would take 1 Instruction for 8 beats!
7. For the Mode Register, We Decided that the Transfer Modes (Read, Write & Verify) were useless as we have a Read/Write signal in our Bus and we put this functionality in the Command Register.
8. We also Decided that the modes of operation were also useless since single transfer is too slow (and Block Transfer can always be stopped by setting HAck to 0) and all transfers will be Demand Transfer since we have a Ready Signal in our Bus! Finally, the Cascade mode was removed as stated before in point no. 6.
9. For the Command Register, we decided to remove the Memory to Memory enable and disable since the DMA treats everything as Registers of Memory!!!
10. The Channel Address Hold was moved to the Mode Register to be Channel specific! This helps because now you can move the same word to multiple places.
11. Normal Timing And Compressed Timing were removed as we were using a strict protocol which was as as efficient as possible so there was no need for this Option.
12. Late Write Selection and Extended Write Selection were also Removed.
13. Since the DReq and DAck were redesigned to be used on different instances of the same Chip, we decided to standardize their activity as Active High! Because it was useless to make it otherwise.
14. Finally, since we have 4 channels then the maximum no. of possible transactions without redundancy are 3 transactions (if we will copy something into the 3 other channels) OR 2 transactions (ex: Copy Ch0 to Ch1 & Ch2 to Ch3). We decided to go with a maximum of 2 Transactions. This is because programming the DMA takes time for the first setup, but after that you only need a minimum no. of instructions to reprogram the DMA for more Transactions (Assuming that we don't Change the Base Address and Word Count for the Channels). So, we decided to store the Source, Destination and Size of each Transaction into the Command Register. If the processor wants only one transaction it can set the source and destination to the same channel and the DMA will understand that this Transaction is not necessary.

Disclaimer Note:

This Project was too large to be done during the given timeframe! As a result, some of the features of the DMA may not work optimally!!! Every part of the DMA was tested separately after it was built. However, we didn't have enough time to retest everything again after the whole DMA was assembled. However, if we had more time and no exams in the near future, we could have made the DMA fully functional & synthesizable and maybe even printed on an FPGA!!!