Burst mode (computing)

Burst mode is a generic electronics term referring to any situation in which a device is transmitting data repeatedly without going through all the steps required to transmit each piece of data in a separate transaction.

Advantages

The main advantage of burst mode over single mode is that the burst mode typically increases the throughput of data transfer. Any bus transaction is typically handled by an arbiter, which decides when it should change the granted master and slaves. In case of burst mode, it is usually more efficient if you allow a master to complete a known length transfer sequence.

The total delay in a data transaction can be typically written as a sum of initial access latency plus sequential access latency.

$$t_{total} = t_{initial} + t_{sequential}$$

Here the sequential latency is same in both single mode and burst mode, but the total initial latency is decreased in burst mode, since the initial delay (usually depends on FSM for the protocol) is caused only once in burst mode. Hence the total latency of the burst transfer is reduced, and hence the data transfer throughput is increased.

It can also be used by slaves that can optimise their responses if they know in advance how many data transfers there will be. The typical example here is a DRAM which has a high initial access latency, but sequential accesses after that can be performed with fewer wait states.[1]

Beats in burst transfer

A beat in a burst transfer is the number of write (or read) transfers from master to slave, that takes place continuously in a transaction. In a burst transfer, the address for write or read transfer is just an incremental value of previous address. Hence in a 4-beat incremental burst transfer (write or read), if the starting address is 'A', then the consecutive addresses will be 'A+m', 'A+2*m', 'A+3*m'. Similarly, in a 8-beat incremental burst transfer (write or read), the addresses will be 'A', 'A+n', 'A+2*n', 'A+3*n', 'A+4*n', 'A+5*n', 'A+6*n', 'A+7*n'.

Example

Q:- A certain SoC master uses a burst mode to communicate (write or read) with its peripheral slave. The transaction contains 32 write transfers. The initial latency for the write transfer is 8ns and burst sequential latency is 0.5ns. Calculate the total latency for single mode (no-burst mode), 4-beat burst mode, 8-beat burst mode and 16-beat burst mode. Calculate the throughput factor increase for each burst mode.

Sol:-

Total latency of single mode = num_transfers x ($t_{initial}$ + $t_{sequential}$) = 32 x (8 + 1x(0.5)) = 32 x 8.5 = 272 ns

Total latency of one 4-beat burst mode = ($t_{initial}$ + $t_{sequential}$) = 8 + 4x(0.5) = 10 ns

For 32 write transactions, required 4-beat transfers = 32/4 = 8

Hence, total latency of 32 write transfers = 10 x 8 = 80 ns

Total throughput increase factor using 4-beat burst mode = single mode latency/(total burst mode latency) = 272/80 = 3.4

Total latency of one 8-beat burst mode = ($t_{initial}$ + $t_{sequential}$) = 8 + 8x(0.5) = 12 ns

For 32 write transactions, required 8-beat transfers = 32/8 = 4

Hence, total latency of 32 write transfers = 12 x 4 = 48 ns

Total throughput increase factor using 8-beat burst mode = single mode latency/(total burst mode latency) = 272/48 = 5.7

Total latency of one 16-beat burst mode = ($t_{initial}$ + $t_{sequential}$) = 8 + 16x(0.5) = 16 ns

For 32 write transactions, required 16-beat transfers = 32/16 = 2

Hence, total latency of 32 write transfers = 16 x 2 = 32 ns

Total throughput increase factor using 16-beat burst mode = single mode latency/(total burst mode latency) = 272/32 = 8.5

From the above calculations, we can conclude that the throughput increases with the number of beats.

Details

The usual reason for having a burst mode capability, or using burst mode, is to increase data throughput.[2] The steps left out while performing a burst mode transaction may include:

- Waiting for input from another device
- Waiting for an internal process to terminate before continuing the transfer of data

- Transmitting information which would be required for a complete transaction, but which is inherent in the use of burst mode[3]

In the case of DMA, the DMA controller and the device are given exclusive access to the bus without interruption; the CPU is also freed from handling device interrupts.

The actual manner in which burst modes work varies from one type of device to another; however, devices that have some sort of a standard burst mode include the following:

- Random access memory (RAM), including EDO, SDRAM, DDR SDRAM, and RDRAM; only the last three are required to send data in burst mode, according to industry standards
- Computer busses such as Conventional PCI, Accelerated Graphics Port, and PCI express
- Hard disk drive (HDD) interfaces such as SCSI and IDE

See also

Electronics portal

- Asynchronous I/O
- Command queue
- Direct memory access (DMA)
- SDRAM burst ordering
- Scatter/gather I/O

References

1. "ARM forums" (https://community.arm.com/developer/ip-products/system/f/soc-design-forum/12883/what-purpose-does-burst-feature-in-ahb-serve). April 2019.
2. *PCI Local Bus Specification Revision 2.2*. Hillsboro, Oregon: PCI Special Interest Group. December 18, 1998. p. 82.
3. *PCI Local Bus Specification Revision 2.2*. Hillsboro, Oregon: PCI Special Interest Group. December 18, 1998. p. 29.