WISHBONE wrapper for Xilinx Memory Interface Generator (MIG)

Version 0.1
April 5, 2011

Paul Baker
pbaker@engineer.com

Preface

Xilinx is a Registered Trademark of Xilinx Inc., and the generated memory interface module is the intellectual property of Xilinx. In accordance with Xilinx's requirements, none of their generated Design has been copied, reproduced, distributed, republished, downloaded, displayed, posted, or transmitted in any form in this specification or supporting files.

The use of Xilinx MIG and the generated Design is beyond the scope of this project, please refer to Xilinx for any questions regarding the program or generated modules. The author has relied upon "*Memory Interface Solutions User Guide*" UG086 (v3.5) article published July 23, 2010[1] and "*Wishbone B4"* published 2010[2] to formulate the wrapper.

Unless otherwise stated, all signal names are provided from the perspective of the Master.

Introduction

High latency memories such as DDR, DDR2 and DDR3 present particular difficulties when interfacing to minimal latency buses such as WISHBONE. WISHBONE permits multiple cycles of latency in the Classic single read and write cycles by stalling the assertion of the ACK signal. This is the method used by Andrey Popelo in the OpenRISC XSoC project.[3] However this approach is highly inefficient for high speed sequential memory accessing. For example, a DDR2 memory with CL=3 requires a minimum of 17 clock cycles between the address being presented and the data being available, however each subsequent data accessed requires only one more clock cycle. In Classic mode every access is separate and must endure the long delay.

Using the WISHBONE Registered Feedback Incrementing Burst Cycle would be ideal because a single latency would be spread over the total number of accesses, increasing total throughput. But according to the WISHBONE specification, the ACK signal specifies both an acknowledge to the Master to proceed with subsequent addresses and that valid data is

1 http://www.xilinx.com/support/documentation/ip_documentation/ug086.pdf
2 http://opencores.org/opencores,wishbone
3 Project available at http://pm.stu.cn.ua/projects/show/xsoc in the Repository path root/trunk/rtl/stu/ddr2mig_to_wb/

available for the Master to receive. This dual meaning cannot be satisfied when interfacing with high latency memory bursts.

This project's aim is to overcome this issue by presenting two different methods of interfacing. The first method presented adheres to the letter of WISHBONE Registered Feedback Incrementing Burst Cycle specification, but in the author's opinion it violates the spirit of WISHBONE. The second method is a streamlined interface and more inline with the way WISHBONE approaches bus transactions, but does not adhere to any already specified mode of operation. It is the hope of the author that the WISHBONE committee will consider the second method for inclusion in the specification as a new Registered Feedback Bus Cycle mode of operation.

WISHBONE Registered Feedback Incrementing Burst Cycle

At first glance it would seem impossible to use the WISHBONE Registered Feedback Incrementing Burst Cycle for interfacing with high latency memory without modifying the way the cycle works. But the WISHBONE specification provides Tags which may be used to provide information on data being presented on their respective array. These tags can be used to adapt the high latency cycle to WISHBONE by “filling in the blanks,” that is to provide in enough address requests to receive the requested data. These address requests are actually bogus, or dummy requests, there to keep the connection open long enough to receive the requested data. Also the data returned by the Slave is dummy data until the actual requested data is available.

To make this scheme work, additional tags are specified to provide the necessary information.

- The Master needs to indicate to the Slave the boundary between valid addresses and the dummy addresses. This address tag, TGA0_O may be called ADRV (Address Valid).
- The Slave needs to indicate to the Master the boundary between dummy data and the requested data, this Slave data tag TGD0_I may be called DATV (Data Valid).

One final complication which arises from this method is the Master needs to provide an End of Burst, or 111 on CTI_O the cycle before the last requested data is provided by the Slave. This can be accomplished by the Master using a counter and counting how many DATV cycles have occurred, and comparing to the total requested. Alternatively the Slave can indicate to the Master the cycle before the End of Burst cycle that the next cycle should be the End of Burst. In this situation a third signal needs to be part of the interface, TGD1_I or EBNC (End Burst Next Cycle).

The following is the waveform for the WISHBONE compliant Registered Feedback Incrementing Burst Cycle.

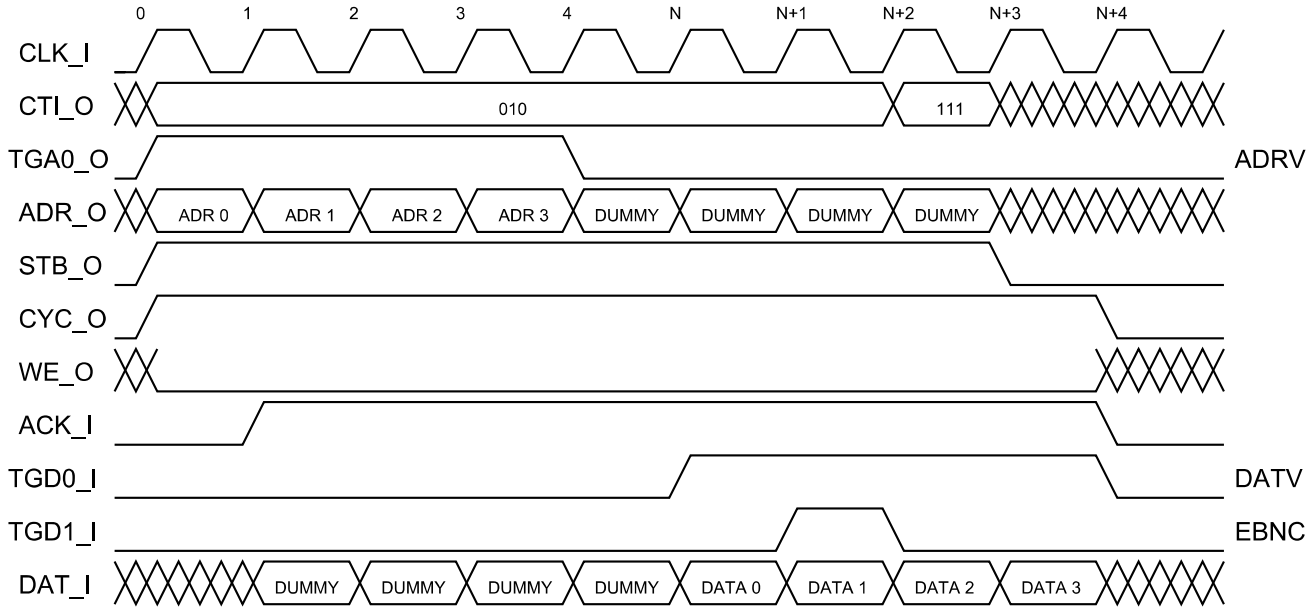

Figure 1: High Latency WISHBONE Registered Feedback Incrementing Burst Cycle

All signals specified in the WISHBONE Registered Feedback Incrementing Burst Cycle are adhered to in Figure 1. For the reader's convenience, the cycles are numbered. Note the cycle labeled N and the subsequent cycles with an offset from N, this is to indicate that cycle 4 may be repeatedly executed. All control signals should remain constant during this repeated state. This state is exited when TGD0_I (DATV) is asserted.

WISHBONE Registered Feedback High Latency Incrementing Burst Cycle

The solution presented in the previous section is adequate and WISHBONE compliant, but is inelegant and contains extra information not needed to execute the data transfer. If WISHBONE's single cycle latency requirement for burst operations was relaxed, a much more elegant solution could be created. The proposed new cycle is presented in the following figure.

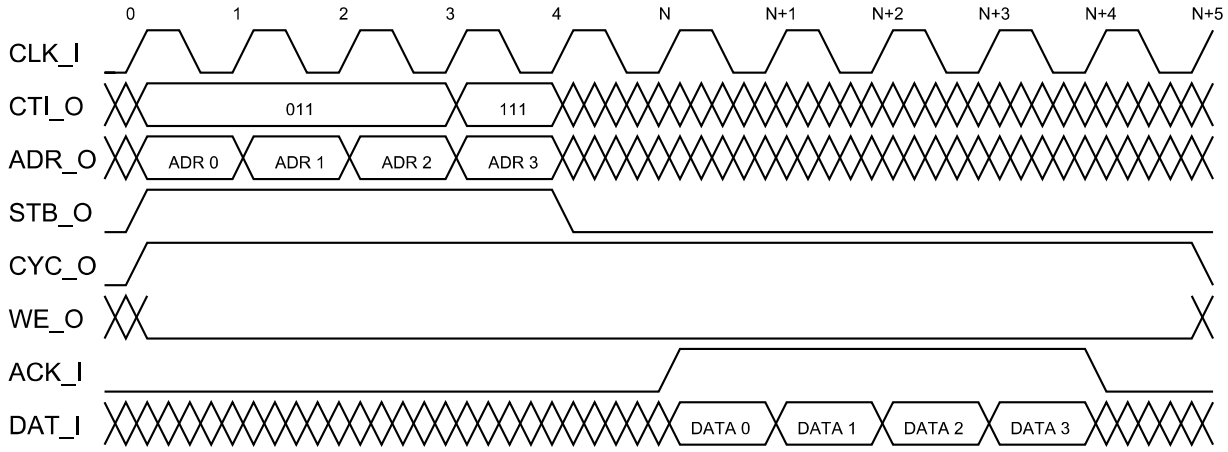

Figure 2: WISHBONE Registered Feedback High Latency Incrementing Burst Cycle

Notice that all tags have been removed, they are not necessary because the CTI_O timing has been restored to indicate the end of the actual burst, and ACK_I once again indicates when valid data is present on DAT_I. ADR_O and DAT_I appear different in Figure 1 than in Figure 2, but the DUMMY cycles in Figure 1 are identical to the don't care cycles in Figure 2. The CTI_O value of 011 was chosen for continuity with the other Cycle Type Identifiers.

CYC_O holds the burst cycle open so that the high latency memory has time to respond, and lasts one cycle longer than in Figure 1. This extra cycle is what the author refers to as Lazy Cycle Termination; after the last data is transferred, the Slave deasserts ACK_I and the Master responds by deasserting CYC_O which ends the burst cycle. If the Slave cannot complete the requested burst, it asserts ERR. It is possible to implement a non-Lazy Cycle Termination (or Standard Cycle Termination) and regain the cycle by using the counter method in the previous solution in either the Master or the Slave (requires a TGD to signal the Master). Lazy requires fewer resources to implement, but Standard may be desired when throughput is important.

If the Slave is unable to receive addresses at the rate the Master is transferring, it is able to assert STALL or RTY. STALL is useful to pause the Master when the Slave's address queue is full. When the memory device is being auto-refreshed the Slave can either assert the blocking STALL, or non-blocking RTY depending on the desired behavior. RTY should be asserted in response to the first address presented by the Master, STALL should be asserted after the Slave's address queue has been filled.

In this revision of the proposal, the Master must be able to accept the data at the provided rate to avoid data-loss. This is not an onerous requirement since the Master dictates the amount of data transferred and can design a suitable interface to handle the influx of data.

Additional considerations

There are limitations to how many memory locations can be accessed in a single burst, a request for refresh may occur during a burst, if the burst continues for too long without providing the MIG with a refresh request the memory may loose data. Also accessing different rows in a single burst is not possible, the Slave may assert ERR when requested addresses are not in the same row, or it may latch the column and bank from the first address and optionally alert the Master the returned data is not from the same column or bank as the request. The memory dictates which BTE mode is available and is by default not selectable by the Master. It is possible to implement a reordering buffer to provide a selectable BTE to the Master, however this will not be implemented.

Many different possible configurations have been proposed, the minimal resources implementation is Lazy Cycle Termination, RTY during memory refresh, and Column and Bank latching without out-of-row alert. The highest throughput implementation is Standard Cycle Termination and STALL during memory refresh, the method of handling out of row requests does not directly affect throughput.

Comments and recommendations are welcome.
Next update(s) will include State Diagrams, Architecture Diagrams and Verilog files.