**DDR Configure**

**Clock Generate**

**Data Capture**

**DDR**

**Controller**

**DDR**

**R/W\_BURST**

**DDR**

**Refresh/PWD**

**Output Alignment**

**Data In**

**Clock**

**Addr**

**Data**

**Ctrl Signals**

**Captured Clock**

**Diff Clock**

**Cmd**

**Data In/Out**

From Memory

Manager Unit

To DDR4 Memory

**Rdy**

DDR4 Memory Controller (MemCtrl) Architecture

The architecture is to define set of the inputs, outputs and main block functions for simplified MemCtrl (we target x8 address mode, read,write, refresh )

1. Inputs/Outputs to Memory Management unit: The DDR4 MemCtrl will receive/output the following signals from Memory Controller unit:

* Input Clock Signal.
* Memory Address for read/write
* Write Data
* Control Signals: Read, Write request, Reset, Number of bytes for Read or Write.
* Rdy: It is a handshake signals to MMU to not receive any request while it is busy. So one request as a time.

1. Input/ Outputs: The DDR4 MemCtrl will output the following signals to DDR4 Memory:

* Differential Clock Signals
* Commands: groups of signals, i.g CKE (clock enable), CS\_n, ACT\_n and Bank Groups, Bank Address, Columns Address, etc..
* Data In/Out: Output data for write, Input data for read.

1. Main Functions:
2. **Clock Capture:** Capture Input Clock Signal and generates a differential clocks to DDR4 (and probably few synchronous clock with different phase shift for Output Alignment Block)

The positive clock (e.g Captured Clock) in a pair of differential output clock will be controlled clock for the entire design.

The Differential Clocks will be same frequency of Input Clock.

Either use #forever … or instantiate a primitive component DCM or PLL, both are easy. (if we need more synchronous clock with different phase-shift then DCM or PLL is more convenience)

1. **Data Capture:** To synchronize the inputs to design clock domain. All input signals (data, address, controls signals) are captured as rising edge of Captured Clock
2. **DDR Controller:** The simple FSM is to arbitrate between Configuration, Read/Write or Pre-fresh processes (based on the captured control signal from Data Capture Block and a timer to periodically activate Pre-fresh)

* The block also to control the timing between the update of Mode Registers

1. **DDR Configuration:** Implement the FSM for sequence of commands of Configuration DDR4 device (section 3.3 in standard)
2. **DDR Pre-fresh/Power down:** Similar DDR Configuration block but for Re-fresh or Power down.
3. **DDR R/W Burst:** Implement the FSM for sequence of commands, timing between the commands, data, recover time(precharge) and strobe outputs for read and write burst (if the FSM is too complex then will break into two: one for read and one for write)

For example of the write request, the block must generate the Active Command by setting CS, CKE, ACT, bank address, bank group, row address, then assert wait state for tbd number of clock cycles the generate Write Command by setting CS, CKE, ACT, bank address, bank group, col address, then asserts wait state for tbd number of clock cycles, the asserts 8-bit data on the bus as center alignment of output clock until finish all the write data, then assert wait state for tbd number of clock cycles for recover time before starting a next read/or write request. Not to be precise, but the idea.

* There is a function to mapping from physical address to topological address (bank grp, addr, row, col). The mapping from physical address to topology is not important and various from one architecture to another. We just simple define eg. MSB bits is bank group, 2nd msb bits is bank address, etc…). Refer to section 2.7 in the Jedec standard for the size of each address components.

1. **Output alignment:** Based on the outputs of the DDR Configuration, Pre-Fresh, R/W Burst, the block will align output address and data to the output differential clocks and strobe outputs. There are different modes: center and egde-aligned to the rising clock for read and write request.

Configure the Data Bus as IO bus.

1. **Data In:** Just to capture the read data from DDR4 Memory