

5 de marzo de 2021

Resumen

1. DRAM notes

This document is intended to be an a simple introduction of how the SDRAM works, look at the main signals of the protocol, explain a little bit how Xilinx FPGA tackles that interface and show how to set the Xilinx MIG.

The basic structure of the DRAMs is just a transistor with a capacitor. The idea is as follows: When the address of coincidence the transistor gets selected and the word line contains either a high or low value which is the bit that is store in the capacitor. To read the procedure is similar, if the address coincide the transistor then a logic element reads the charge inside the capacitor. Note that every time that you read you are erasing the data from the capacitor, so if you change the address that you are reading you have to re-write the data that you read. Also the capacitor gets discharged over the time, so there should be some logic that refresh each value, that's why this types of memory are called dynamic.

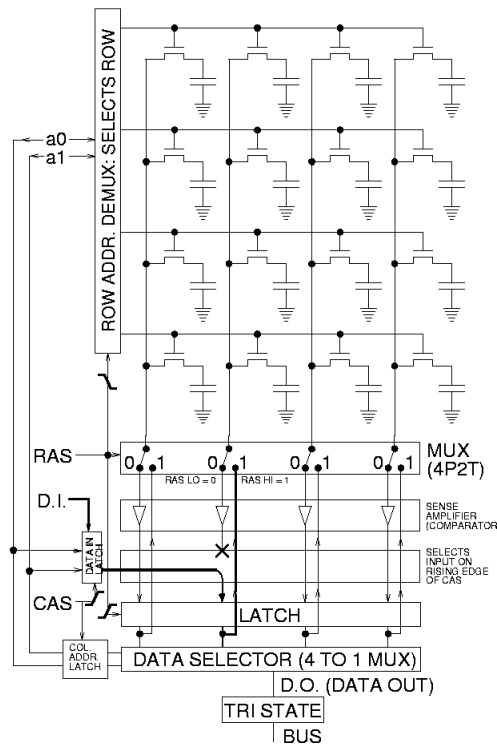


Figura 1: Basic DRAM diagram

The diagram of the DRAM in 1 consists in two indices, the row and the column. When you select a row the data gets read and charged into a row buffer, then you read the data using the column address.

The idea is that if you later try to read a column in the same row you don't have to go to the memory itself to obtain the data, it already in the buffer.

There are three main commands to read or write values in the DRAM:

- **Activate:** Open the selected row and place it into the row buffer.
- **Read/Write:** Read or write the column value.
- **Precharge:** Re-write the data into the column.

If the row is already open you just have to read/write the data, if there is a request for a closed row you have to pay and write the data into the row, then move the data from the correspondent address to the row buffer and then you could read it. So in this case the best is to open a row a write/read the columns in this row before change to another one.

2. DRAM Hierarchy

The DRAM has a hierarchical architecture:

- Dual in line memory module (DIMM) is the typical RAM module in computers that everybody has in his mind. Most DIMMs are built using "x4" or "x8" memory chips with nine or eight chips per side where the number refers to the data width of the DRAM chips in bits. For example x4 means that the data width per side is 36 in each side.

There are a lot of variations of the DIMMs, like for example the SODIMM which is a Small Outline DIMM.

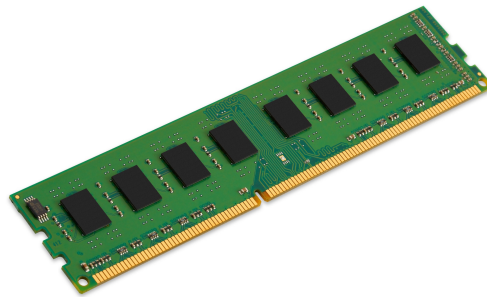


Figura 2: DIMM example

- Rank: This is a way to address the data in the DIMM, for example if we have x4 with 9 DRAMs per side you have the 36 bits. The controller has to access to 72 bits per transaction, so when issuing a command reads or writes both sides of the DIMM in that case the DIMM is single rank. If the DIMM is x8 then we have the 72 bits per side, so we only have to address one side. This is a dual rank, and we have to include in the address to which rank we are issuing the command.


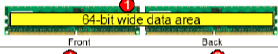
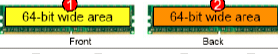

Single-sided module		1 rank
Double-sided module		1 rank
Double-sided module		2 ranks
Double-sided module		4 ranks

Figura 3: DRAM rank examples

- Chips in a bank: Each rank is composed by several chips. For example the x8 is composed by 8 chips per rank. Each one of those chips receives 1 byte of

data, so for example if the data is 64 bits the first chip receive data[0:7], the second one data[8:15],... and the last one data[56:63].

- Banks: Each of the chips has banks which is a third dimension in the figure 1. That means we have several repetition of the structure of rows and columns and in the address we have to select which one we want to read or write.

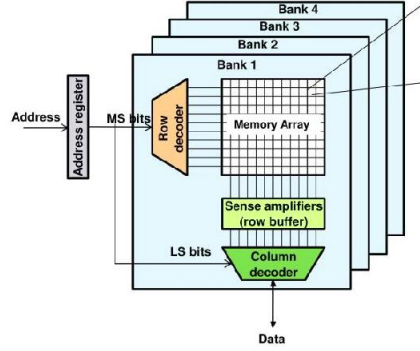


Figura 4: Bank example

- Finally we get into the row-column structure we first present.

After all that, the architecture could be summarize in figure

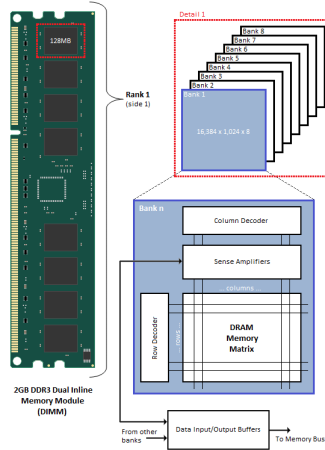


Figura 5: DRAM topology.

In the memory interface generator you have to give as inputs, among other things, the target type of memory, the column, row and bank numbers, if the rank uses 8 or 9 bits (the 9th bit is for correction), etc.

3. Signals

Before starting is good to say that the DRAM uses a protocol with the differential signals standard in order to use both clock edges (that's why is called double rate). Usually the data is sent using several lanes at high data rate, when is received the data is parallelized to be process in a lower clock rate. For example a 2n prefetch means that internally the system is running at the half of the transfer rate but has a bus with the double size.

Said that here are the main signals (to see all look at the **JEDEC STANDARD DDR3 SDRAM: JESD79-3C**) as a notation the # means the inverse signal:

- CK, CK#: clock signals.
- CKE: clock enable.
- CS: Chip select. This signal enables to select the rank if is not a single rank.
- RAS, CAS, WE: this signals names are an heritage from SRAM. Here define the type of command.
- DM: data mask. As its names suggest mask the valid data.
- BA0-BA2: Bank address input.
- A0-A15: Address input, in active command is the row address. In read/write commands is the column address.
- DQ: data input/output.
- DQS, DQS#: data strobe.

The data strobes signals are bi directional signals that are emitted by the master of the bus, in the write mode is send by the controller (in our case the FPGA) in the read mode is send by the DRAM. It has three stages and are meant to be flags for the devices, like “brace yourself I’m coming with a bunch of data”.

- Preamble: It's a time window where the sender gives a time windw for the receiver to get ready for the transfer.
- Toggling: When the data starts to be transfer the stroe starts to toggle between low and high at the clock frequency while the data burst last.
- Postamble: After the data burst is transfer there is a time window where the bus is in low.

There is a little (big) detail about this strobes signals. In read transaction is edge aligned with the data, so it changes are made in the rising edge of the bus clock. **BUT** for the write transaction this strobe signal are center aligned, which means that the strobe is running with 90° phase difference with the data channel.

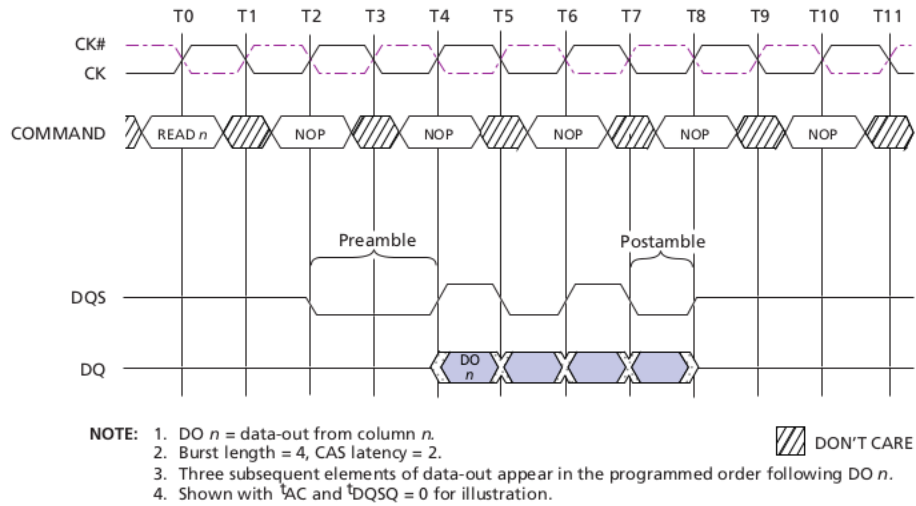


Figure 6: Read transaction.

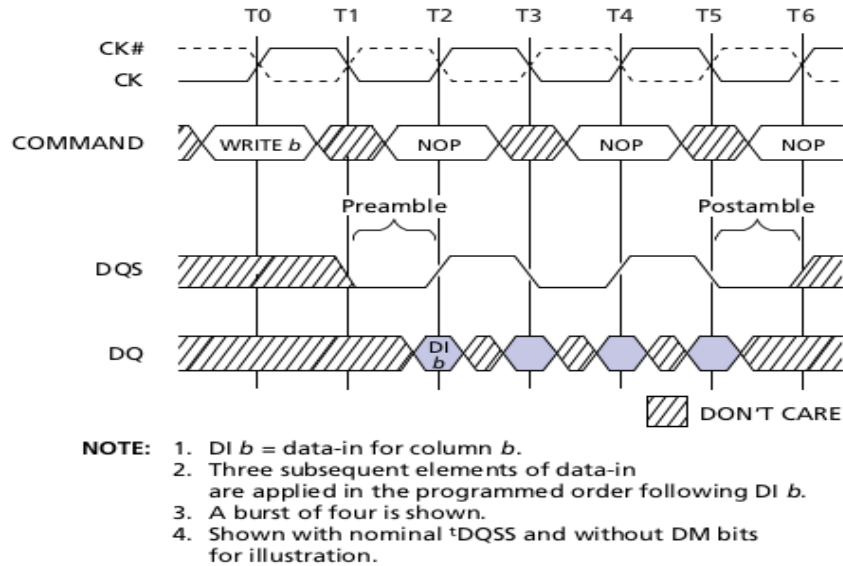


Figure 7: Write transaction.

4. Xilinx solution

Like you should be thinking the main problem here are the DQ, DQS signals. Because they change their phase with command that they are sending you can't get the signals with a simple PLL.

Also the DRAMs runs at much higher frequency that the FPGA itself could handle. So we have to make a commitment, we cant get the standard with the usual re-configurable hardware so the solution is to use some dedicated hardware for that propose.

In particular Xilinx uses Phasers wich are who generate the clocks with their correspondent reative phases, align the data also uses dome FIFOs to cross clock domains, order the data, etc. And to send and receive the data use OSERDES and ISERDES. Where Serdes are Serialize-Deserialize logic that generates serial data running a higher clock rates from lower rate parallel inputs.

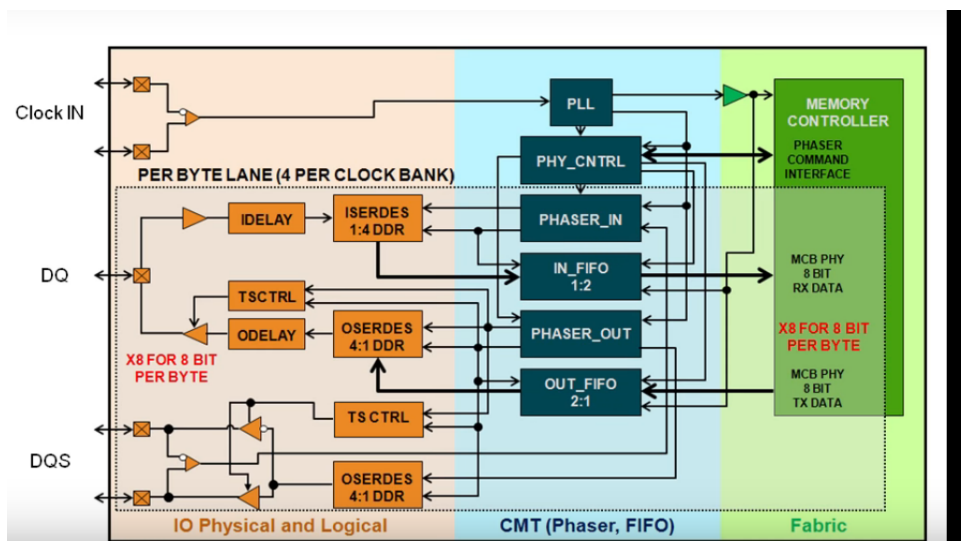


Figure 8: MIG block design.

Now we are entering to a dark place.. The physical interface that Xilinx give to us is really bad documented, if you want to use the IP directly there should be no problem but requires experience to program this things out of the IP.

For the DRAM we are fine because the DDR3 IP is free, but some times you would want to build your own interface to not paying for an IP but the physical part is not documented and you are stuck.. well that's life.

Besides the physical part (PHY) the Xilinx core generates a controller for the DRAM and get us a nice AXI interface so also hide the complexity of the addressing

and other stuffs to the user. At the end you should be able to target the addresses of the DRAM in the same way you write a AXI slave.

5. Using the Memory interface generator

With all that we had present you should be ready to use the MIG to build a custom part. If you use a supported board you should have a default settings and just have to issue the block automation.

Here we present how to configure the MT8JSF25664HDZ for a kintex-420t. This a supported DRAM and has its default values, but we are going to configure it using a the custom options jut to show how to do it.

The first 3 pages are straightforward, select **Create Design**, in the next page we select out device **xc7k420t-ffg90** and in the third page we mark the SDRAM3 controller.

The fourth page is where we configure the settings for the DRAM that we are entering.

For the clock period we have to take a look to the supported speed of the DRAM that we are using and the supported frequency of the board. The figure 9 shows the speeds supported for the DRAM. The frequency supported by the kintex could be found in the **Kintex-7 FPGAs Data Sheet:DC and AC Switching Characteristics**.

Table 3: Part Numbers and Timing Parameters – 2GB Modules

Base device: MT41J128M16, 1 2Gb DDR3 SDRAM

Part Number ²	Module Density	Configuration	Module Bandwidth	Memory Clock/ Data Rate	Clock Cycles (CL-'RCD-'RP)
MT8JSF25664HDZ-1G6__	2GB	256 Meg x 64	12.8 GB/s	1.25ns/1600 MT/s	11-11-11
MT8JSF25664HDZ-1G4__	2GB	256 Meg x 64	10.6 GB/s	1.5ns/1333 MT/s	9-9-9
MT8JSF25664HDZ-1G1__	2GB	256 Meg x 64	8.5 GB/s	1.87ns/1066 MT/s	7-7-7

Figure 9: MT8JSF25664HDZ datasheet caption

Table 18: Maximum Physical Interface (PHY) Rate for Memory Interfaces IP available with the Memory Interface Generator (FF and RF Packages)⁽¹⁾⁽²⁾

Memory Standard	I/O Bank Type	V _{CCAUX_IO}	Speed Grade						Units
			1.0V				0.95V	0.9V	
			-3	-2/-2LE	-1	-1M/-1LM/-1Q	-2LI	-2LE	
4:1 Memory Controllers									
DDR3	HP	2.0V	1866 ⁽³⁾	1866 ⁽³⁾	1600	1066	1600	1333	Mb/s
	HP	1.8V	1600	1333	1066	800	1333	1066	Mb/s
	HR	N/A	1066	1066	800	800	1066	800	Mb/s

Figure 10: Kintex datasheet caption.

As you may think we are going to use the 1:4 memory controller which means that the controller is running at 1/4 of the PHY frequency.

With that in mind we set the clock rate in 1.875ps which is equivalent to 533.33MHz. This means that we are operating at 1066 MHz, which is supported by the both devices. (By the way here we use an HR IO bank, that's why the VCCAux its blocked).

We set the memory type as **SODIMM** and then we click into the **Create Custom Part**.

This is going to open a new window where you have to fill the options with memory options. A tweak you need to know is that there is no visual option to select the rank, so if you are using a DRAM with double rank you have to select a double rank-DRAM as a base(I dont know if higher ranks are supported).

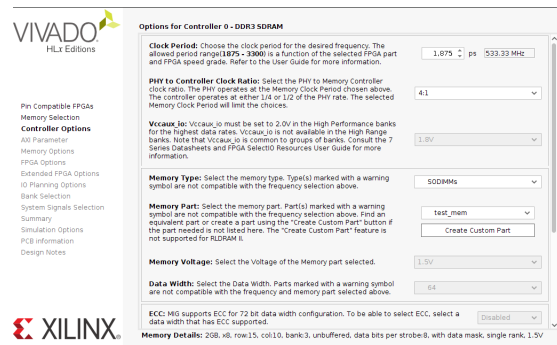


Figure 11: Controller options

You have to look at the DRAM datasheet to search for the timing parameters, for example the table 12 shows the trcd and trrd, but there are a lot of values missing in the datasheet :(. The official documentation of JEDEC has a large table with to calculate this parameters, but to have all the parameters you have to search for them across the entire document.

Table 1: Key Timing Parameters

Speed Grade	Industry Nomenclature	Data Rate (MT/s)							^t RCD (ns)	^t RP (ns)	^t RC (ns)
		CL = 11	CL = 10	CL = 9	CL = 8	CL = 7	CL = 6	CL = 5			
-1G6	PC3-12800	1600	1333	1333	1066	1066	800	667	13.125	13.125	48.125
-1G4	PC3-10600	—	1333	1333	1066	1066	800	667	13.125	13.125	49.125
-1G1	PC3-8500	—	—	—	1066	1066	800	667	13.125	13.125	50.625
-1G0	PC3-8500	—	—	—	1066	—	800	667	15	15	52.5
-80B	PC3-6400	—	—	—	—	—	800	667	15	15	52.5

Figure 12: Some timing parameter

I recommend using the templates of similar devices and use their default values and if the system doesn't work go back and modify this page. As we are using a supported device we are going to use the default time values that comes with this DRAM.

The other important features are the bank, row, column numbers of your device. The MT8JSF25664HDZ addressing are shown in the figure 13. So we have 15 row address, 10 column address and 3 bank address. In the last instance this values sets the size of the DRAM.

Table 2: Addressing

Parameter	1GB	2GB
Refresh count	8K	8K
Row address	16K A[13:0]	32K A[14:0]
Device bank address	8 BA[2:0]	8 BA[2:0]
Device configuration	1Gb (128 Meg x 8)	2Gb (256 Meg x 8)
Column address	1K A[9:0]	1K A[9:0]
Module rank address	1 50#	1 50#

Figura 13: MT8JSF25664HDZ addressing

You should check in the DRAM that the other values are right (for example your device has ECC ie it has 72 or 64 bits, or the rank is right?). Check the memory details in the bottom of the window.

The maximum value for the row address is 16 and for the column address is 11. With the 3 banks this give us 8GB, if you want to use a 16GB you have to use as template a dual rank memory.

We keep the bank machines in 4 and the ordering in Normal.

The next window is the AXI parameters. Put here whatever you want, I put 512 because usually the DRAM has 8 length burst of 64bits, so we have a data burst of $64 * 8 = 512$, but you could use different numbers.

In the next page, you need to give as input the FPGA clock that you are going to use to feed the MIG, in my case I set the value to 200MHz because I have a 100MHz on board clock, so I only need to multiply it by two factor. Here you could also select if the MIG is going to generate output clocks for the use of the FPGA, we are not going to use them.

You could scroll a little bit in this page and set to set the termination status, you need to search for the right type in the DRAM datasheet. For our case we keep everything as default but you should check the impedance of your device before moving on.

You could also select to disable the Chip select (CS) which only make sense for the single rank devices. The idea is that if you are designing a FPGA you could save

one pin, like we have a board with all the pins attached we don't care, we keep it enable.

The final option in this window is the memory addressing type that our DRAM has. To know that we need to look at the pin configuration in the DRAM datasheet, in specific the table in 13 tell us how the memory is addressed, the columns are first, followed by the rows and the banks are separated in a different signal group. We keep the default bank-row-column option. And click in the next button.

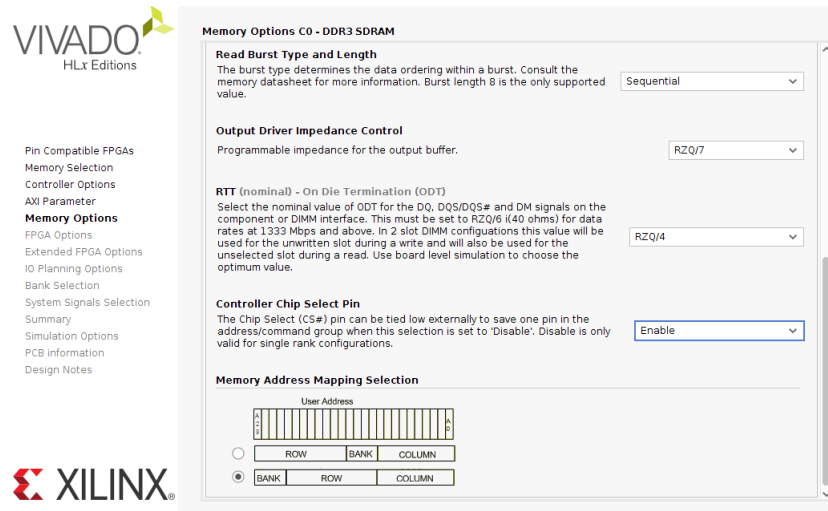


Figura 14: MIG memory options

The next page are about how to clock the system, like we are going to use an internal clock we set the system clock without any buffer and in the reference clock we select to **Use system clock**. Also we set the reset polarity to be active low, which is the typical standard.

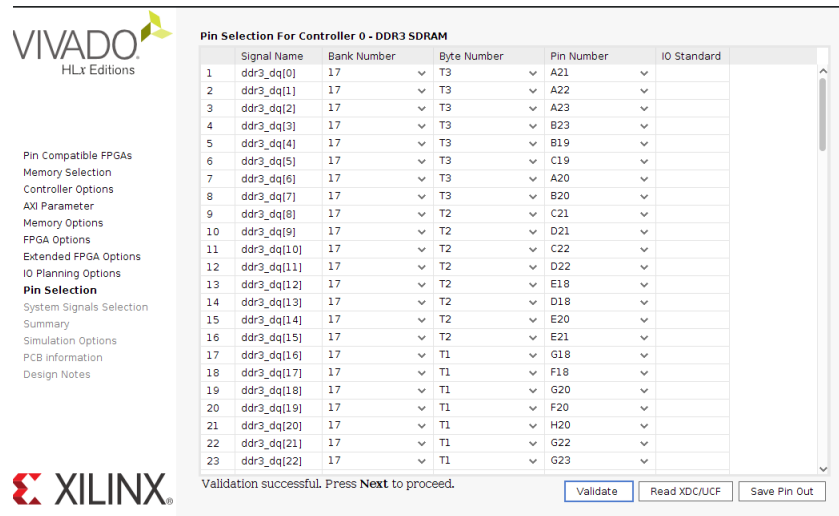
We keep unmark the Internal Vref because our board already handles the powering of the devices externally. Also we keep the power reduction option and the XADC instantiation (allows correction due thermal variations).

Click on next and now we set the internal impedance termination to be 50Ohm.

In the IO planning we have two options, to make a new design and to use an existing one. Again, like our board has fixed the SODIMM interface we select the Fixed pin out option. Now you have to map each pin with it correspondent task. The hard way to do it is to search for each pin in the schematic and map it to the each component, but your vendor should had gave you an xdc or ucf file with the pin assignment (some times its in the board files that you put in the Xilinx directory, if you dont have it Xilinx gives a templete of their chips, but you have to fill every pin with his right component). You could upload a file with the assignment, but note

that it needs to be feed with the exact name.

When you are ready press validate, if the engine doesnt complain press next.



VIVADO
HLx Editions

Pin Compatible FPGAs
Memory Selection
Controller Options
AXI Parameter
Memory Options
FPGA Options
Extended FPGA Options
IO Planning Options
Pin Selection
System Signals Selection
Summary
Simulation Options
PCB Information
Design Notes

Pin Selection for Controller 0 - DDR3 SDRAM

	Signal Name	Bank Number	Byte Number	Pin Number	IO Standard
1	ddr3_dq[0]	17	T3	A21	
2	ddr3_dq[1]	17	T3	A22	
3	ddr3_dq[2]	17	T3	A23	
4	ddr3_dq[3]	17	T3	B23	
5	ddr3_dq[4]	17	T3	B19	
6	ddr3_dq[5]	17	T3	C19	
7	ddr3_dq[6]	17	T3	A20	
8	ddr3_dq[7]	17	T3	B20	
9	ddr3_dq[8]	17	T2	C21	
10	ddr3_dq[9]	17	T2	D21	
11	ddr3_dq[10]	17	T2	C22	
12	ddr3_dq[11]	17	T2	D22	
13	ddr3_dq[12]	17	T2	E18	
14	ddr3_dq[13]	17	T2	D18	
15	ddr3_dq[14]	17	T2	E20	
16	ddr3_dq[15]	17	T2	E21	
17	ddr3_dq[16]	17	T1	G18	
18	ddr3_dq[17]	17	T1	F18	
19	ddr3_dq[18]	17	T1	G20	
20	ddr3_dq[19]	17	T1	F20	
21	ddr3_dq[20]	17	T1	H20	
22	ddr3_dq[21]	17	T1	G22	
23	ddr3_dq[22]	17	T1	G23	

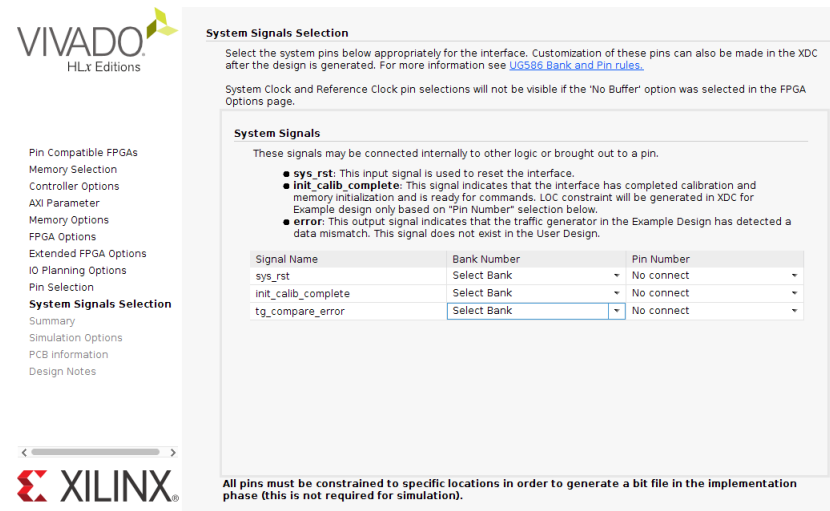
Validation successful. Press **Next** to proceed.

Validate Read XDC/UCF Save Pin Out

XILINX

Figura 15: Pin selection.

The System signals page has some signals that could be connected into a pin or be internally driven by some FPGA logic. We keep them unconnected.



VIVADO
HLx Editions

Pin Compatible FPGAs
Memory Selection
Controller Options
AXI Parameter
Memory Options
FPGA Options
Extended FPGA Options
IO Planning Options
Pin Selection
System Signals Selection
Summary
Simulation Options
PCB Information
Design Notes

System Signals Selection

Select the system pins below appropriately for the interface. Customization of these pins can also be made in the XDC after the design is generated. For more information see [UG386: Bank and Pin rules](#).

System Clock and Reference Clock pin selections will not be visible if the 'No Buffer' option was selected in the FPGA Options page.

System Signals

These signals may be connected internally to other logic or brought out to a pin.

- **sys_rst**: This input signal is used to reset the interface.
- **init_calib_complete**: This signal indicates that the interface has completed calibration and memory initialization and is ready for commands. LOC constraint will be generated in XDC for Example design only based on "Pin Number" selection below.
- **error**: This output signal indicates that the traffic generator in the Example Design has detected a data mismatch. This signal does not exist in the User Design.

Signal Name	Bank Number	Pin Number
sys_rst	Select Bank	No connect
init_calib_complete	Select Bank	No connect
tg_compare_error	Select Bank	No connect

All pins must be constrained to specific locations in order to generate a bit file in the implementation phase (this is not required for simulation).

XILINX

Figura 16: Signals selection

Now we are almost ready, click next and a summary should appear. Is always good to take a look, when you are sure that everything looks good press next and

keep pressing next until you see the generate button. Then and only then we are ready!

, , ”