Using the ECP5 for Libre-SOC prototyping

Verification of the Libre-SOC core
using an OrangeCrab feather form factor dev board
and gram to drive the ddr3 ram and booting Linux

FOSDEM2024

Sponsored by NLnet

January 31, 2024

Why using an ECP5 and OrangeCrab?

- ECP5 and ICE40 have libre toolchains (Yosys, nextpnr)
- Microwatt already supports the OrangeCrab using LiteDRAM:
  LiteDRAM depens on original Migen and does not fit into ls2
  I was unable to rebuild the original one from Microwatt
  and decided to continue working on (nMigen based) gram
- ECP5 is big enough for prototyping the Libre-SOC core
  When I started porting ls2 to the OrangeCrab I was able to
  run coldboot.c and began adding support for gram
- Because for 5 years I Always Wanted To Design A GPU
- ICE40 also used by Valve and Bitcraze in several SteamVR
  products and the Lighthouse positioning deck

Why nMigen and gram?

- nMigen+LambdaSoC port of the LiteDRAM core
  avoiding using old Migen and LiteX
  only using nMigen used by Libre-SOC
- nMigen is much more powerful than Verilog and VHDL
  and easier to use for anyone that knows python
  (and works with yosys/nextpnr/gcc)
- gram is simplified, it only supports ECP5 atm
  and maybe some Xilinx FPGAs in the future
- Wanting to learn how to use the dram phy that comes
  with the ECP5. Then booting linux using the Libre-SOC core

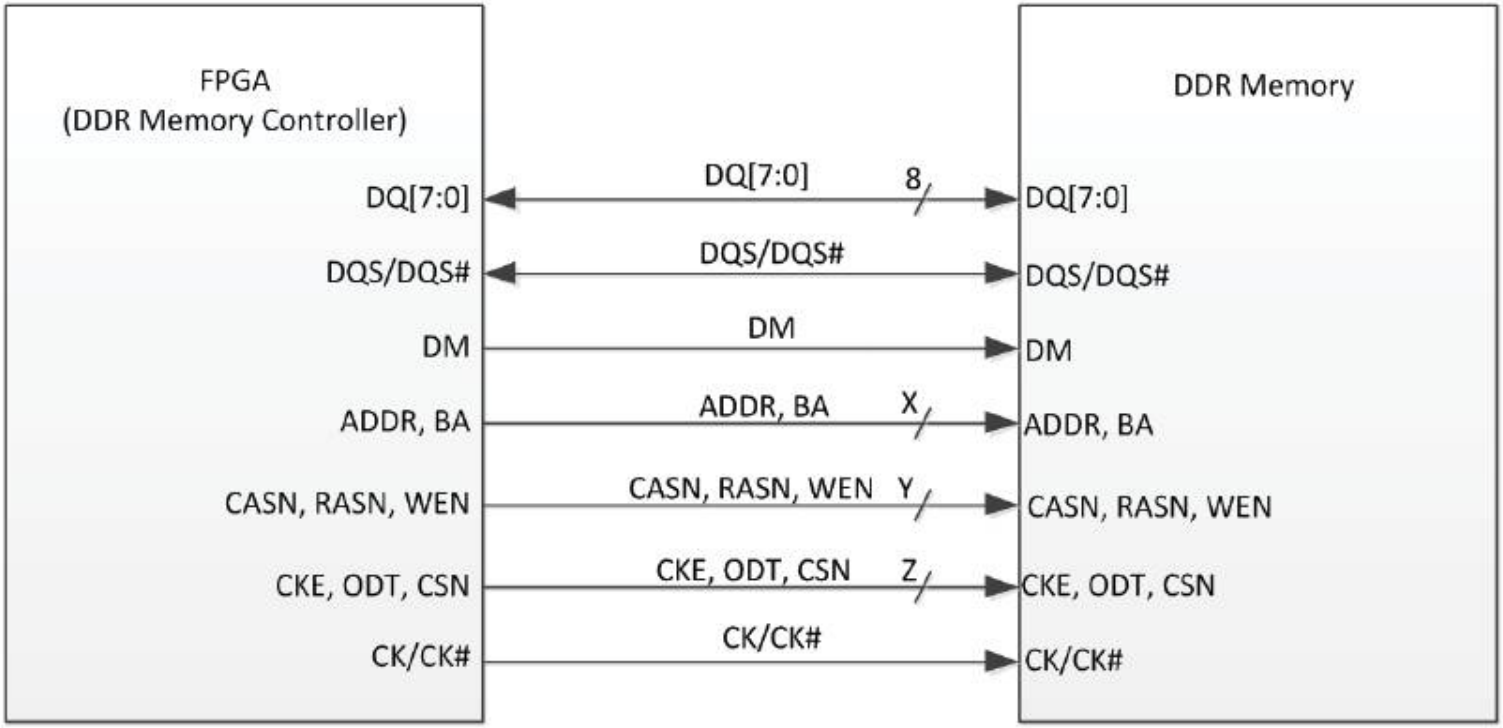
How do DRAM interfaces work

- Multiple generarions of DDR-SDRAM interfaces
  POWER9: DDR4-SDRAM POWER10: DDR5-SDRAM
- OrangeCrab has DDR3L Memory, 128 Mbytes (1Gbit)
  64M x16, 1.35V low voltage operation (can boot linux)
- See nmigen-boards for OrangeCrab pins
- Controller are found in gram and lite-dram
- Ideally gram and lite-dram should show identical behaviour
  Reality: only lite-dram works out of the box
- If it does not work debugging is hard
  Found out that burstdet signal is not asserted on read

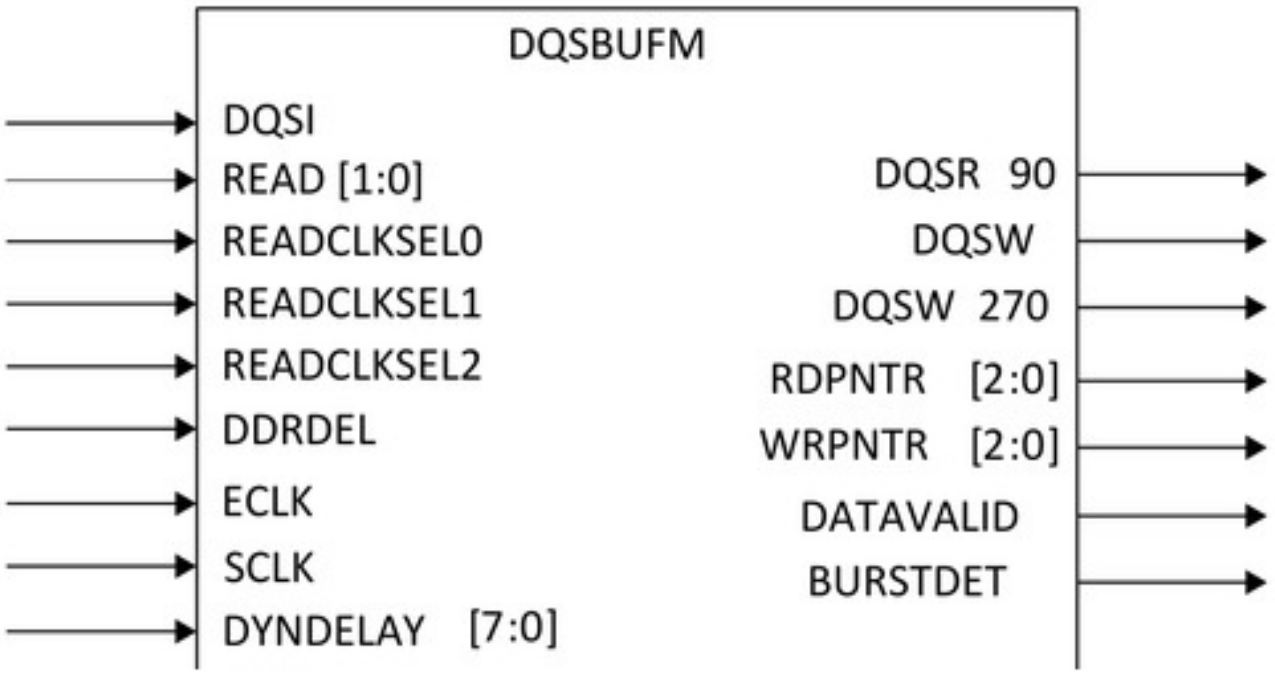
The ECP5 DRAM controller

- see manual for ECP5 and ECP5-5G High-Speed I/O Interface comes with many built-in blocks: DQSBUFM, DDRDLLA, ... *DQSBUFM* is the most interesting one
- FPGA-TN-02035-1.3 documents DQSBUF
  Data Strobe Control Block : Generates a BURSTDET output that can be used to validate the READ pulse positioning
- DQSBUFM element is used for all the DDR Memory interfaces on ECP5 only. Other blocks used on Xilinx
- Python implementation of DDR PHY for ECP5
  git.libre-soc.org / gram.git / gram / phy / ecp5ddrphy.py
  github.com / enjoy-digital / litedram / phy / ecp5ddrphy.py

Typical DDR2/DDR3/DDR3L Memory Interface

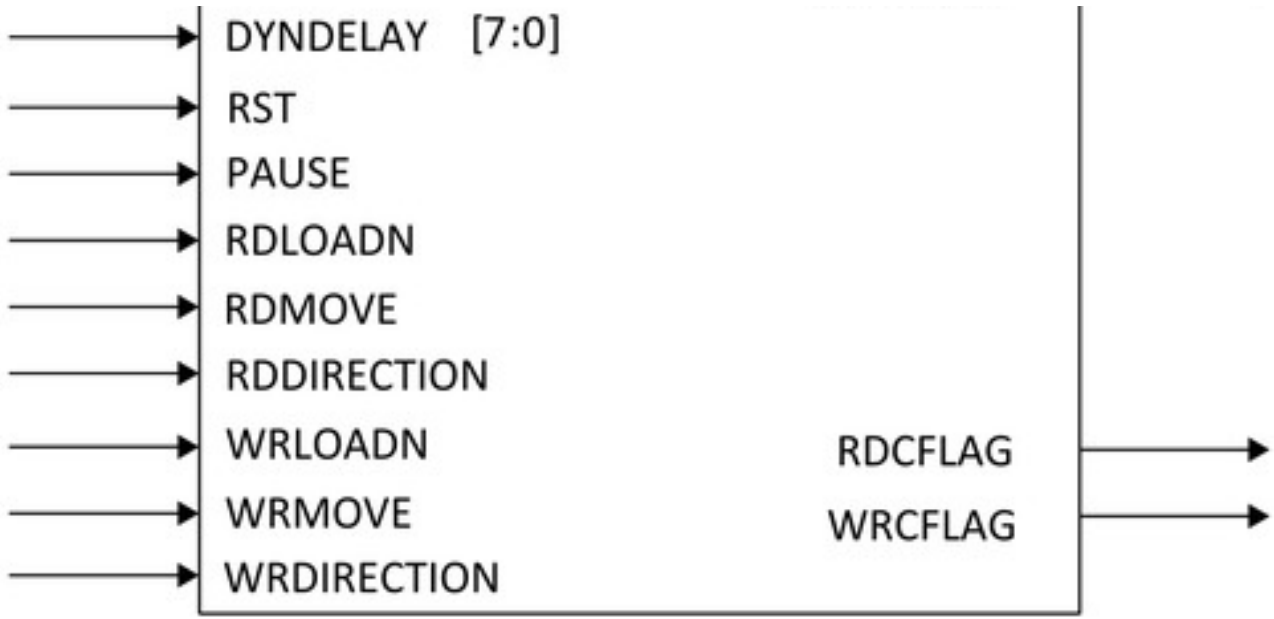


DQSBUFM – Inputs



- BURSTDET used for read leveling
- DATAVALID controls dfi.phases[0] and dfi.phases[1] latch

DQSBUFM – Outputs



- PAUSE controlled by init.pause and dqsbufm_manager.pause
- READCLKSEL used to set read leveling delay (3 bits)

How to use libgram

- struct gramCtx with base addresses and user data
- must call gram_init(ctx, profile, ddr_base, core_base, phy_base)

- first phy is set to software control
- init seq is peformed
- then calibration is loaded
- phy is set to hardware control back again
- finally memtest must run, we can boot linux if passed
- calibration is different from litedram

Read leveling in litedram

- only read leveling used by ECP5
- leveling must be done for each phy module
- inner loop for bitslip
- A test each read window (bitslip)
- B find min/max delays
- take bitslip with best score
- Re-do leveling on best read window
- Sync all DQSBUFM's before scan(A) / level(B)
- Live demo using Microwatt

Write/readback test under software control

- Generate pseudo-random sequence
- Activate test row
- Write pseudo-random sequence
- ECP5: reset burstdet for current module
- Read/Check pseudo-random sequence
- Precharge
- Read back test pattern
- Verify bytes matching current module
- ECP5: check burstdet for current module
- DQSBUFM's not synced if burstdet is 0
- not yet implemented in gram

Debugging with BeagleWire

- Fully Open iCE40 FPGA BeagleBone Cape
- 32 MB SDRAM and controller written in Verilog
- GPMC port access from the BeagleBone

- no port to nMigen yet, toolchain on BBB
- connect to ECP5 to exchange data with host PC
- use SPI or I2CSlave for debugging

- use iCE40/BBB as ROM emulator
- intercept read/write to SDRAM
- run gdbserver on BeagleBoneBlack

- Long term: make EOMA68 card with Solid Silicon X1
- Two FPGAs and 512MB of DDR3 RAM

Porting nMigen to BeagleWire

- There is only a LiteX port ATM
- Toolchain (yosys/arachne-pnr) running on BeagleBone
- Use python to control cross compile

- run yosys on host PC, replace arachne-pnr
- generate bitstring for multiple FPGAs if needed
- control flashing via ssh, upload bitstring via scp

- Unrelated to Libre-SOC: plans to make Lighthouse PMOD
- LibrePlanet 2022: The LibreVR Project
- http://librevr.isengaara.de

- VR Headset will depend on Libre-SOC GPU
- run DOOM on iCE40/ECP5 FPGA - VGA output needed

Summary

- Microwatt can boot linux+buildroot on the OrangeCrab using LiteDRAM. Libre-SOC using ls2 needs more work on gram, including porting to other FPGAs.
- High speed interfaces differ between FPGA models and vendors and are hard to debug without using a simulator.
- Software gets more complex for larger designs, development becomes more expensive. Changes easily break everything.
- Once DRAM is working we can add other interfaces.
- Larger FPGA may be needed to prototype advanced features including GPU und display controller
- We can use the BeagleWire as a host computer interface

The end

Thank you

Questions?

- Discussion: http://lists.libre-soc.org
- Freenode IRC #libre-soc
- http://libre-soc.org/
- http://nlnet.nl/PET
- https://libre-soc.org/nlnet/#faq