MICROELECTRONIC SYSTEMS DESIGN RESEARCH GROUP

Using gem5 for DRAM Exploration

Presented by: Éder F. Zulian

Microelectronic Systems Design Research Group
University of Kaiserslautern

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Outline

- **DRAM Evolution**
- Motivation for Memory Research
- Accurate and Fast Models are Needed
- SystemC / TLM2.0 coupling in gem5
- Gem5 Tips & Tricks

DRAM Evolution

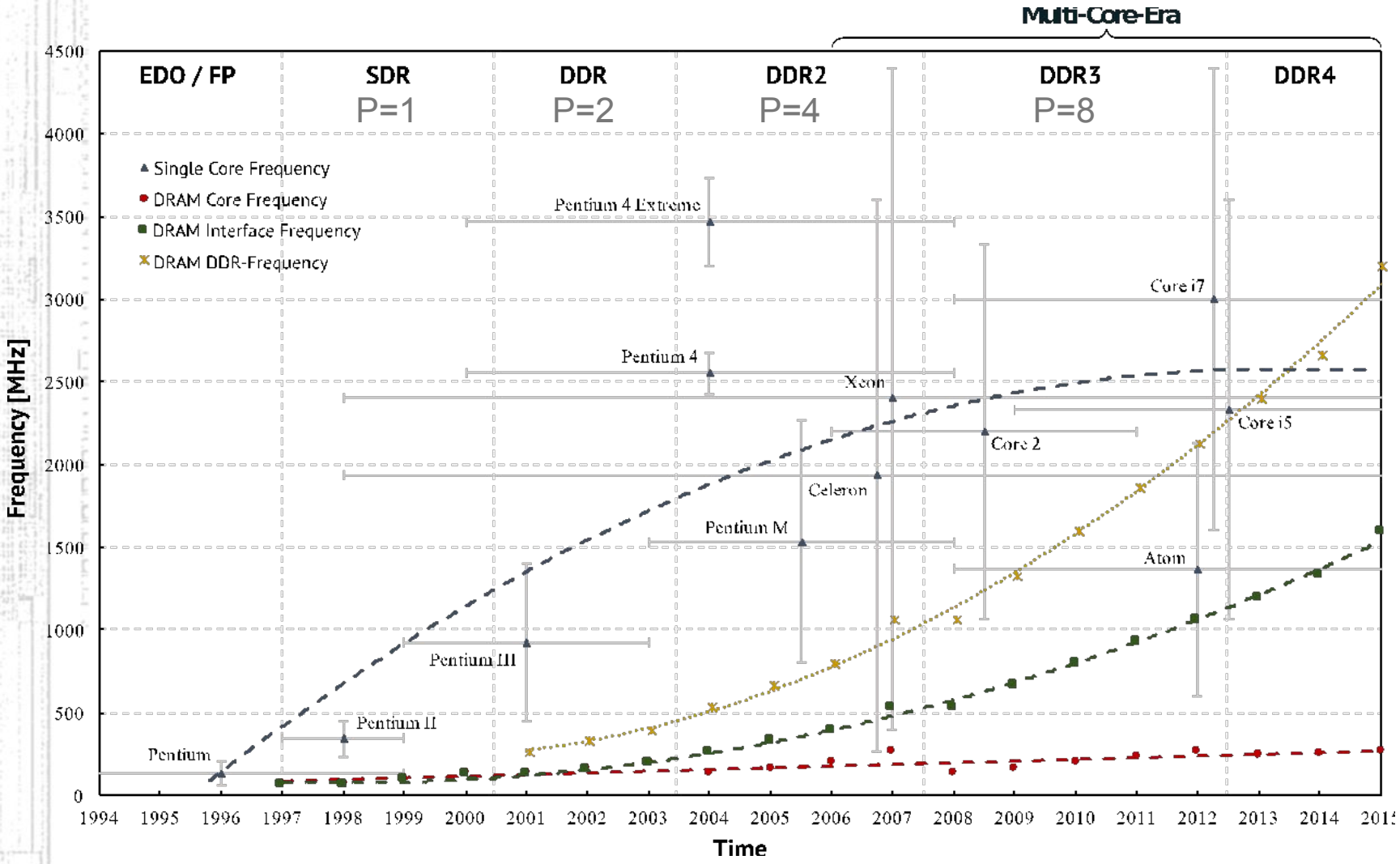
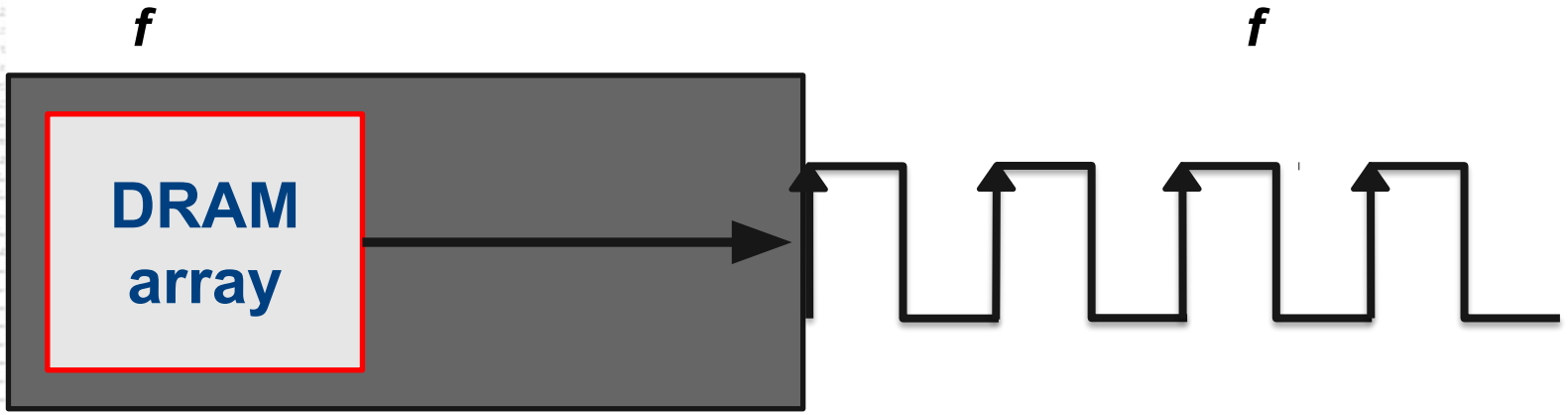


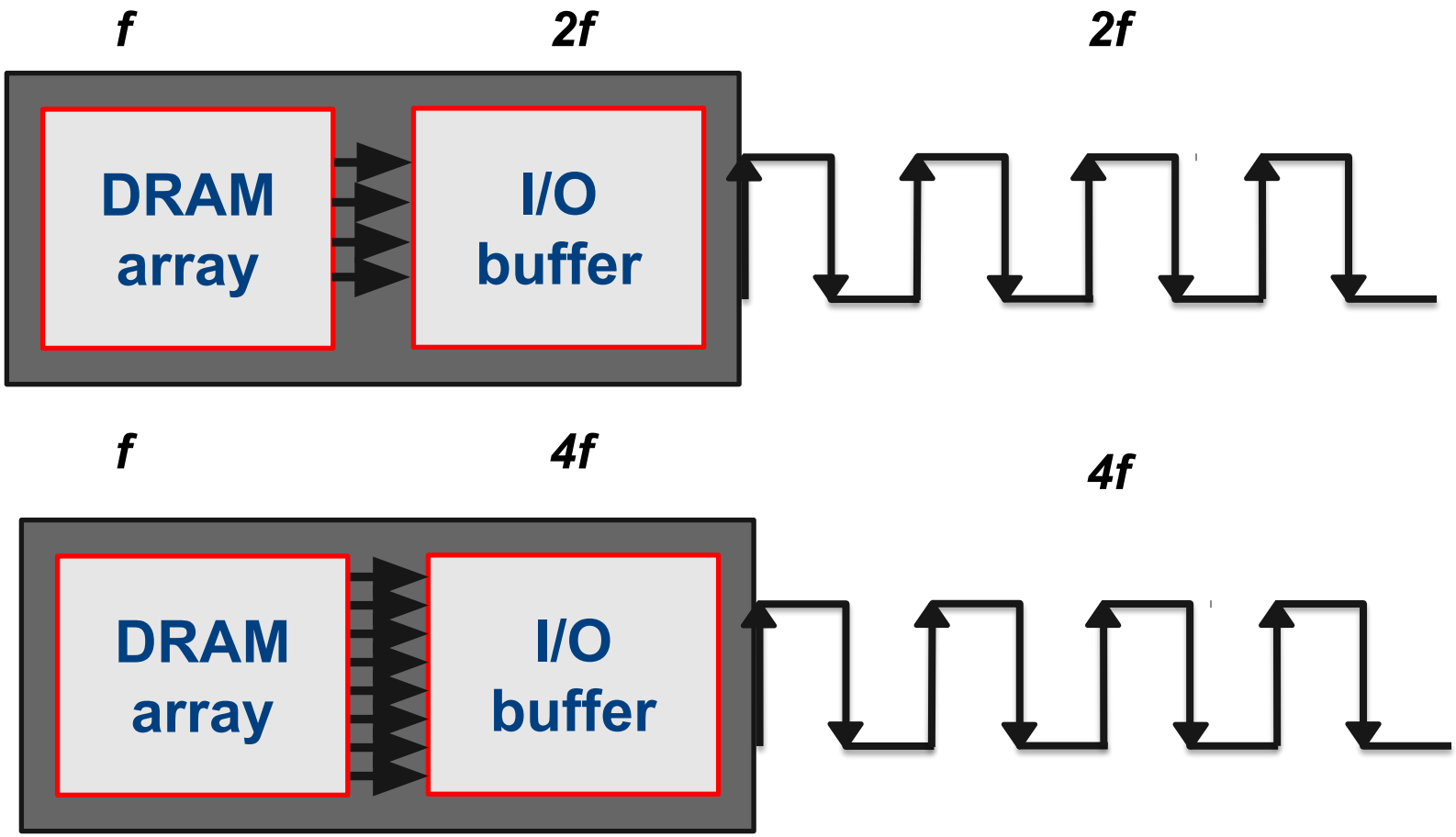
Image from Matthias Jung PhD Thesis

SDR and DDR



Based on: What Every Programmer Should Know About Memory, by Ulrich Drepper

DDR2, DDR3 and DDR4



$Power \propto f$ $Power \propto V^2$

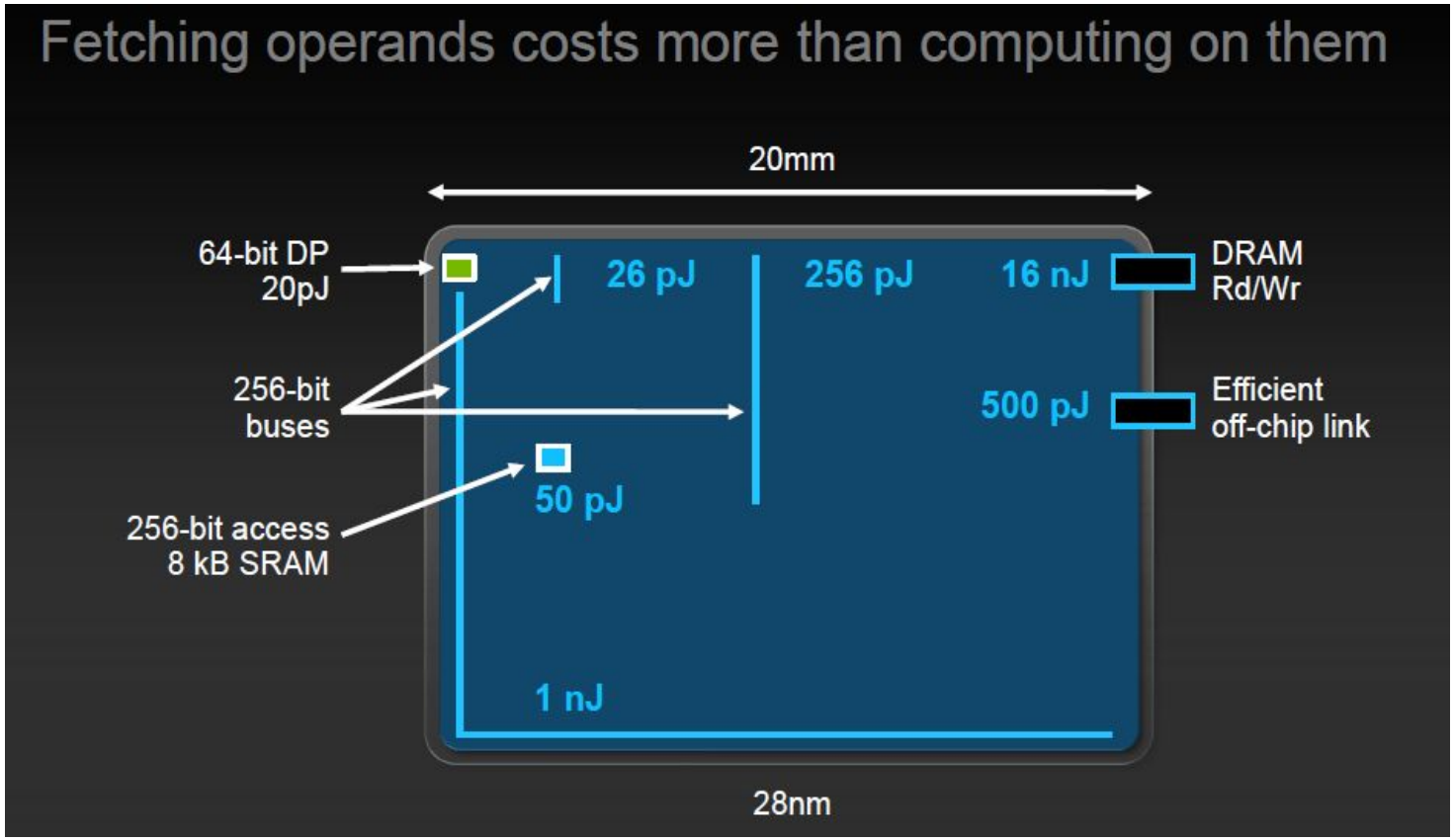
Outline

- DRAM Evolution
- **Motivation for Memory Research**
- Accurate and Fast Models are Needed
- SystemC / TLM2.0 coupling in gem5
- Gem5 Tips & Tricks

Motivation for Memory Research

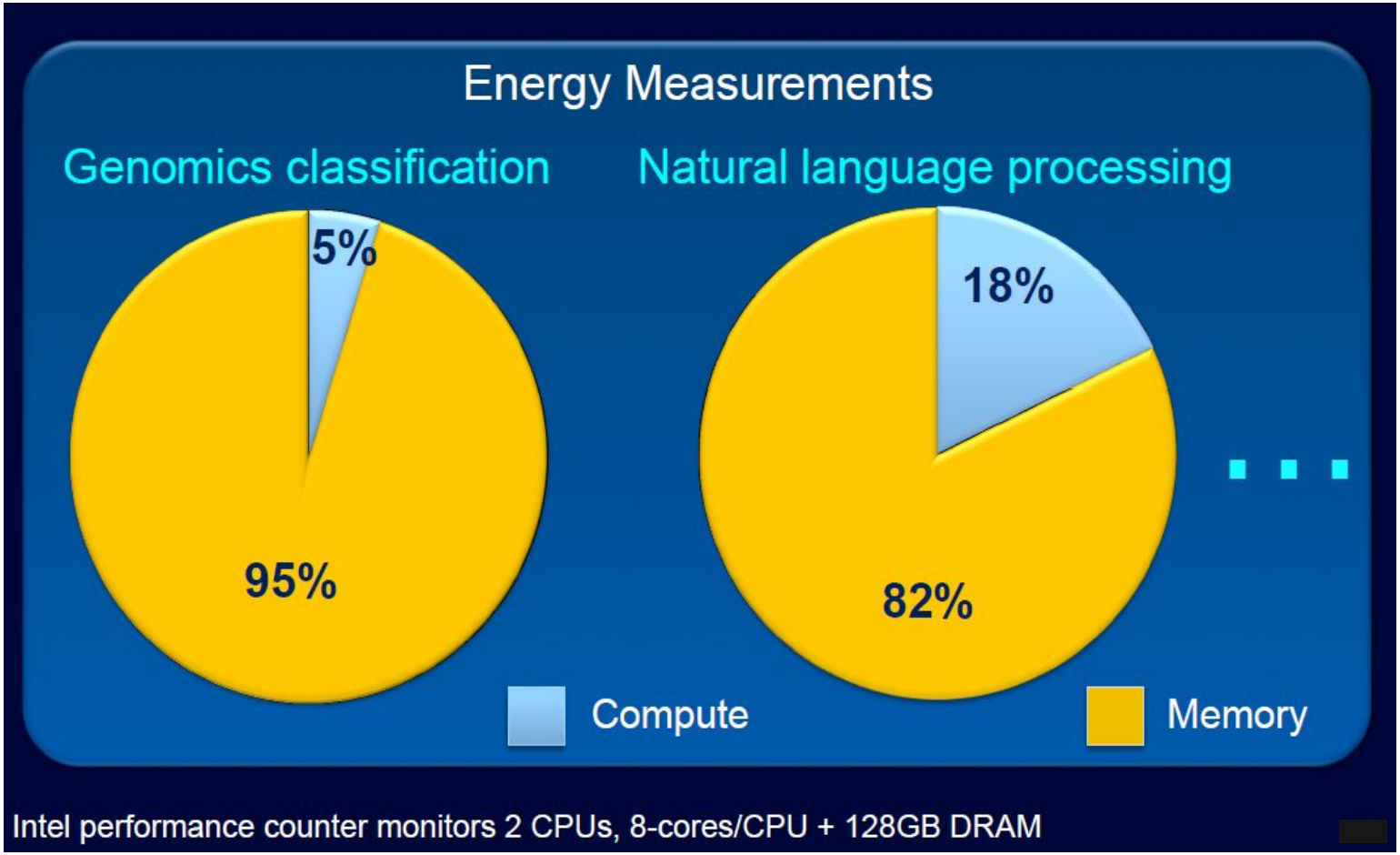
- Widening gap between memory and CPU speeds
- Multiprocessors continue to scale
- Higher communication demands within the same power budget
- Memory-bound programs do not scale with CPU speed improvements
- Performance of many applications are already memory-bounded
- Huge memory data sets
- Working set size (local and global)
- Several memory technologies, HMC, HBM, WIDE I/O, DDRx SDRAM, eDRAM, ...
- For the NVDIMM-P paradigm:
  - DRAM transparent to the host or visible to the host?
  - Challenges for host controller, DIMM controller, OS, ...

Moving data is Energy Expensive



Source: GPU Computing to exascale and beyond, Bill Dally slides - NVIDIA

DRAM Energy Contribution



Source: Subhasish Mitra

Outline

- DRAM Evolution
- Motivation for Memory Research
- **Accurate and Fast Models are Needed**
- SystemC / TLM2.0 coupling in gem5
- Gem5 Tips & Tricks

VP is key for memory exploration

Virtual prototypes enable us to get a good grasp of:

- Power usage profiles
- Energy efficiency
- Latency
- Sustained mem. bandwidth
- Bottlenecks
- Impact of architectural changes

Valuable tools make us agile.

gem5.org *www.doulos.com/knowhow/systemc* *www.uni-kl.de/3d-dram/tools*

DRAM Power Model on gem5

The DRAM Power Model on gem5 is implemented by DRAMPower

- DRAMPower is build as a library and linked to gem5
- Each Rank object has an instance of the DRAMPower object
- Mem. Specs are given to the constructor
- During operation DRAM commands are passed to the library
- Energy components (rd, rw, act, pre, ref,...) are returned together with the total energy for the current time window


```
$ ls ext/drampower
$ vim src/mem/dram_ctrl.hh
$ vim src/mem/dram_ctrl.cc
$ grep “power.powerlib” * -nrIil
```

www.es.ele.tue.nl/drampower

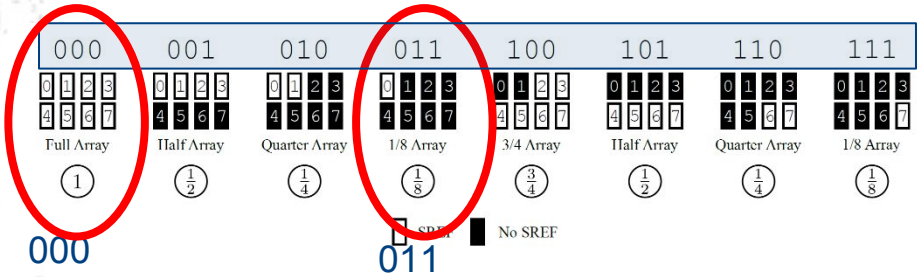
github.com/tukl-msd/DRAMPower

Coming soon to gem5!

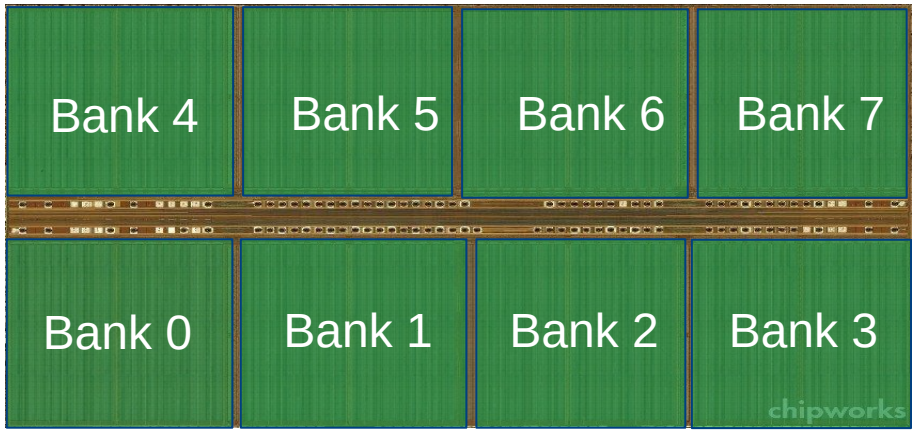
- Bankwise DRAMPower
  - DRAM banks are getting denser (e.g., DDR2 to DDR4 bank density increases from 0.5 Gb to 2 Gb. Similar trend for LPDDRx).
  - Per bank contribution to the total power is increasing. Chip level power estimations may get less accurate when per-bank features are used.
  - Required for modeling features like independent per-bank refresh and PASR.
- Temperature aware DRAMPower
  - Exponential increase in currents with temperature

E.g., Partial Array Self Refresh (PASR)

Self Refresh PASR Modes



Full Array 1 $1/8^{th}$ Array $\frac{1}{8}$



- Power down mode with maximum energy savings
- Complete DRAM is blocked from other commands
- DRAM manages refreshes internally
- DDR3 DRAM has 8 PASR modes

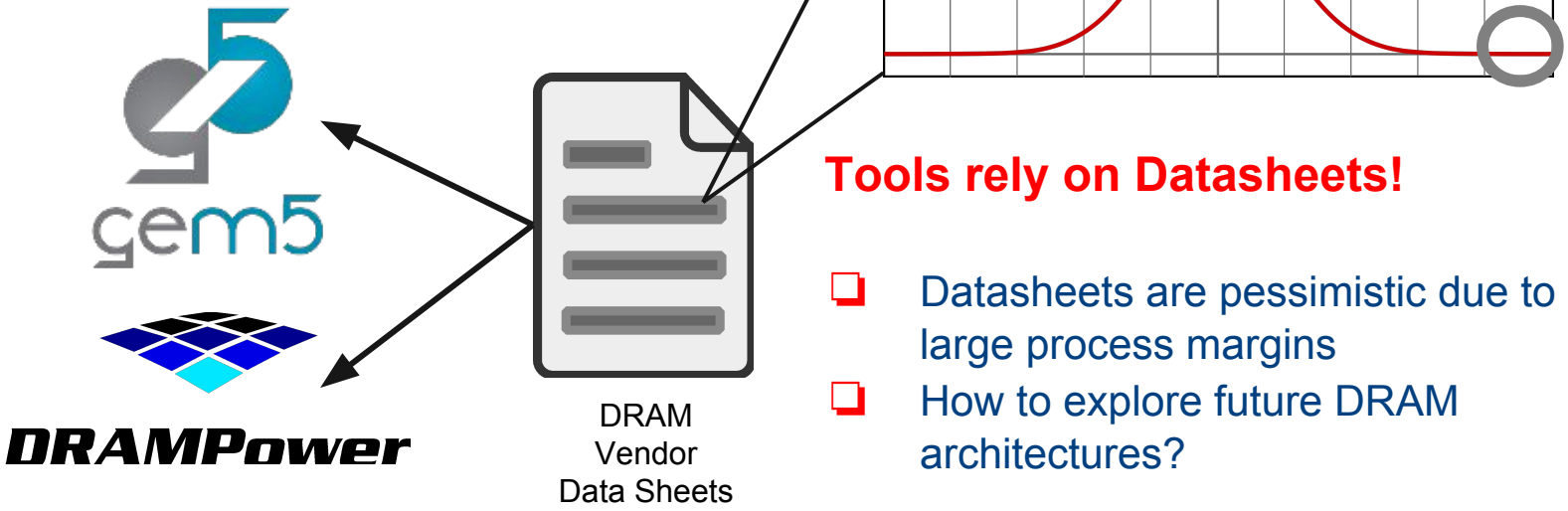
Courtesy Figure: chipworks.com

Memory research with gem5

What do we need to explore DRAMs?

- Realistic Workload which requires a full system simulator
- Realistic Controller Model
- DRAM power model

For Example:



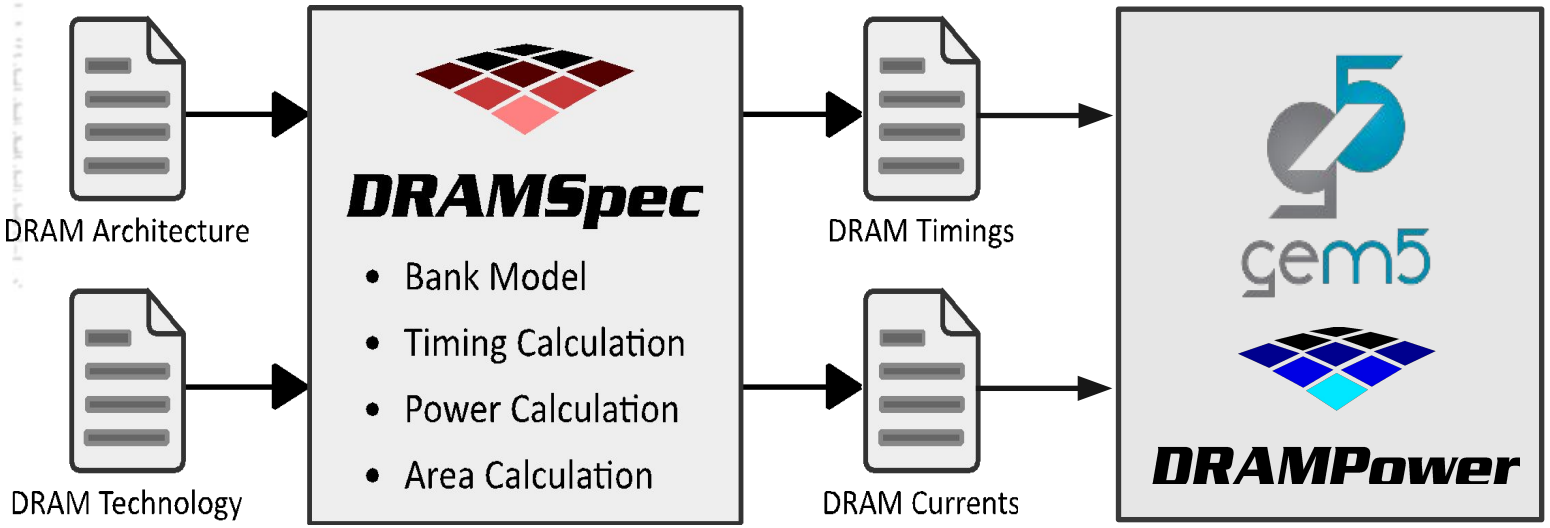
Tools rely on Datasheets!

- Datasheets are pessimistic due to large process margins
- How to explore future DRAM architectures?

Estimation of future DRAM devices

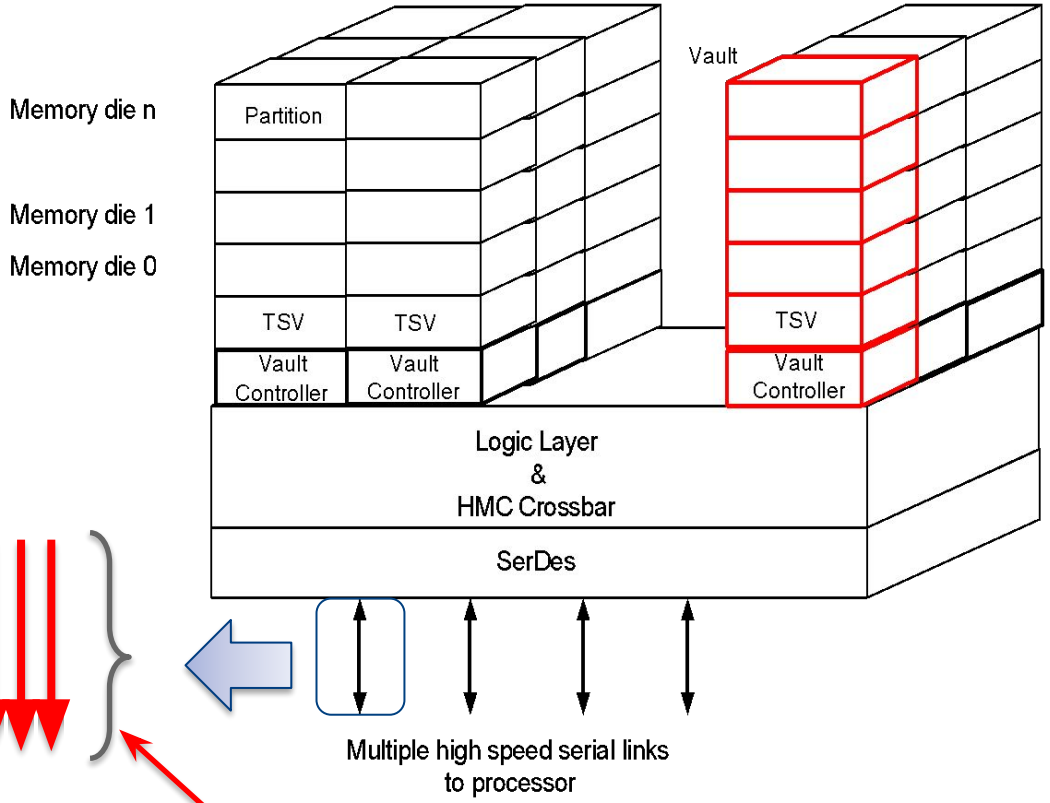
DRAMSpec generates datasheets for current and future DRAM devices

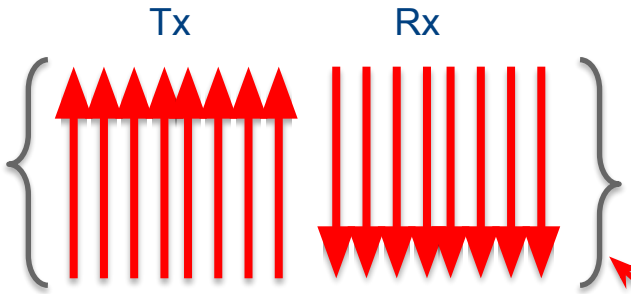
- DRAMSpec is verified against datasheets and measurements
- It can even be used by non-DRAM experts
- Very fast execution time compared to circuit level simulation
- Generate specs to DRAMPower and gem5 (e.g., HMC model)



Hybrid Memory Cube

- 3D stack of DRAM layers
- Logic layer at the base
- HMC crossbar for routing
- Per-vault controller
- TSV connecting dies
- Serial links connect HMC to host





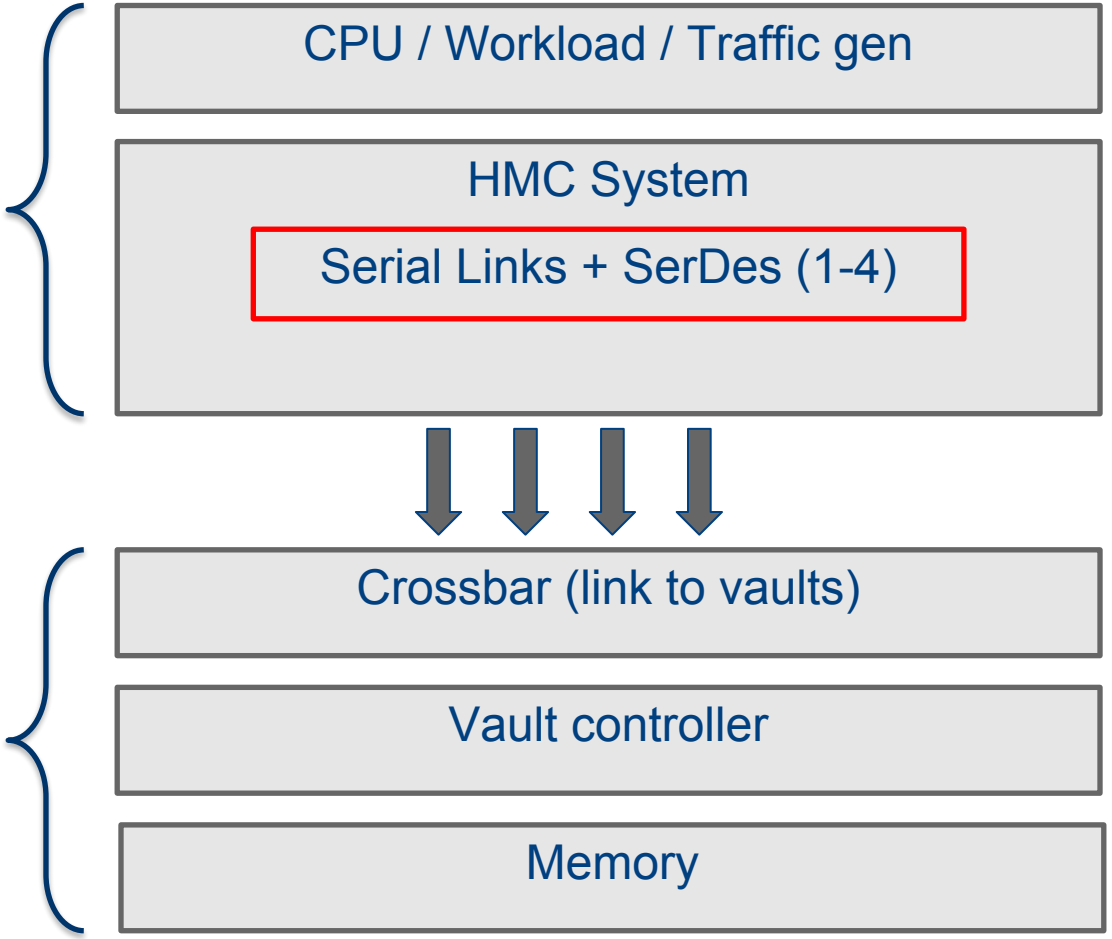
Each line working at Gpbs

Max. Bandwidth per link:
16 lanes x 2 x 10 Gbps
320 Gbps / link
(160 Gbps each direction)

HMC Simulation Model

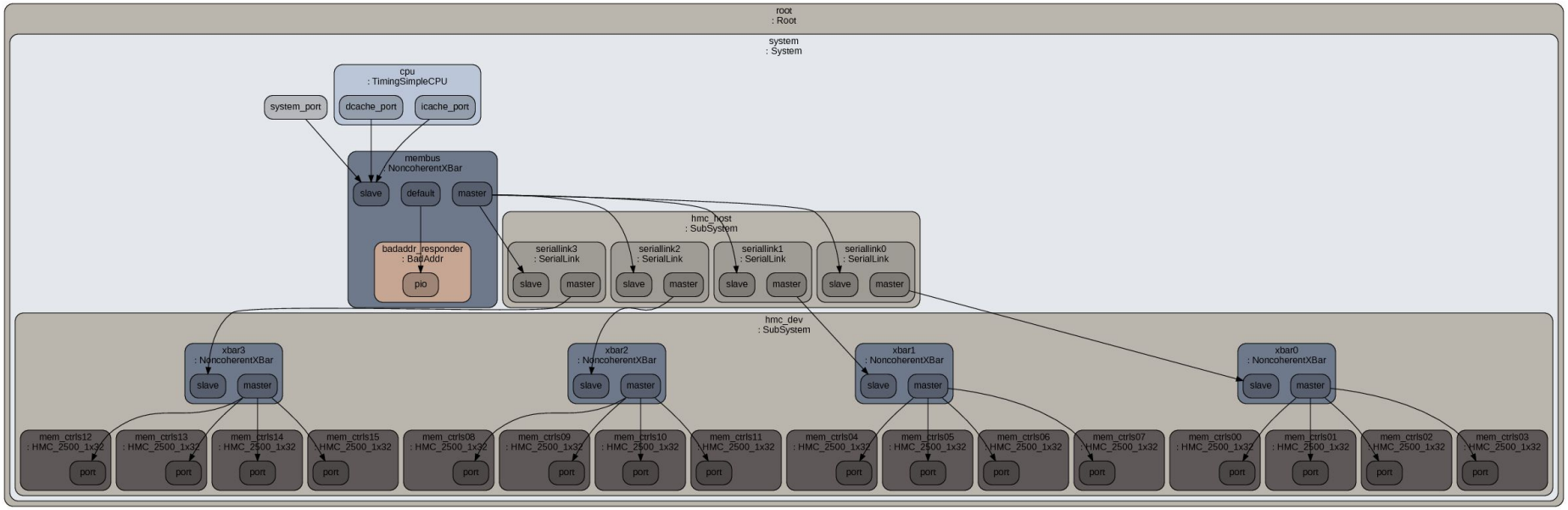
HMC Host side contains Traffic generator setup and model of serial links

Device contains crossbars vault controller and memory vaults



MICROELECTRONIC SYSTEMS DESIGN RESEARCH GROUP

HMC hello world on gem5



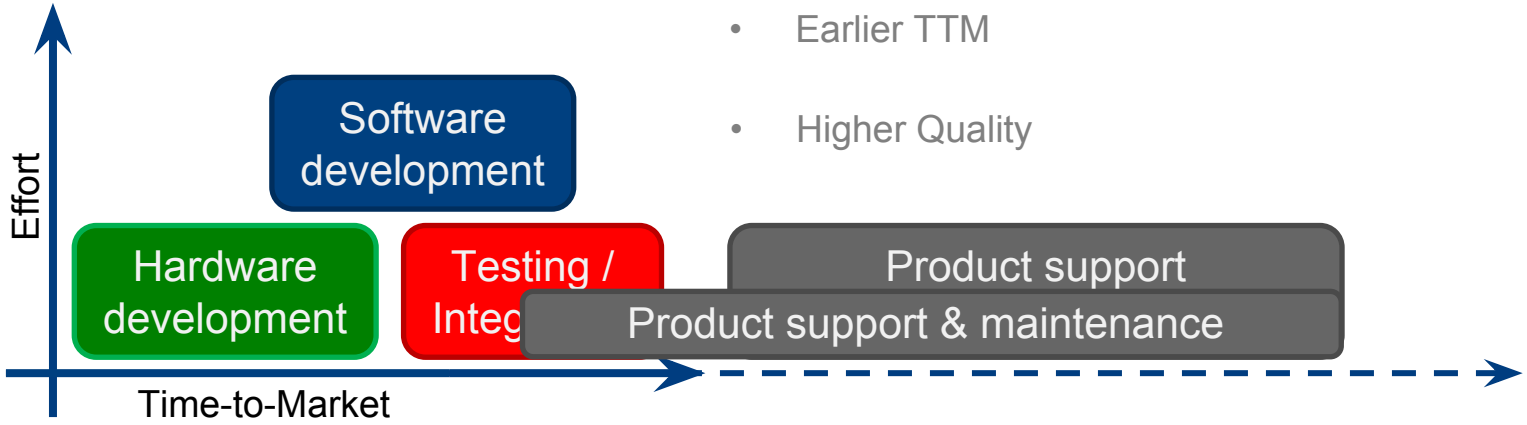

```
$ build/ARM/gem5.fast -d hmc_hello configs/example/hmc_hello.py
$ cd hmc_hello
$ okular config.dot.pdf
$ vim configs/example/hmc_hello.py
```

Outline

- DRAM Evolution
- Motivation for Memory Research
- Accurate and Fast Models are Needed
- **SystemC / TLM2.0 coupling in gem5**
- Gem5 Tips & Tricks

Virtual Prototypes In Industry

- High-speed functional software models of physical hardware
- Visibility and controllability over the entire system
- Powerful debugging and analysis tools
- Reuse of components for future projects
- Fast design space exploration (for HW engineers)
- Easy to exchange, worldwide
- **Concurrent HW and SW development:**

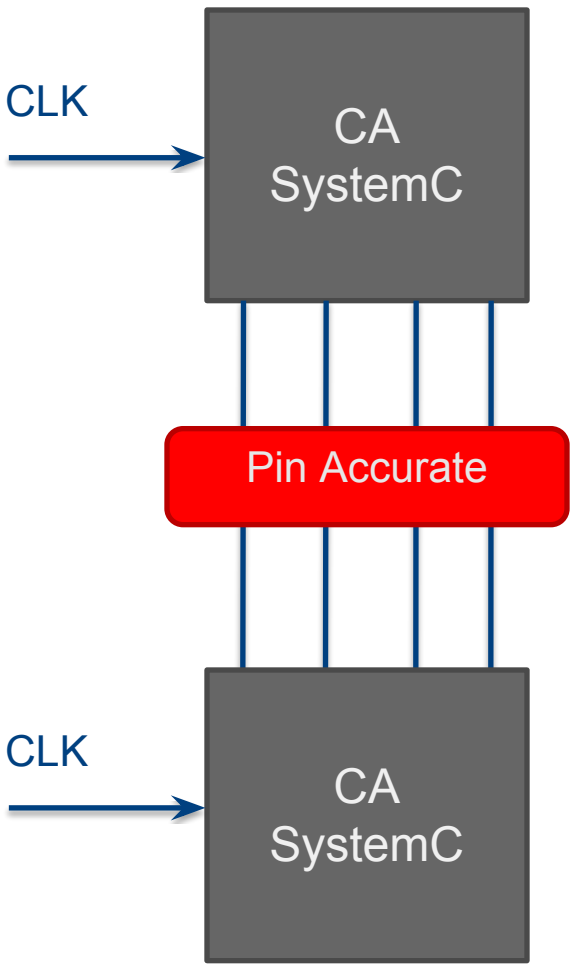


SystemC IEEE 1666

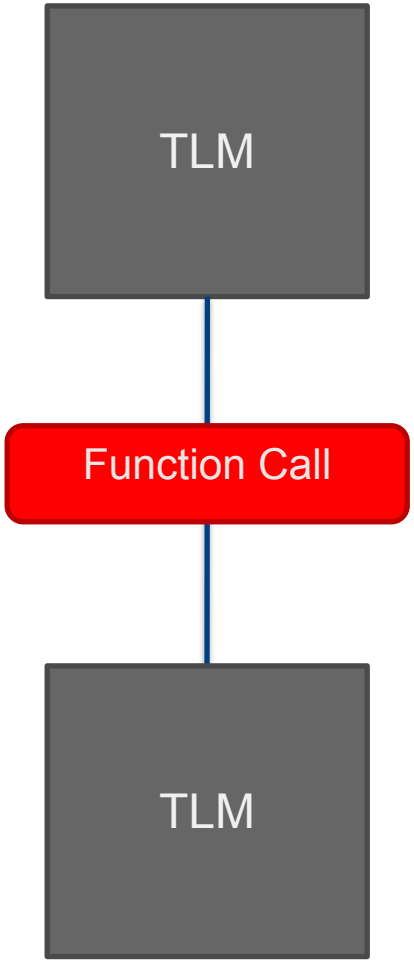
- Modeling language for HW and SW components
- Extends C++ to an event-driven simulation kernel
- Different levels of accuracy
- IEEE Standard, Maintained by Accellera
- 10-100x Faster than CA VHDL/Verilog Simulation

→**However, standard CA SystemC is not fast enough to boot, an operating system.**

Transaction Level Modeling



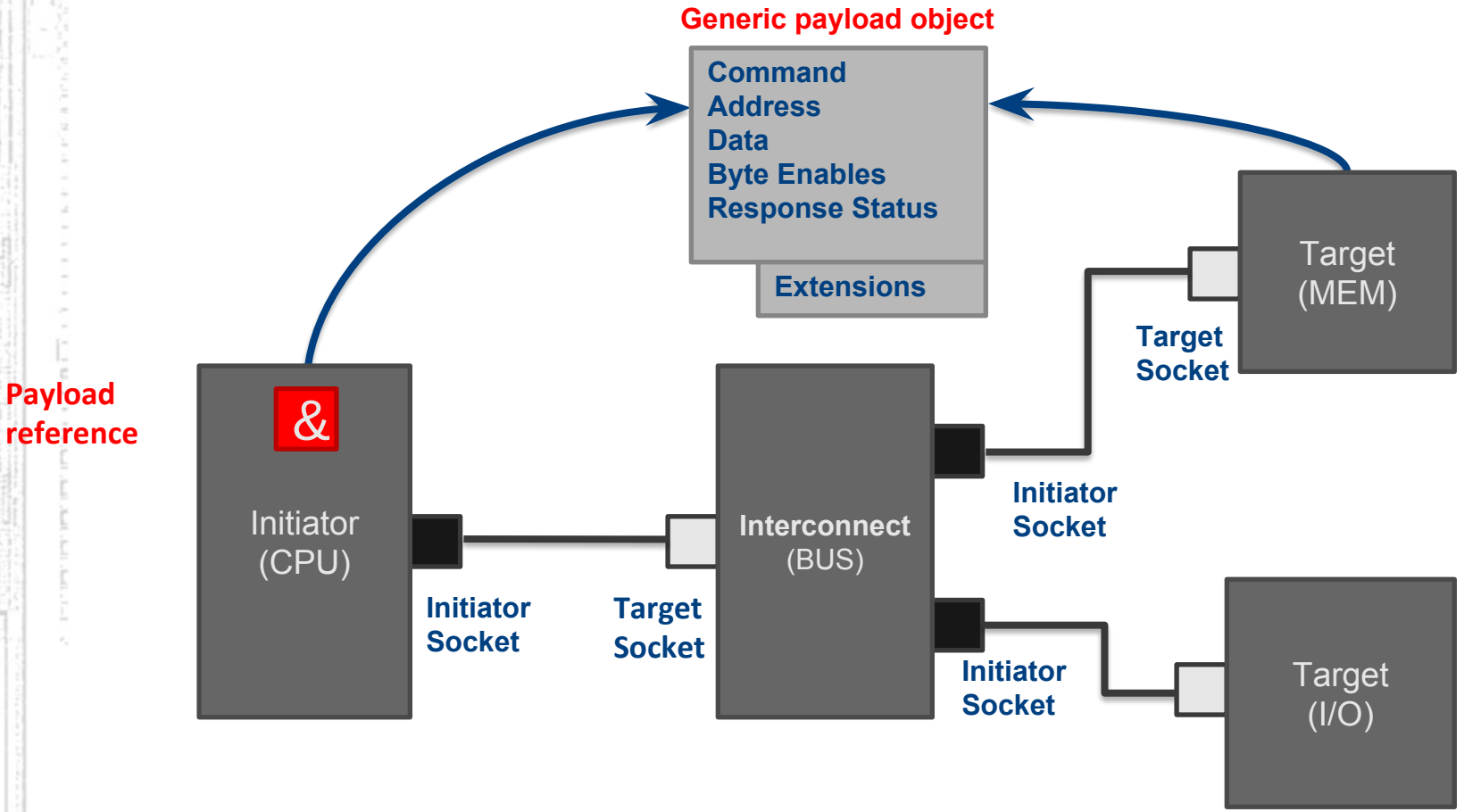
Simulate every event!



100-10,000 X faster simulation!

Source: Doulos website www.doulos.com

Generic Payload



Tool Vendors for TLM 2.0 VP

TLM is widely used in Industry:

The market of virtual platform tools:

- Synopsys - Platform Architect
- Cadence - Virtual System Platform
- Mentor Graphics - Vista Virtual prototyping
- Imperas - OpenVP
- ASTC - VLAB Works

Virtual Platform Core Models:

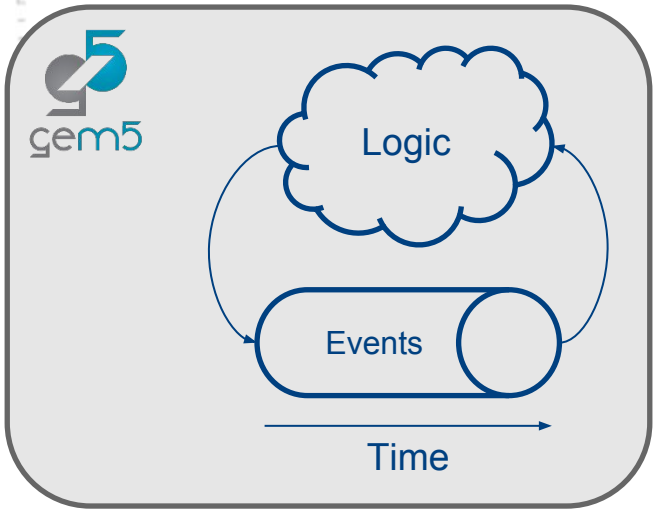
- ARM (Fastmodels):
  - only LT models based on JIT, non-free, library
- ARM Carbon (Former Carbon Design Systems):
  - Cycle Accurate (CA) Models in TLM Wrapper, non-free, library
- Imperas / OVP:
  - only LT, Free

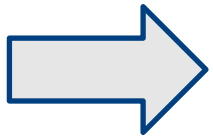
→ **An accurate, freely available and changeable core model is needed**

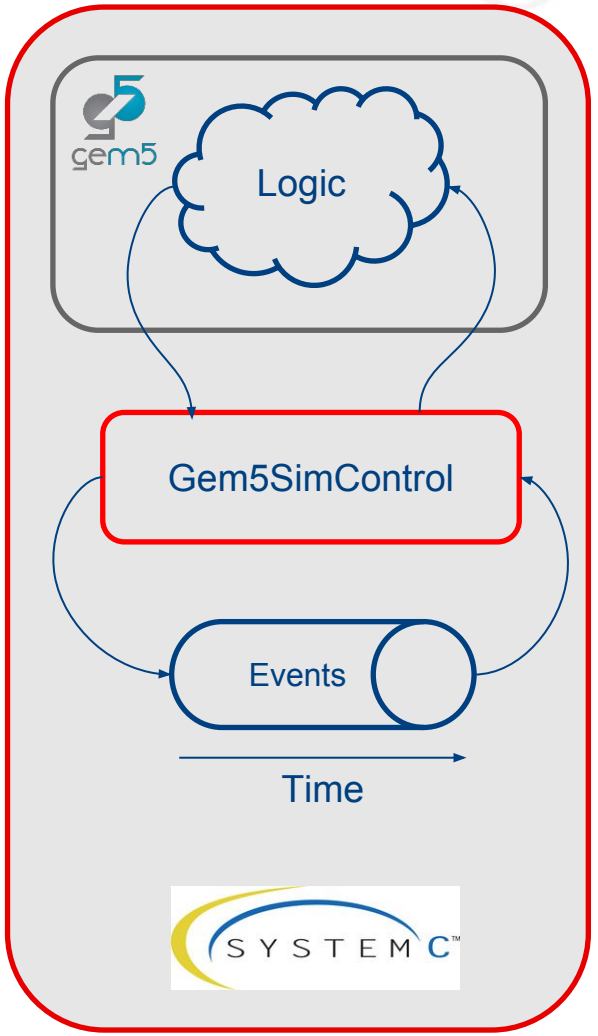
Coupling gem5 with SystemC

gem5 supports a SystemC coupling:

- Gem5 is build as a C++ library.
- It is linked into a SystemC simulation.
- A SystemC object implements the gem5 event queue.
- Communication is done via TLM







Transaction Models in gem5

Timing

- The most detailed access: queuing delay + resource contention
- Similar to the TLM **nb_transport** interface.

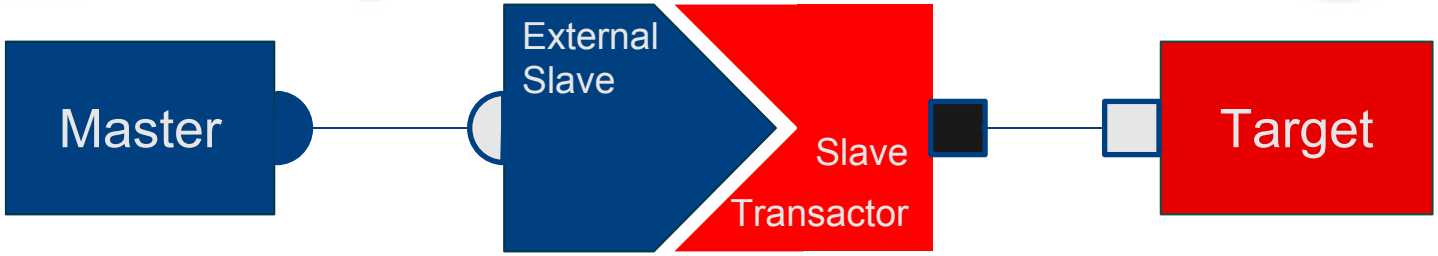
Atomic

- Accesses are a faster than detailed access
- Used for **fast forwarding** and **warming up caches**
- Similar to the TLM **b_transport** interface
- Not good for performance simulation

Functional

- Similar to **transport_dbg** e.g. loading binaries, avoiding deadlocks in multi-level cache coherent networks

Converting between TLM and gem5



recvFunctional(…) → transport_dbg(…)

recvAtomic(…) → b_transport(…)

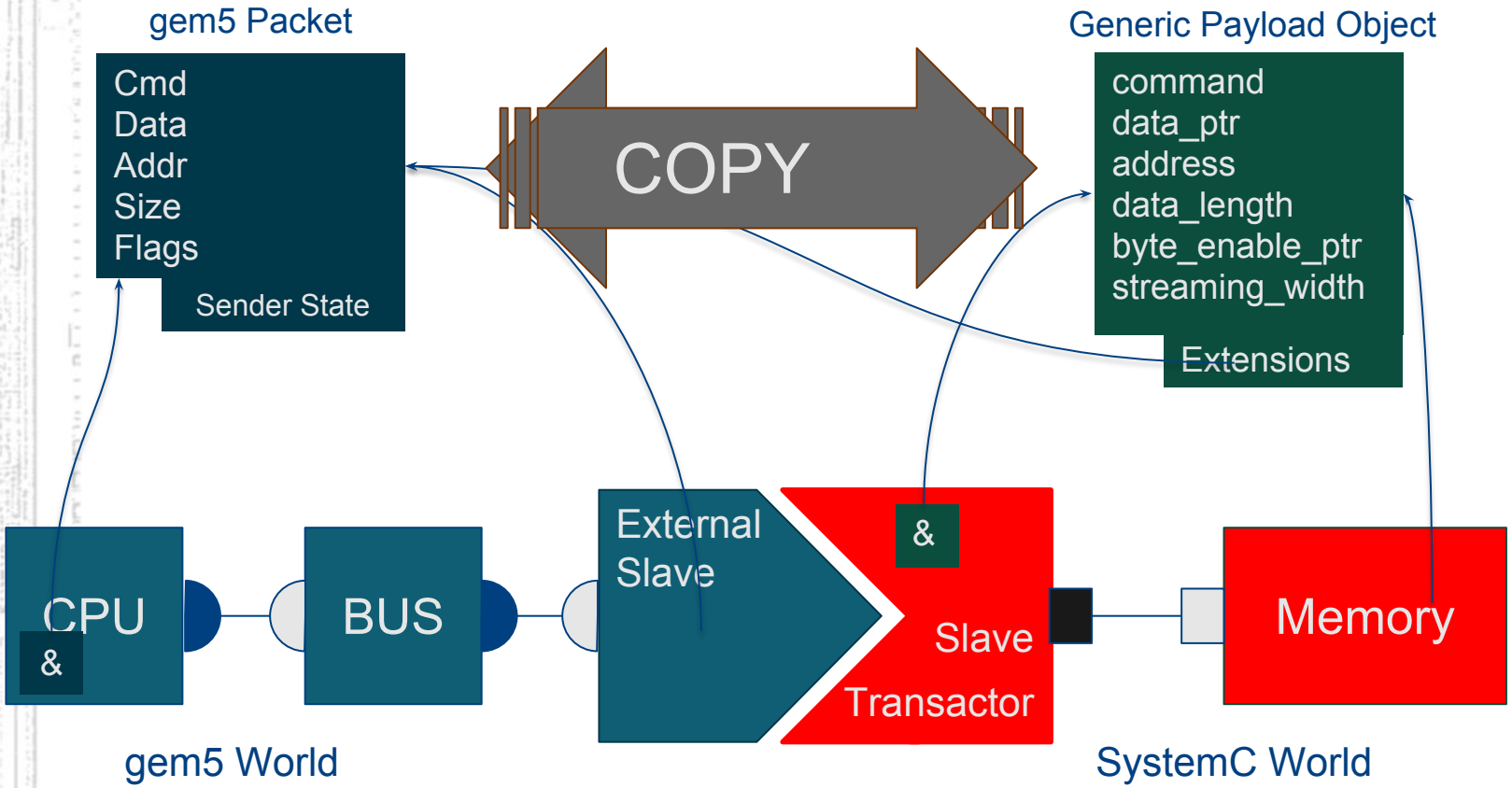
recvTimingReq(…) → nb_transport(…)

transport_dbg(…) → recvFunctional(…)

b_transport(…) → recvAtomic(…)

nb_transport(…) → recvTimingReq(…)

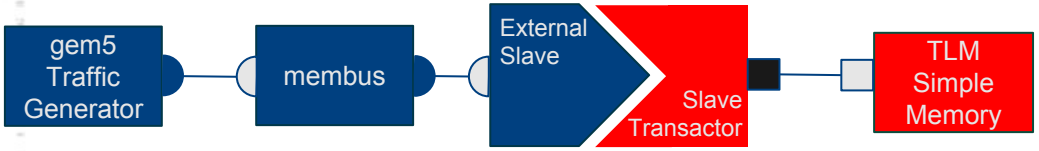
Transaction Explained



How to get Started?

Study the Examples in /gem5/utils/tlm/

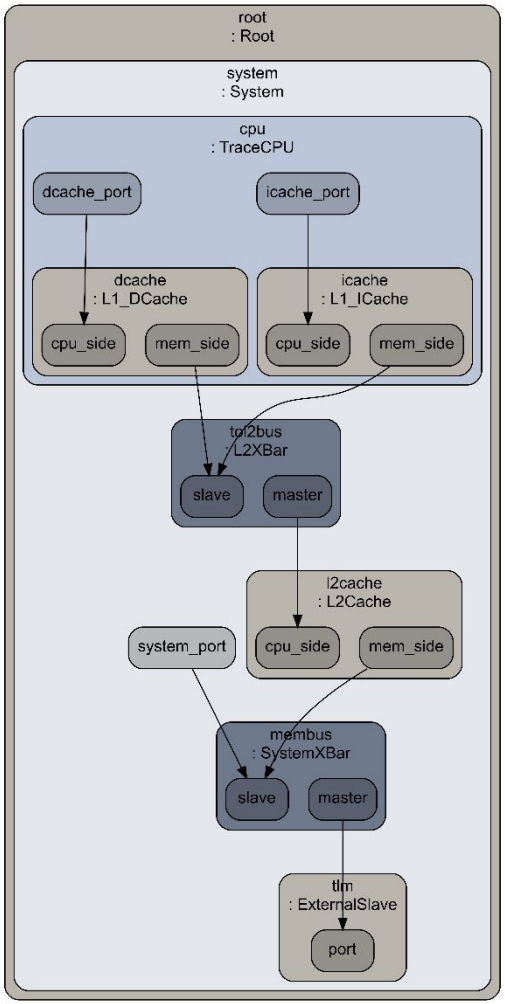
- Slave Example:



- Master Example:

- Full System Example:

```
../../build/ARM/gem5.opt ../../configs/example/fs.py                \
--tlm-memory=transactor --cpu-type=TimingSimpleCPU --num-cpu=1      \
--mem-type=SimpleMemory --mem-size=512MB --mem-channels=1 --caches \
--l2cache --machine-type=VExpress_EMM                               \
--dtb-filename=vexpress.aarch32.ll_20131205.0-gem5.1cpu.dtb          \
--kernel=vmlinux.aarch32.ll_20131205.0-gem5                         \
--disk-image=linux-aarch32-ael.img
```



Practical Usage: General Flow

1. Compile gem5 normally: scons build/ARM/gem5.opt
2. Compile gem5 as a library:
   scons --with-cxx-config --without-python build/ARM/libgem5_opt.so
3. Include the gem5 modules Gem5SimControl and Gem5SlaveTransactor and/or Gem5MasterTransactor in your SystemC project and connect them to your SystemC models. Be sure to pass an individual port name to the constructor of each transactor.
4. Compile your project and link against the gem5 library.
5. Run normal gem5 with a custom python script or fs.py with --tlm-memory=<port-name> to generate m5out/config.ini. Be sure to set the tlm_data attribute of the External Masters/Slaves to the port name of the corresponding SystemC transactor.
6. Run your SystemC project and pass the m5out/config.ini file to your Gem5SimControl object.

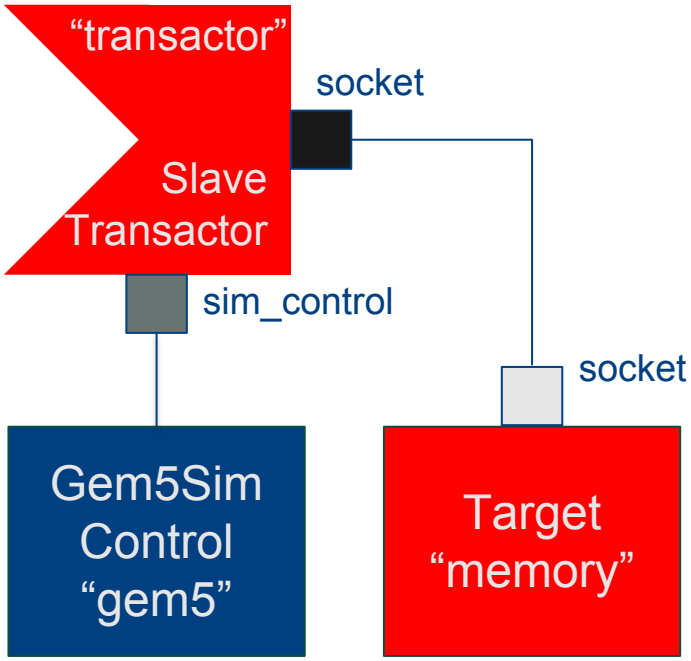
A Memory Module in SystemC

```
struct Target: public sc_module {
  // TLM interface socket:
  tlm_utils::simple_target_socket<Target>   socket;

  // Storage
  unsigned char *mem;

  // Constructor
  Target(sc_core::sc_module_name  name, /* ... */);
  SC_HAS_PROCESS(Target);

  // TLM interface functions
  virtual void b_transport(tlm::tlm_generic_payload& trans,
                                      sc_time& delay);
  virtual unsigned int transport_dbg(tlm::tlm_generic_payload& trans);
  virtual tlm::tlm_sync_enum nb_transport_fw(
                tlm::tlm_generic_payload& trans,
                tlm::tlm_phase& phase,
                sc_time& delay);

  // ...
};
```

→ util/tlm/examples/slave_port/sc_target.hh

Connect the Memory to gem5

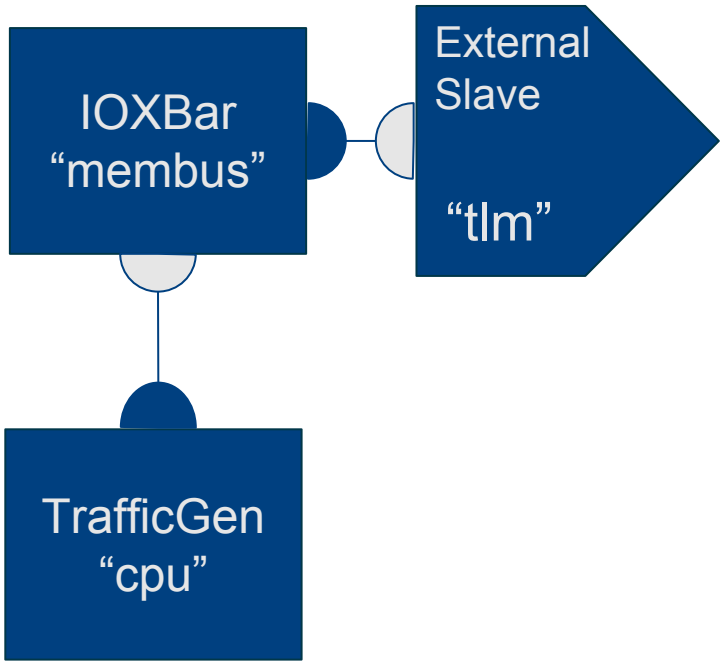
```
int sc_main(int argc, char **argv)
{
    // Instantiate all modules
    Gem5SystemC::Gem5SimControl
            sim_control("gem5", /* config ... */);
    Gem5SystemC::Gem5SlaveTransactor
            transactor("transactor", "transactor");
    Target memory("memory", /* config ... */);

    // Bind modules
    memory.socket.bind(transactor.socket);
    transactor.sim_control.bind(sim_control);

    // Start simulation
    sc_core::sc_start();

    return EXIT_SUCCESS;
}
```



→ util/tlm/examples/slave_port/main.cc

Configure gem5

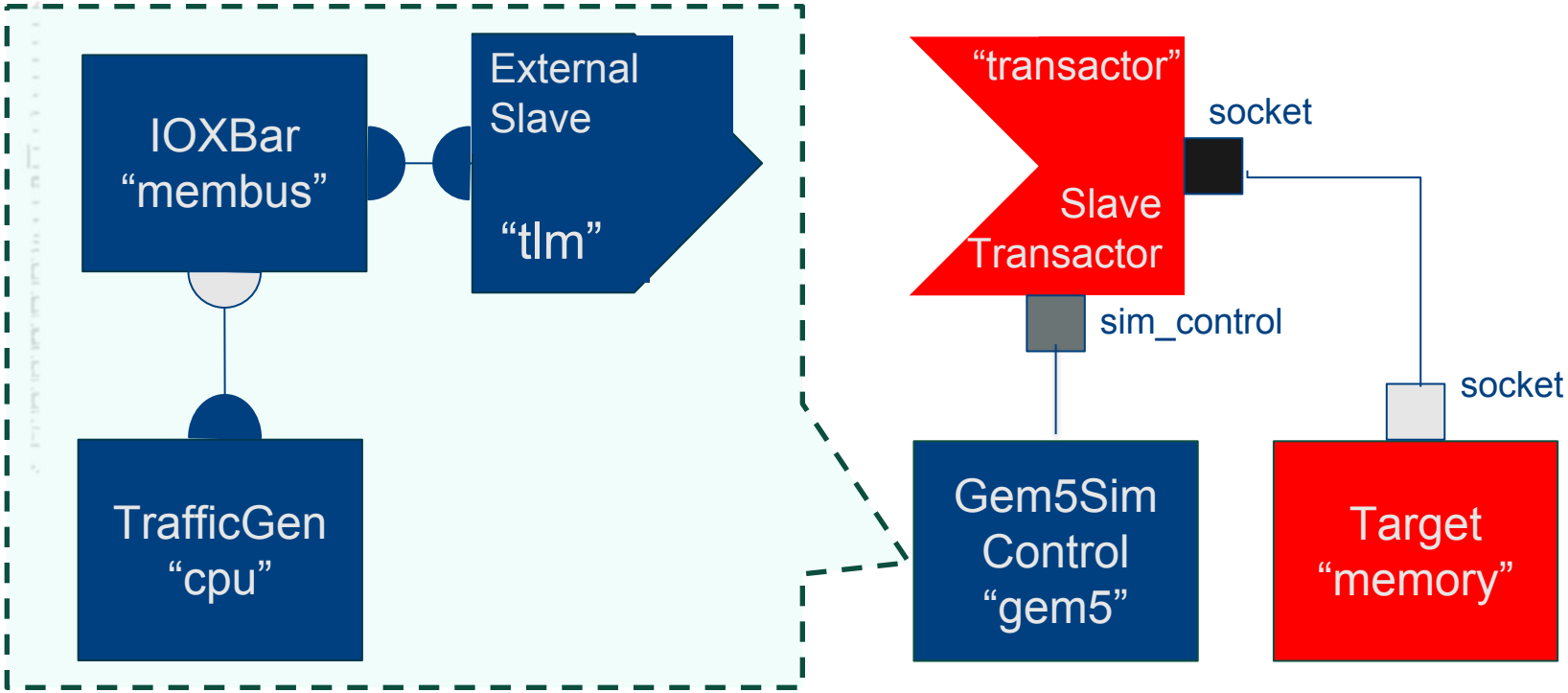
```
# Create a system with a Crossbar and a TrafficGenerator
system = System()
system.membus = IOXBar(width = 16)
# This must be instanciated, even if not needed
system.physmem = SimpleMemory()
system.cpu = TrafficGen(config_file = "tgen.cfg")
system.clk_domain = SrcClockDomain(clock = '1.5GHz',
    voltage_domain = VoltageDomain(voltage = '1V'))

# Create an external TLM port:
system.tlm = ExternalSlave()
system.tlm.addr_ranges = [AddrRange('512MB')]
system.tlm.port_type = "tlm_slave"
system.tlm.port_data = "transactor"

# Route the connections:
system.cpu.port = system.membus.slave
system.system_port = system.membus.slave
system.membus.master = system.tlm.port

# Start the simulation:
root = Root(full_system = False, system = system)
root.system.mem_mode = 'timing'
m5.instantiate()
m5.simulate()
```



→ util/tlm/conf/tlm_slave.py

Run the Simulation

1. Build the example: $ cd util/tlm && scons
2. Create a gem5 config.ini file:
   $ ../../build/ARM/gem5.opt conf/tlm_slave.py
3. Run the simulation:
   $ build/examples/slave_port/gem5.sc m5out/config.ini



Simulation Output

```
$ build/examples/slave_port/gem5.sc m5out/config.ini -e 200000 -d TrafficGen
[...]
0 s (=) : sc_main Start of Simulation
info: Entering event queue @ 0.  Starting simulation...
5 ns (=) : system.cpu LinearGen::getNextPacket: r to addr 0, size 4
5 ns (=) : system.cpu Next event scheduled at 10000
10 ns (=) : system.cpu LinearGen::getNextPacket: w to addr 4, size 4
15 ns (=) : system.cpu Received retry
15 ns (=) : system.cpu LinearGen::getNextPacket: r to addr 8, size 4
16675 ps (=) : system.cpu Received retry
75 ns (=) : system.cpu Received retry
75 ns (=) : system.cpu LinearGen::getNextPacket: r to addr c, size 4
[...]
Exit at tick 200000, cause: simulate() limit reached
```

The binary expects various options:

- -e end of simulation at tick
- -d set a gem5 debug flag

Outline

- DRAM Evolution
- Motivation for Memory Research
- Accurate and Fast Models are Needed
- SystemC / TLM2.0 coupling in gem5
- **Gem5 Tips & Tricks**

gem5 Tips & Tricks

Motivation:

- gem5 has Lots of features, therefore lots of details!
- gem5's learning curve starts slow.
- People get motivated to deep exploration when basic things work.
- Easy to remember all those tricks we learned in the past if they are in a repo!


```
$ git clone https://github.com/tukl-msd/gem5.TnT.git
$ cd gem5.TnT
$ sudo bash dep_install.sh
$ bash get_essential_repos.sh
$ bash get_essential_fs.sh
$ bash get_benchmarks.sh
$ cd arch/arm
$ bash run_arm_fs_android_ics.sh
```

github.com/tukl-msd/gem5.TnT

Take Away Message

- Memory subsystem is complex
- DRAM devices have complex interface timing
- Large varying access time and energy consumption
- It is important to model this complex behavior very accurately but at the same time with high simulation speed
- Several interfaces HMC serial, HBM highly parallel, 3D DRAM TSVs + external interface
- Temperature issues (refresh interval)
- Hybrid Memory Systems (DRAM + NVM)
- From HW to OS many challenges!

“The Memory System: You Can't Avoid It,You Can't Ignore It, You Can't Fake It”. (Bruce Jacob)

MICROELECTRONIC SYSTEMS DESIGN RESEARCH GROUP

Acknowledgements

Many thanks to the people that developed and/or collaborated to the research projects mentioned in this presentation.

Aasheesh Kolli, Abdul Mutaal, Ali Saidi, Andreas Hansson, Aniruddha N. Udipi, Arkaprava Basu, Benny Akesson, Bradford Beckmann, Carl Rheinländer, Chirag Sudarshan, Christian Menard, Christian Weis, David A. Wood, Deepak M. Mathew, Derek R. Hower, Felipe Prado, Gabriel Black, Jason Lowe-Power, Jeronimo Castrillon, Joel Hestness, Karthik Chandrasekar, Kees Goossens, Korey Sewell, Mark D. Hill, Martin Schultheis, Matthias Jung, Muhammad Shoaib, Nathan Binkert, Neha Agarwal, Nilay Vaish, Norbert Wehn, Omar Naji, Radhika Jagtap, Rathijit Sen Stefan Diestelhorst, Steven K. Reinhardt, Somayeh Sardashti, Subash Kannoth, Sven Goossens, Thomas Wenisch, Tushar Krishna, Yonghui Li.

Other contributions and collaborations

- **Elastic Traces** - an alternative approach to speed-up simulation
- **DRAM power-down states in gem5** - more sophisticated memory controller model

Thank you

For more information visit ems.eit.uni-kl.de

References

[1] **System Simulation with gem5 and SystemC: The Keystone for Full Interoperability**, Christian Menard, Matthias Jung, Jeronimo Castrillon, Norbert Wehn, *Proceedings of the IEEE International Conference on Embedded Computer Systems Architectures Modeling and Simulation (SAMOS)*, Jul 2017.

[2] **Exploring System Performance using Elastic Traces: Fast, Accurate and Portable**, Radhika Jagtap, Stefan Diestelhorst, Andreas Hansson, Matthias Jung, Norbert Wehn, *IEEE International Conference on Embedded Computer Systems Architectures Modeling and Simulation (SAMOS)*, July, 2016, Samos Island, Greece.

[3] **DRAMPower: Open-source DRAM Power & Energy Estimation Tool** Karthik Chandrasekar, Christian Weis, Yonghui Li, Sven Goossens, Matthias Jung, Omar Naji, Benny Akesson, Norbert Wehn, and Kees Goossens URL: http://www.drampower.info

[4] **A Bank-Wise DRAM Power Model for System Simulations** D. M. Mathew, É. F. Zulian, S. Kannoth, M. Jung, C. Weis, N. Wehn. *International Conference on High-Performance and Embedded Architectures and Compilers 2017 (HiPEAC), Workshop on: Rapid Simulation and Performance Evaluation: Methods and Tools (RAPIDO)*, January, 2017, Stockholm, Sweden.

[5] **Simulating DRAM controllers for future system architecture exploration**. Andreas Hansson, Neha Agarwal, Aasheesh Kolli, Thomas Wenisch and Aniruddha N. Udipi. In Proceedings of the International Symposium on Performance Analysis of Systems and Software (ISPASS), March 2014.

[6] **The gem5 Simulator**. Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. May 2011, ACM SIGARCH Computer Architecture News.

[7] **DRAMSpec: A High-Level DRAM Timing, Power and Area Exploration Tool** C. Weis, A. Mutaal, O. Naji, M. Jung, A. Hansson, N. Wehn. *International Journal of Parallel Programming (IJPP)*, Springer, 2016.

References

[8] **Integrating DRAM Power-Down Modes in gem5 and Quantifying their Impact** R. Jagtap, M. Jung, W. Elsasser, C. Weis, A. Hansson, N. Wehn. *International Symposium on Memory Systems (MEMSYS 2017)*, October, 2017, Washington, DC, USA.

[9] **Exploring system performance using elastic traces: Fast, accurate and portable**. Radhika Jagtap, Matthias Jung, Stephan Diestelhorst, Andreas Hansson, Norbert Wehn. IEEE International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS), 2016

[10] **The Memory System: You Can't Avoid It, You Can't Ignore It, You Can't Fake It**, Bruce Jacob, Morgan and Claypool Publishers, 2009

[11] **What every programmer should know about memory**, *Ulrich Drepper, 2007, URL:*