2022
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
EUROPE
MUNICH, GERMANY
DECEMBER 6 - 7, 2022

The Open Source DRAM Simulator DRAMSys4.0

Dr. Matthias Jung, Fraunhofer IESE

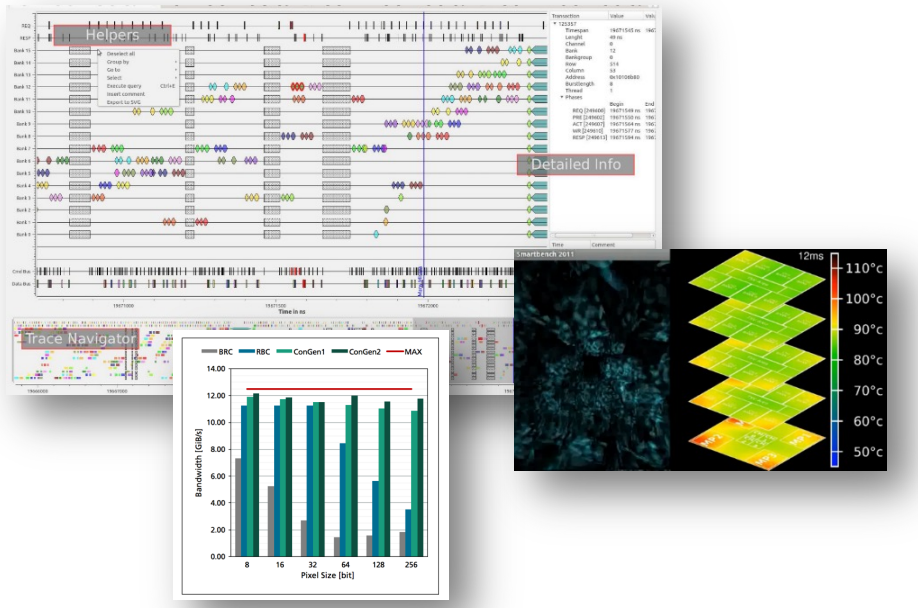
Fraunhofer IESE

accellera SYSTEMS INITIATIVE

DRAMSys in a Nutshell

Simulation and Design Space Exploration of Modern DRAM-based Memory Systems:

- Which DRAM configuration?
- When to support DDR5 or LPDDR5?
- How to configure the memory controller?
- What is the system-level application behavior?



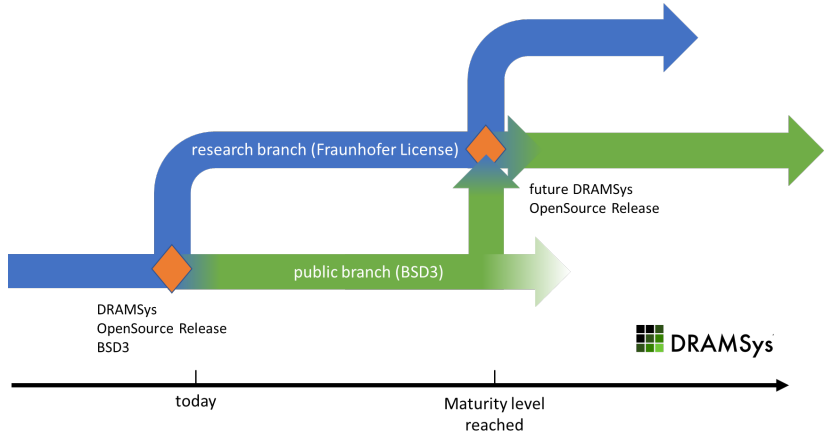
DRAMSys Offers:

- High-speed and flexible models of all standards
- Fast and accurate design space exploration
- Early identification of bottlenecks
- Connection to cores (e.g. SystemC, gem5, …)

Fork me on Github!

DRAMSys Open Source Model

- Open source: DDR3/4, LPDDR4, Wide I/O 1/2, GDDR5/X, GDDR6, and HBM2
- Commercial/academic licenses: DDR5, LPDDR5, HBM3, Trace Analyzer tool
- New standard models will be open-sourced when a level of maturity is reached
- Customer-specific consulting, modifications and developments



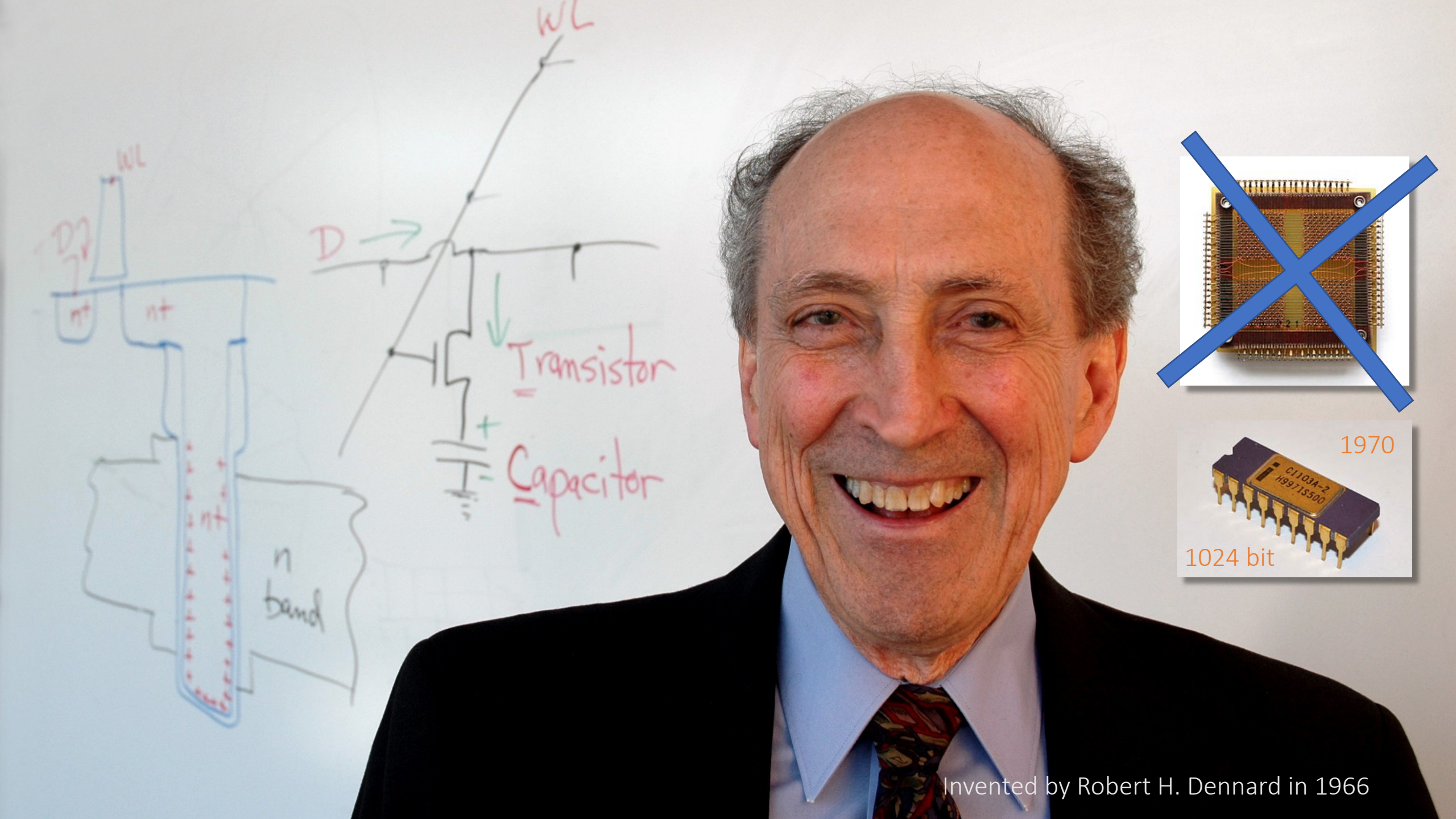
Thanks to our Key Partners:

Rambus

NOKIA

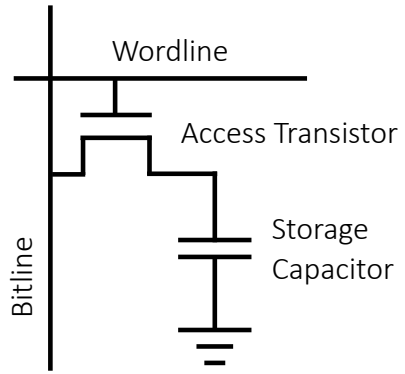
Mercedes-Benz

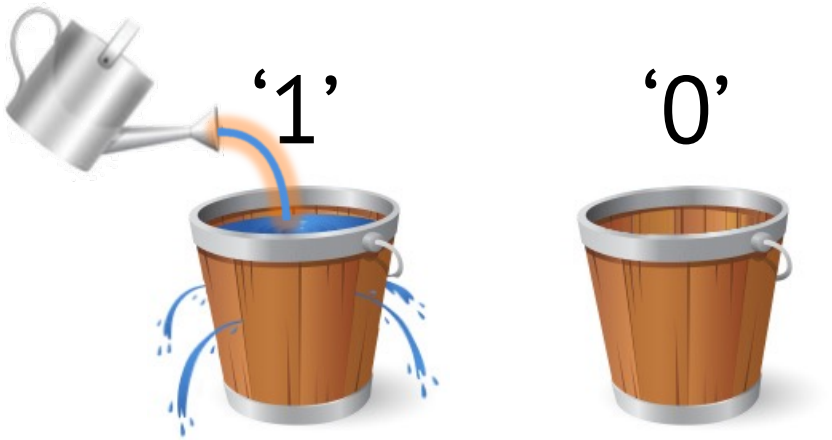
Recap: How does DRAM Work?



Invented by Robert H. Dennard in 1966

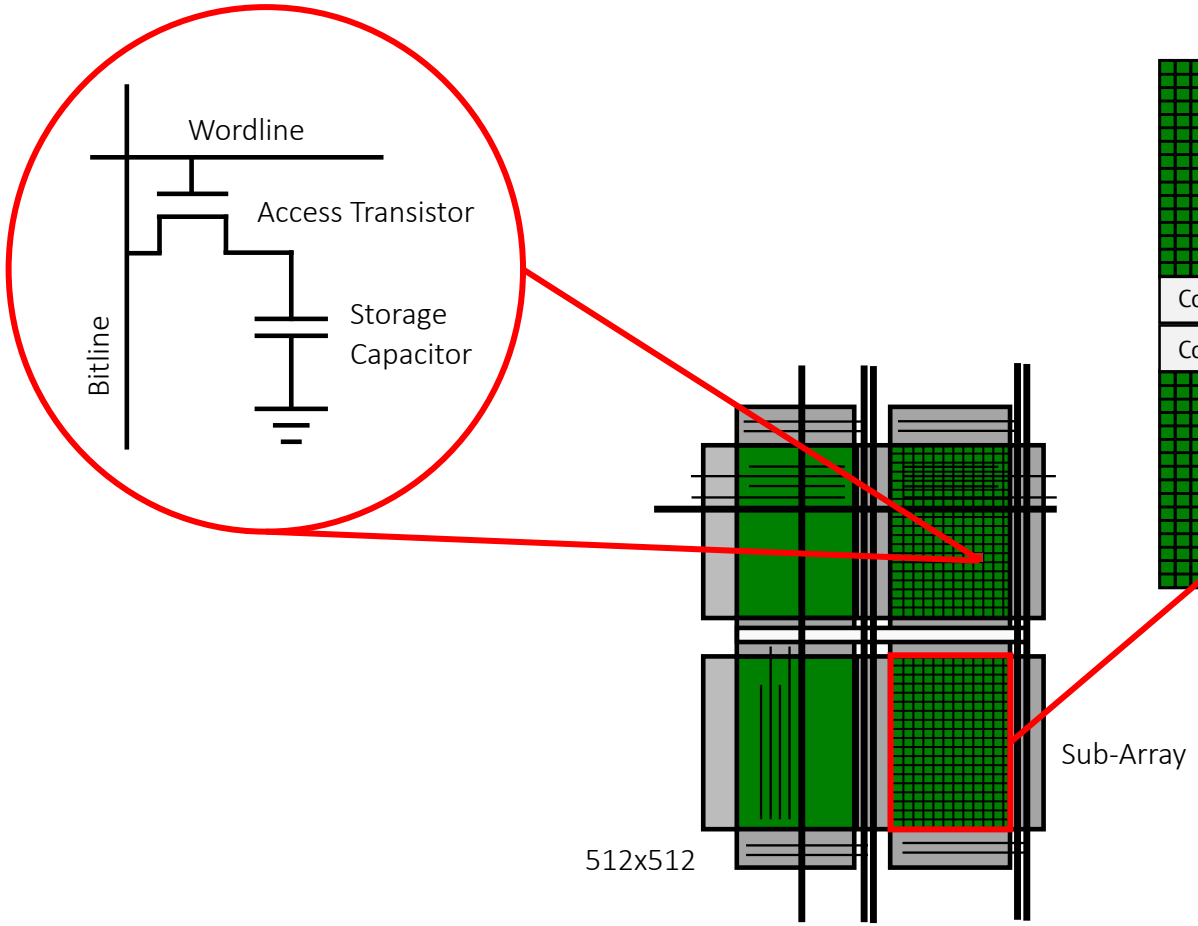
The DRAM Cell

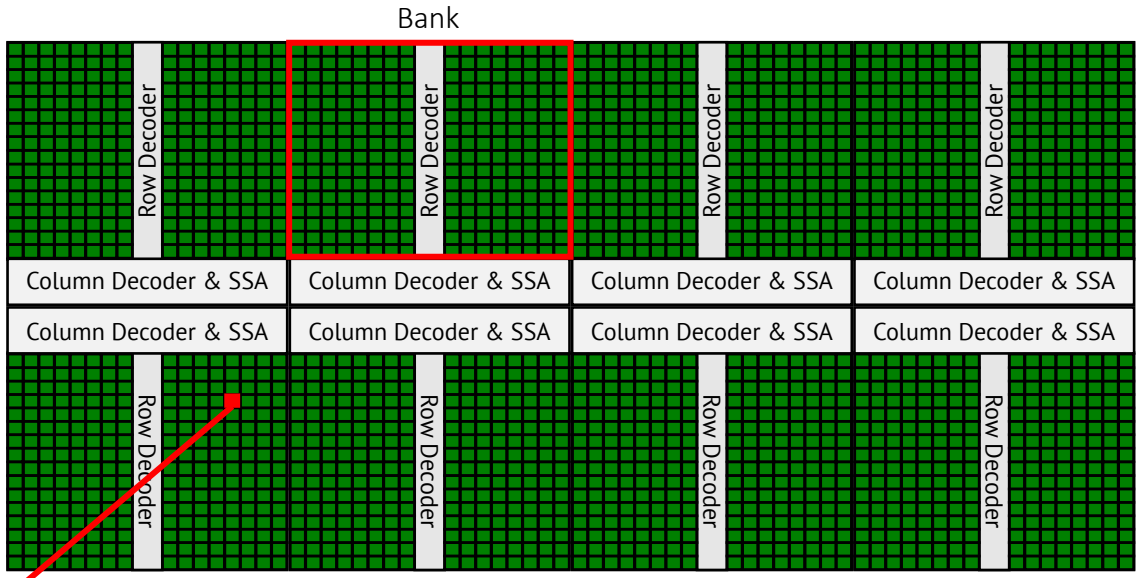




- Data is stored by capacity
- Cell is selected with access transistor
- Charged capacitor represents a ‘1’
- Discharged capacitor represents a ‘0’
- Memory is volatile
- Cell is leaky
- Refresh needed → dynamic

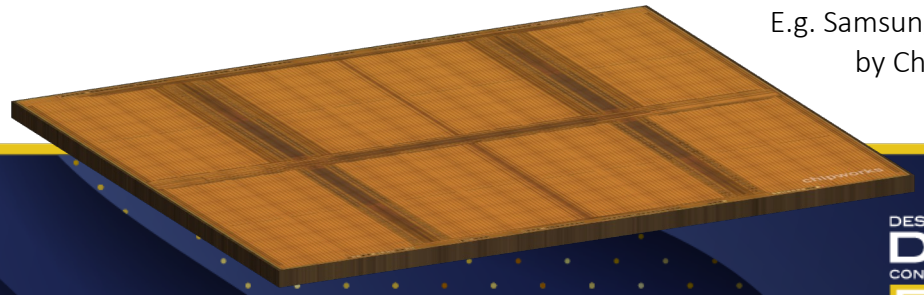
The DRAM Device / Operation



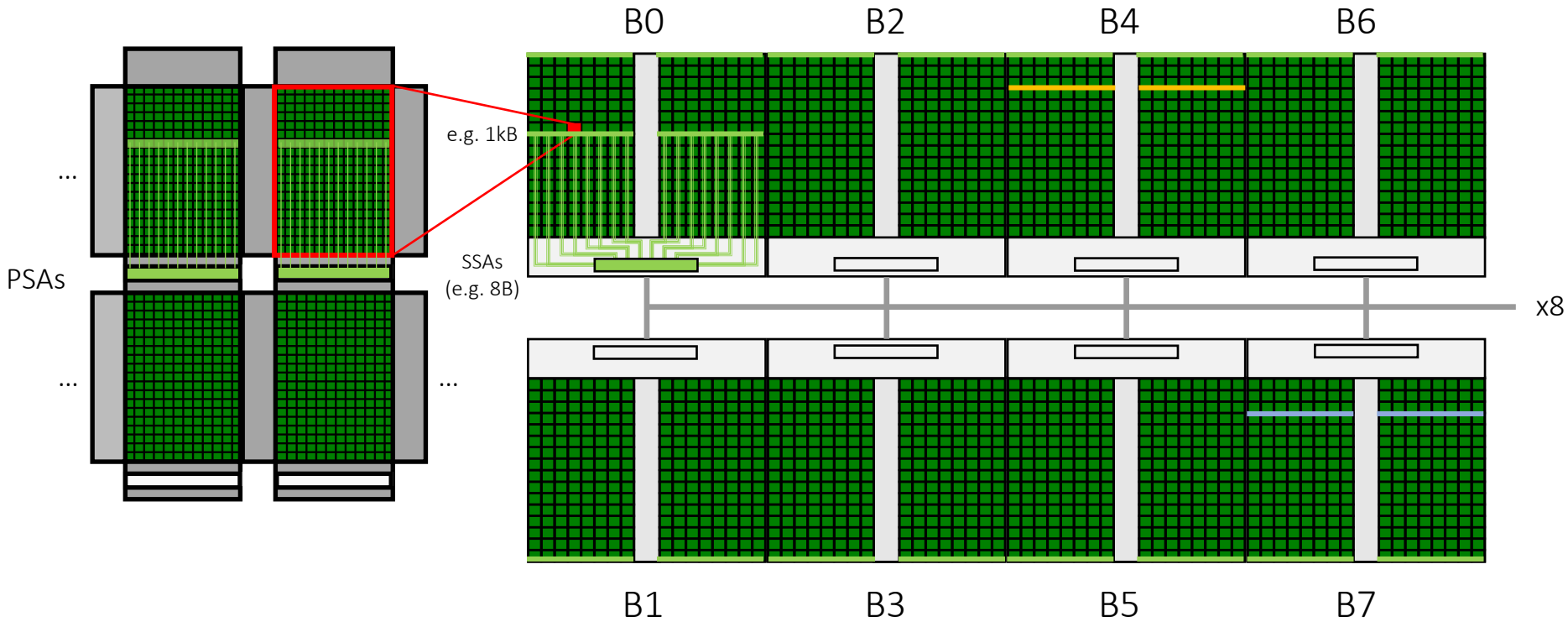


- Using Sub-Arrays for efficient wiring
- Bank parallelism, but banks share data and command bus

E.g. Samsung DDR3,
by Chipworks



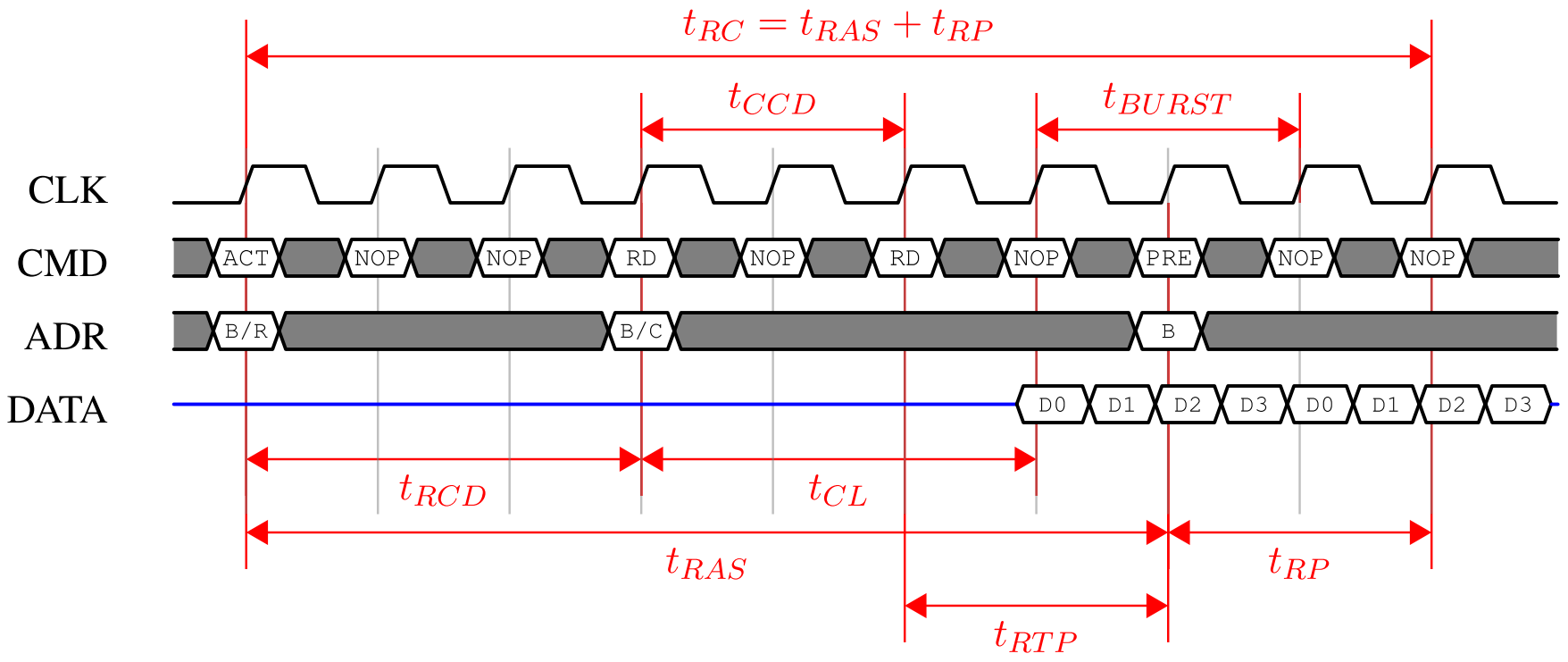
DRAMs Basic Operations



Important DRAM Commands:

- **ACT**: Activates a specific row in a specific bank (sensing into PSA) [*tRCD*]
- **RD**: Read from activated row (prefetch from PSA to SSA and burst out) [*tCL + tBURST*]
- **PRE**: Precharges set LWL=0 set LBL=VDD/2 [*tRP*]
- **REFA:** DRAM cells are leaky and have to be refreshed [*tREFI & tRFC*]

JEDEC Standard: e.g. Timing Dependencies

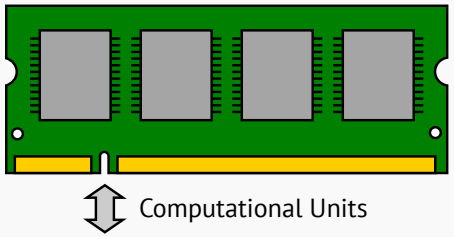


Timing dependencies must be fulfilled by the DRAM controller

Different DRAM Subsystems

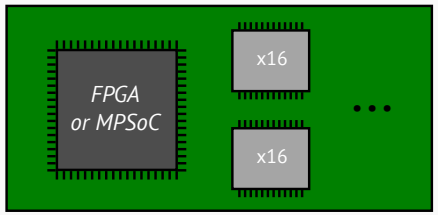
DIMM Based:

General Purpose Computers
e.g. DDR3, DDR4



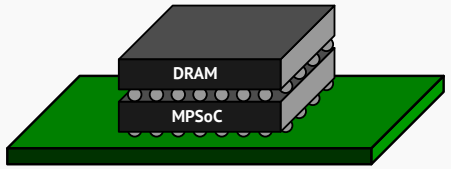
Device Based:

Embedded / Tablets / Graphic Cards
e.g. LPDDR3, GDDR5



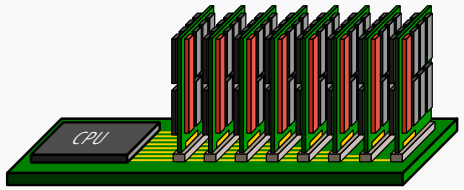
Package on Package (PoP):

Soldered on top of the MPSoC.
Smartphones
e.g. LPDDR3, LPDDR4



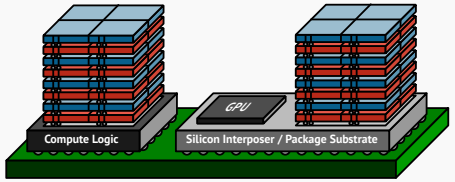
Buffer on Board:

Memory Controller on Buffer Chip,
Serial Connection
e.g. FBDIMM, IBM CDIMM, Intel SMI/SMB



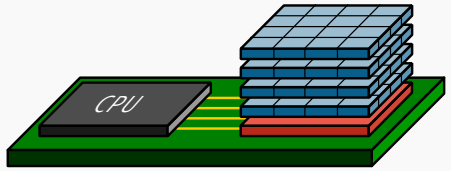
3D/2.5D-Integrated:

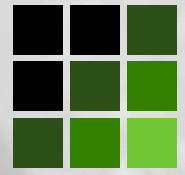
Stacked on Logic or Silicon Interposer
by means of TSVs
e.g. Wide I/O, HBM



Memory Cube:

3D-Stacked, Memory Controller on
Bottom Layer, Serial Interconnect (SerDes)
e.g. HMC, SMC



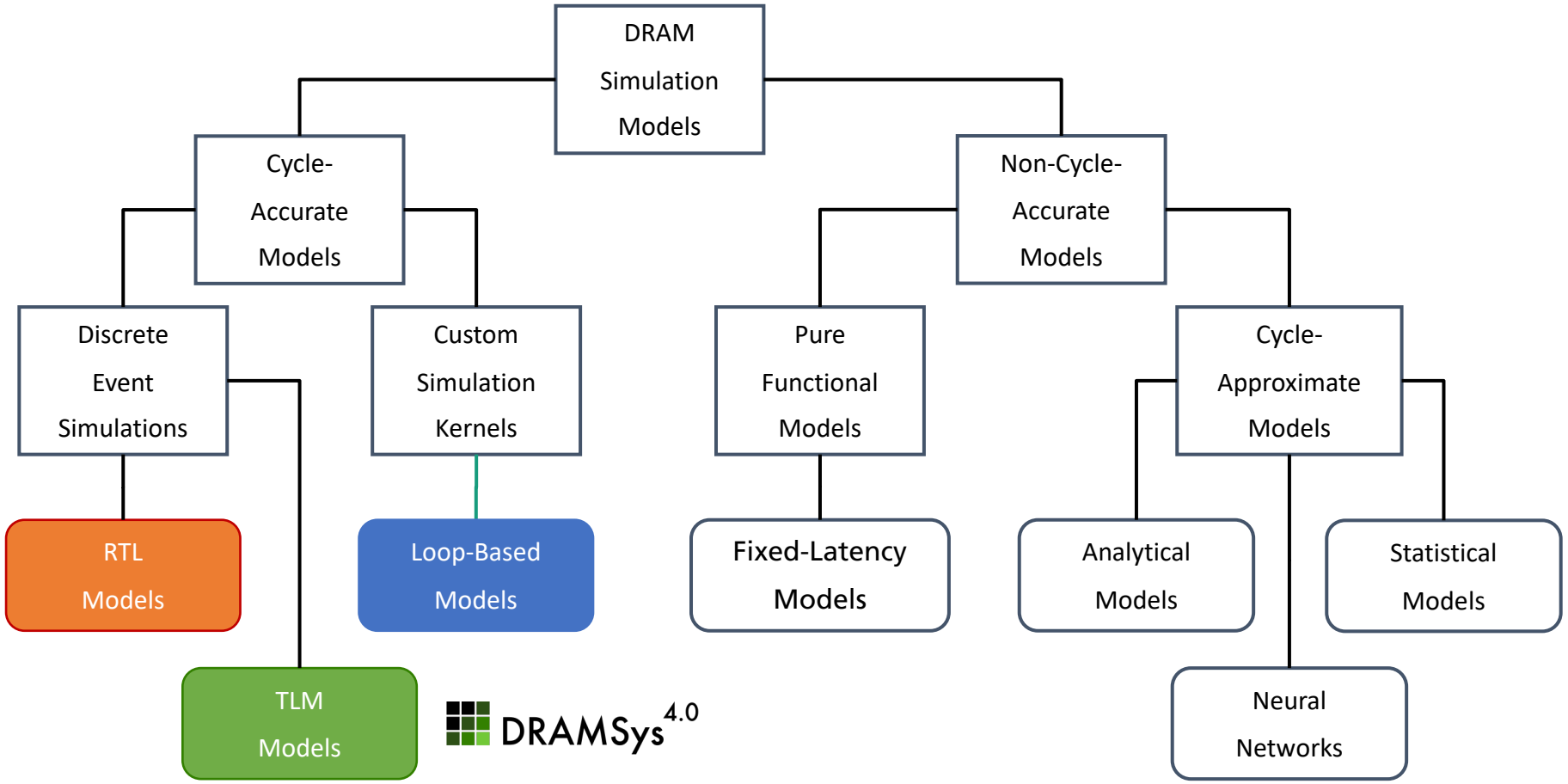


DRAMSys 4.0

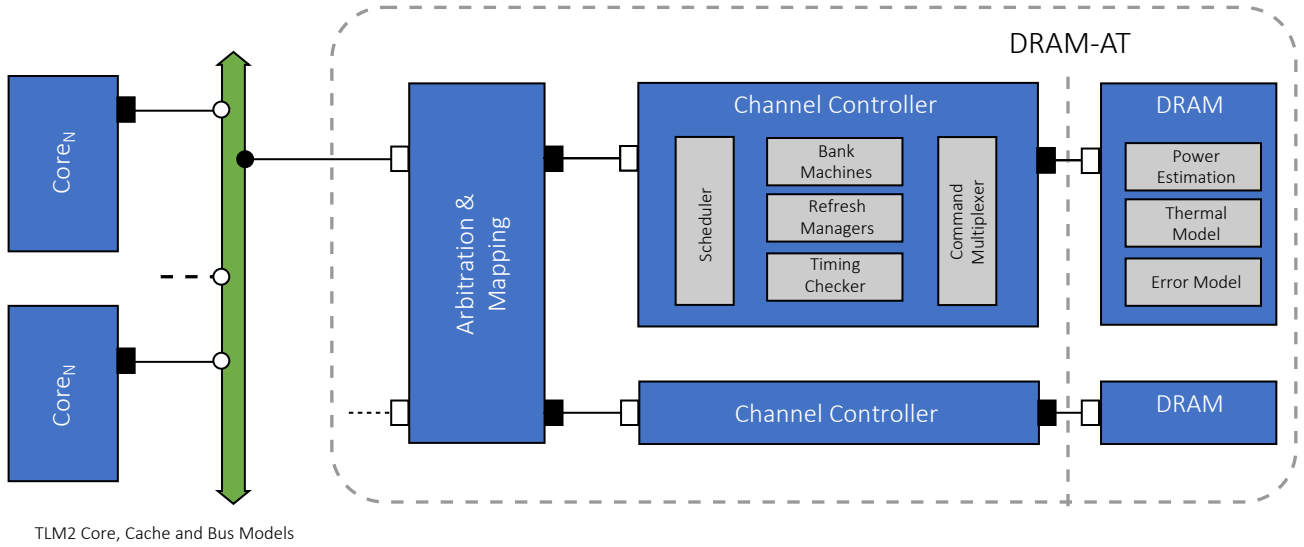
Functional Models	Non-Functional	Analysis
TLM DRAMml	Power Thermal Errors	Trace Analyzer

accellera
SYSTEMS INITIATIVE

DRAM Simulation Models



DRAMSys Architecture

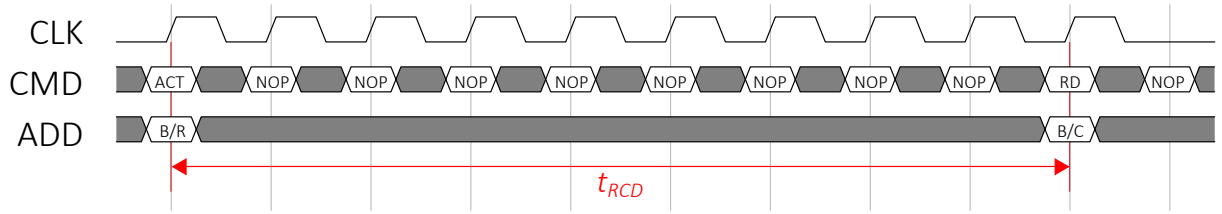


- Based on SystemC TLM2, compliant with TLM-AT coding style
- Flexible SW-Architecuture to support various JEDEC DRAM standards (e.g., DDR4, LPDDR4, GDDR6, HBM, …)
- For RTL-like accuracy a custom TLM protocol (DRAM-AT) is used

Custom TLM Protocol

- Simulation speed can be increased by reducing the number of events
- Clock signal has the highest event generation rate
- Do we need to simulate each clock cycle to generate cycle-accurate results?

RTL

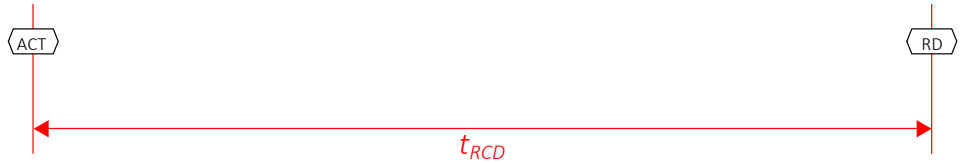


- Simulation of state changes is sufficient, idle clock cycles can be skipped!
  - Large event reduction at low memory access densities
  - No loss of accuracy

Custom TLM Protocol

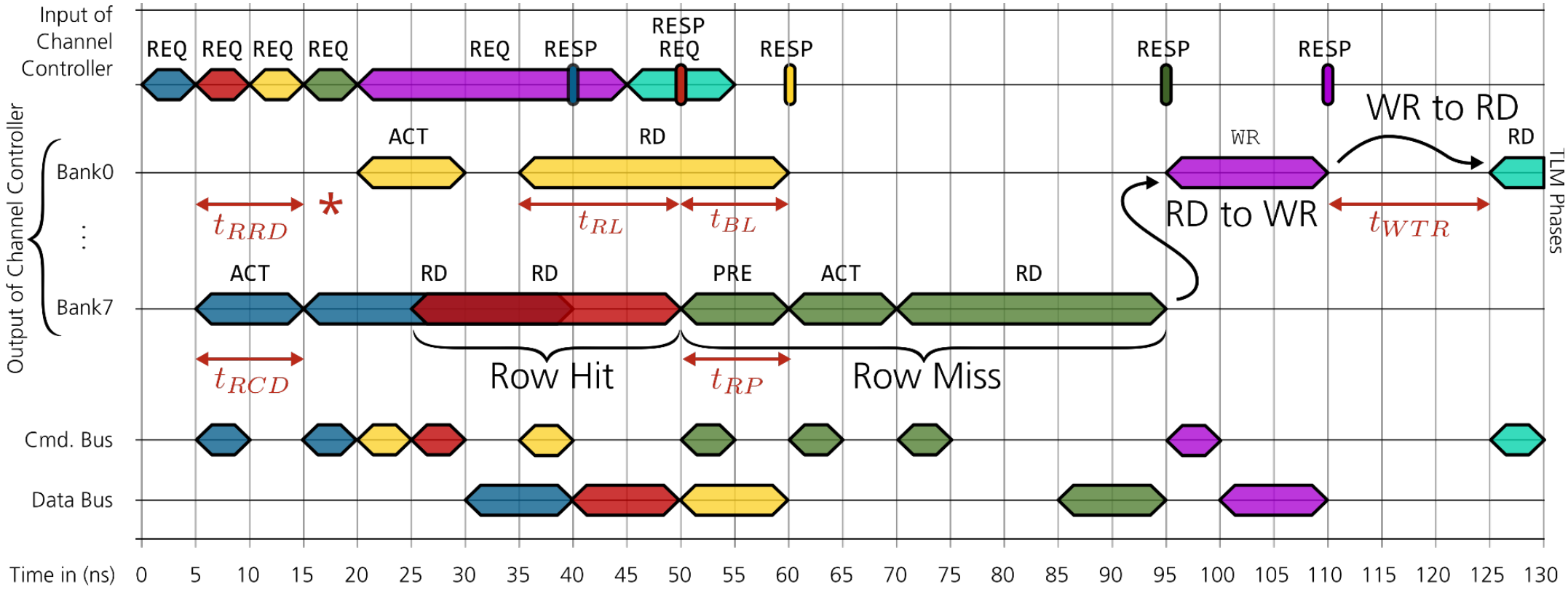
- Simulation speed can be increased by reducing the number of events
- Clock signal has the highest event generation rate
- Do we need to simulate each clock cycle to generate cycle-accurate results?

TLM

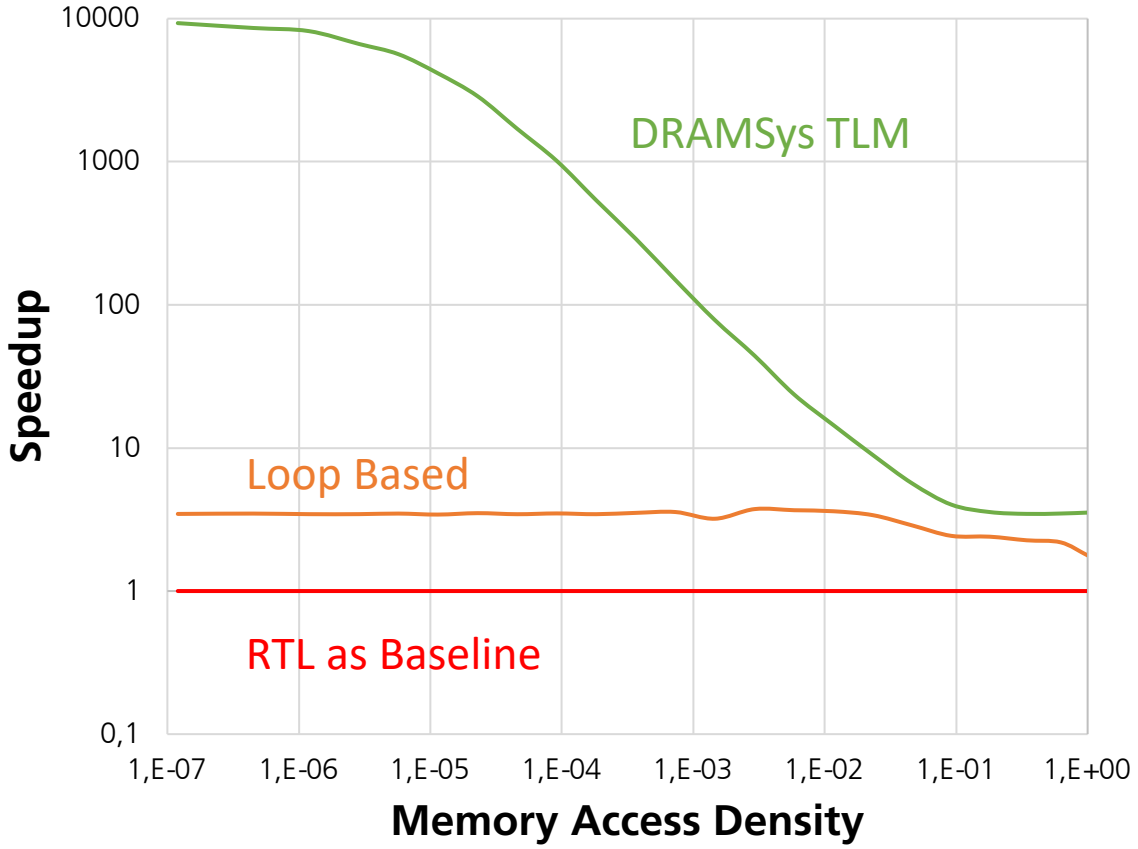


- Simulation of state changes is sufficient, idle clock cycles can be skipped!
  - Large event reduction at low memory access densities
  - No loss of accuracy

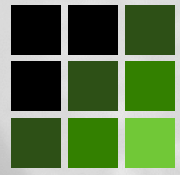
Custom TLM Protocol



DRAMSys Simulation Speed



- Simulation of only the important events
- Speedup from 4x to 10.000x depending on trace density
- Average speedups depend on applications
- Typical values: 400x
- 100% RTL Accuracy

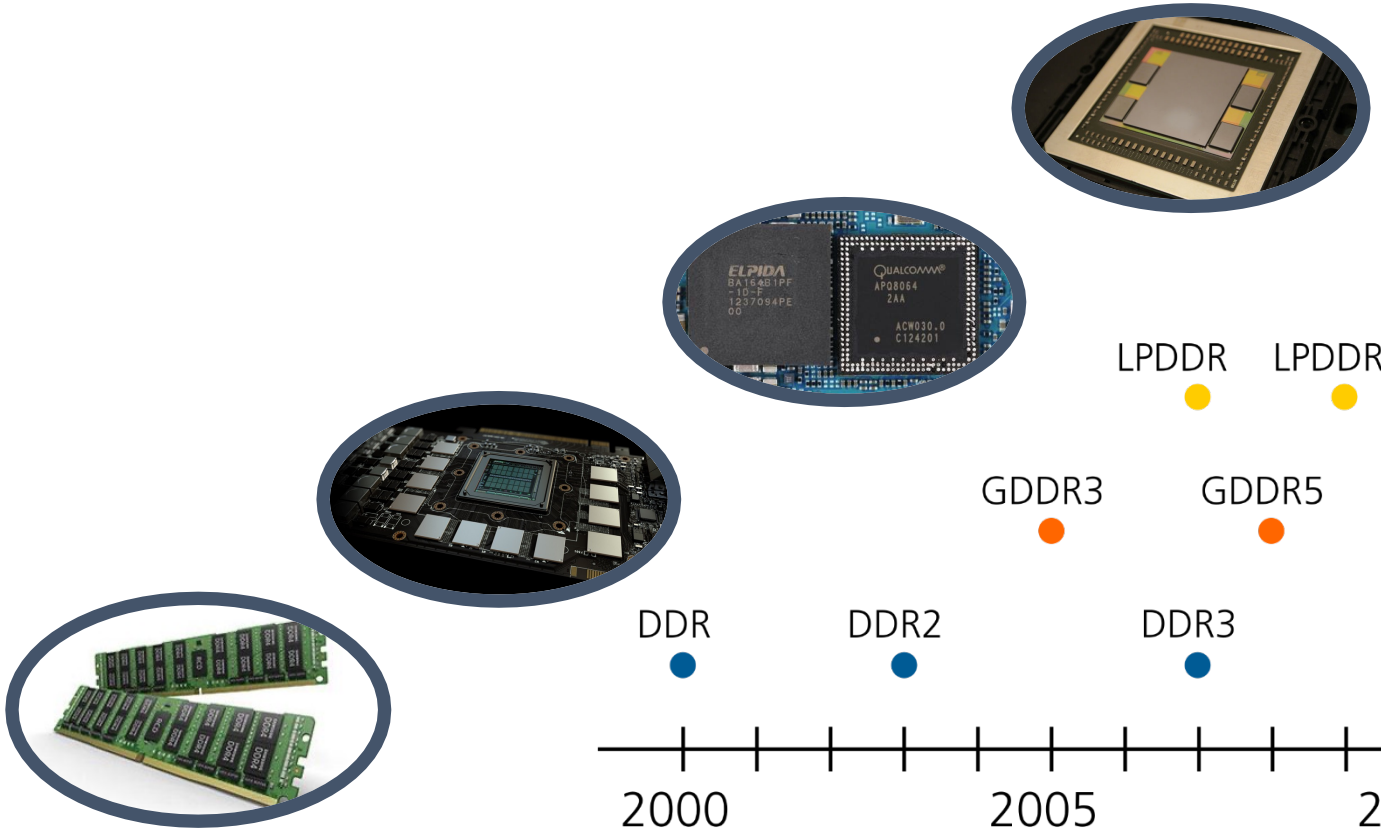


DRAMSys 4.0

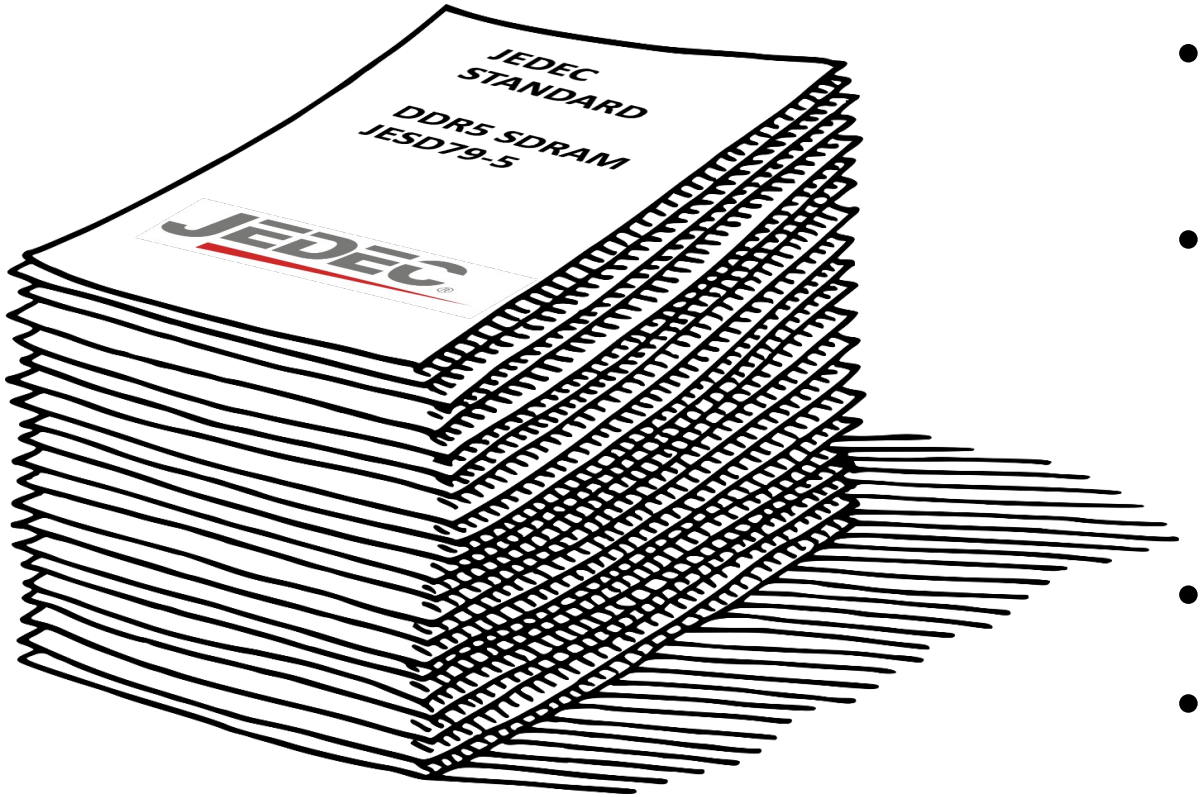
Functional Models	Non-Functional	Analysis
TLM DRAMml	Power Thermal Errors	Trace Analyzer

accellera
SYSTEMS INITIATIVE

Number of DRAM Standards is Growing!

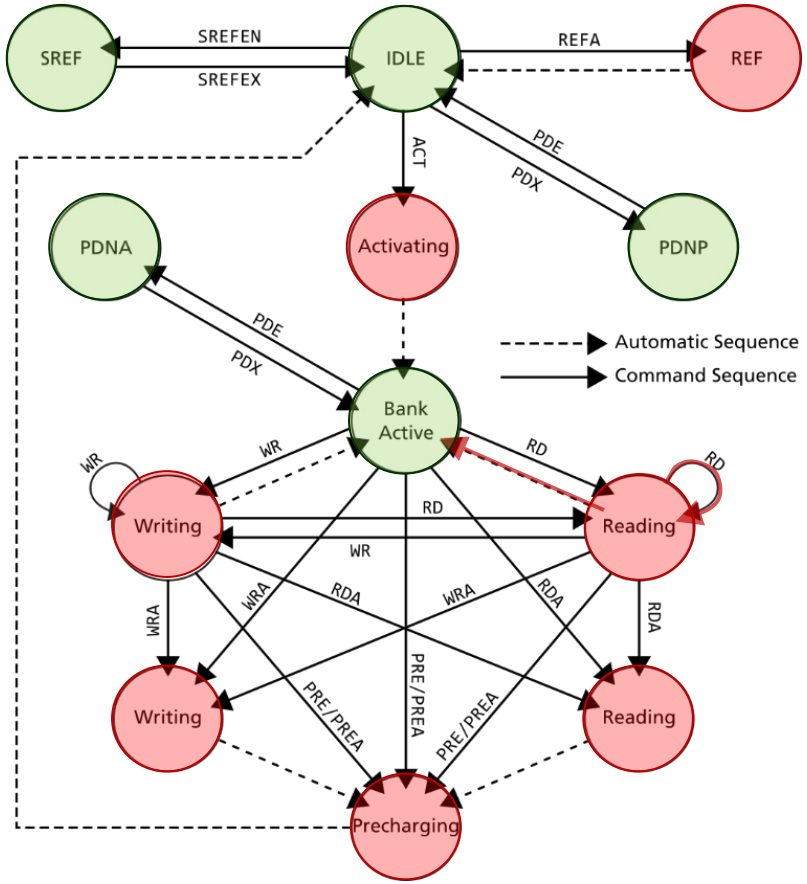


DDR5 JEDEC Standards



- Defines commands, states, timings and interface properties
- Very complex protocol
  - DDR3: 226 pages
  - DDR4: 266 pages
  - DDR5: 496 pages
- Descriptions are not formal
- And not even correct …

JEDEC Standard Description “State Machine”



DDR3 JEDEC Standard:

“This simplified State Diagram is intended to provide an overview of the possible state transitions and the commands to control them. In particular, situations involving more than one bank, the enabling or disabling of on-die termination, and some other events are not captured in full detail.”

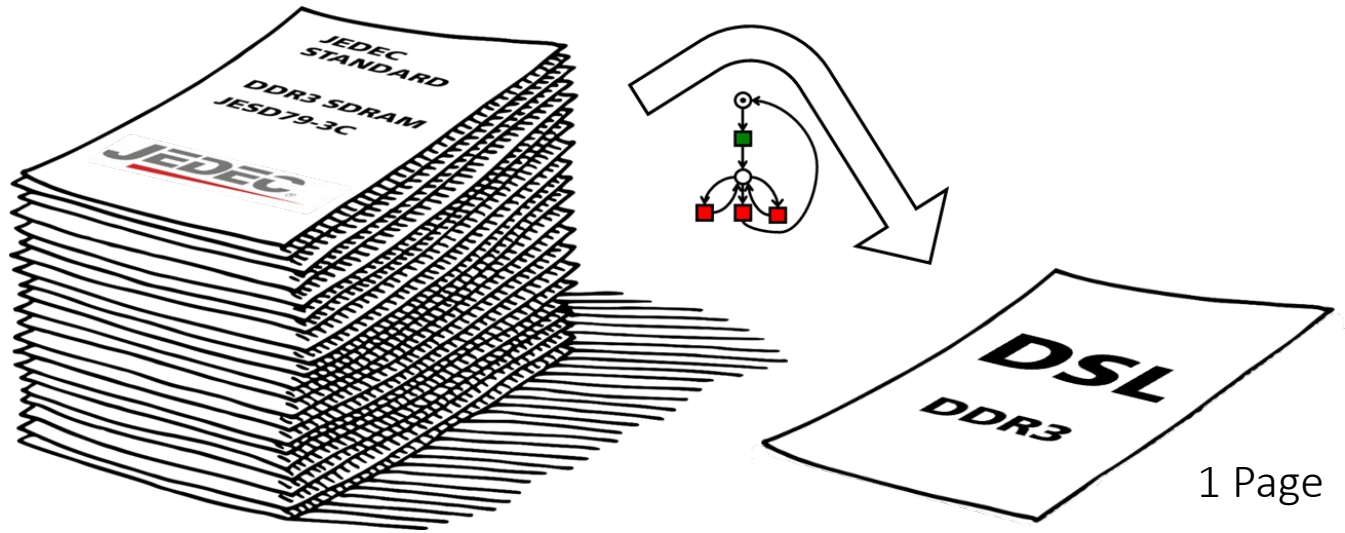
Drawbacks:

- Only 1 Bank shown i.e. no bank parallelism (because of state explosion)
- States like *Activating*, *Precharging, REF …* do not exist (There are only 5 state types!)
- Double States (2x *Reading* and *Writing*)
- Inconsistencies using automatic sequences (eg. Reading state)

DRAMml: a formal Description for JEDEC Standards

The ideal case: A *formal* language, which has the power to …

Information of ~100 Pages in the Standard

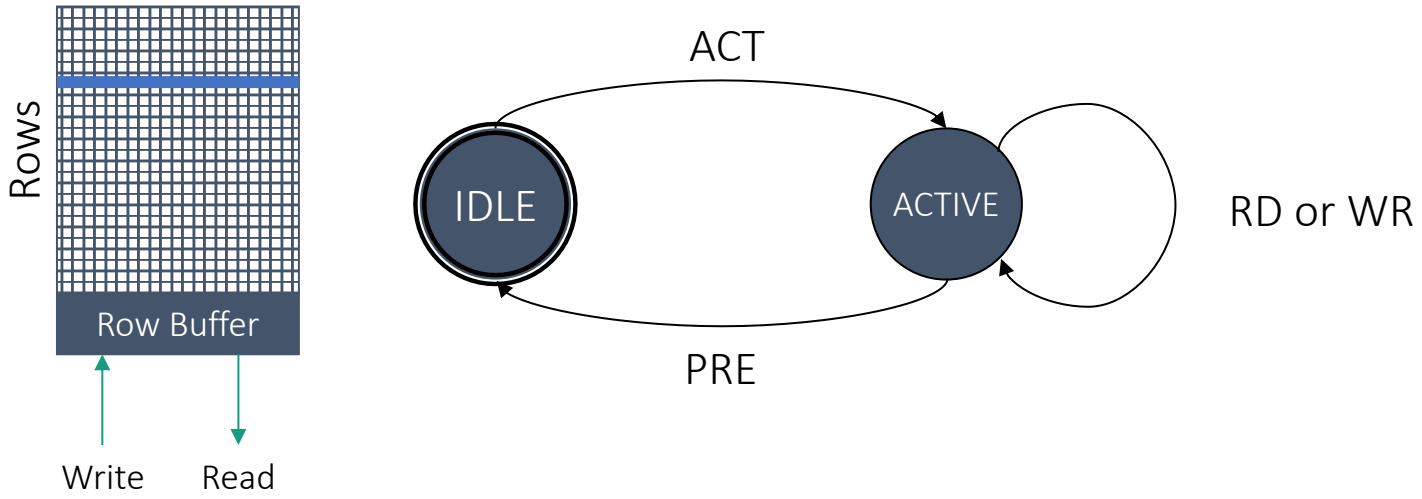


1 Page

A new standard requires a serious amount of handcraft:

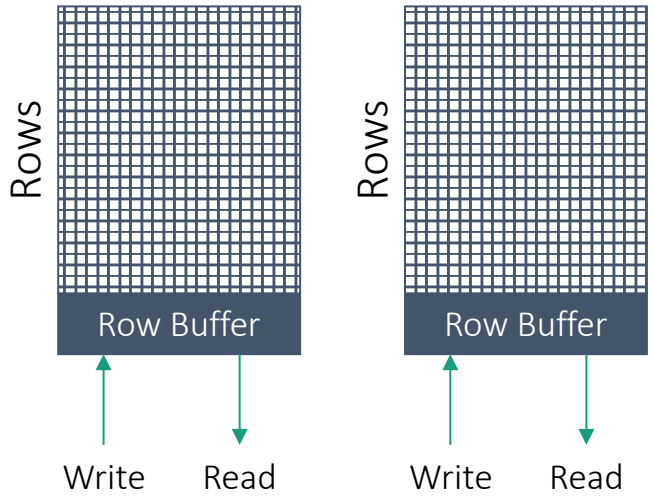
- New models for fast simulation and verification
- Adapt memory models and HW IP every time

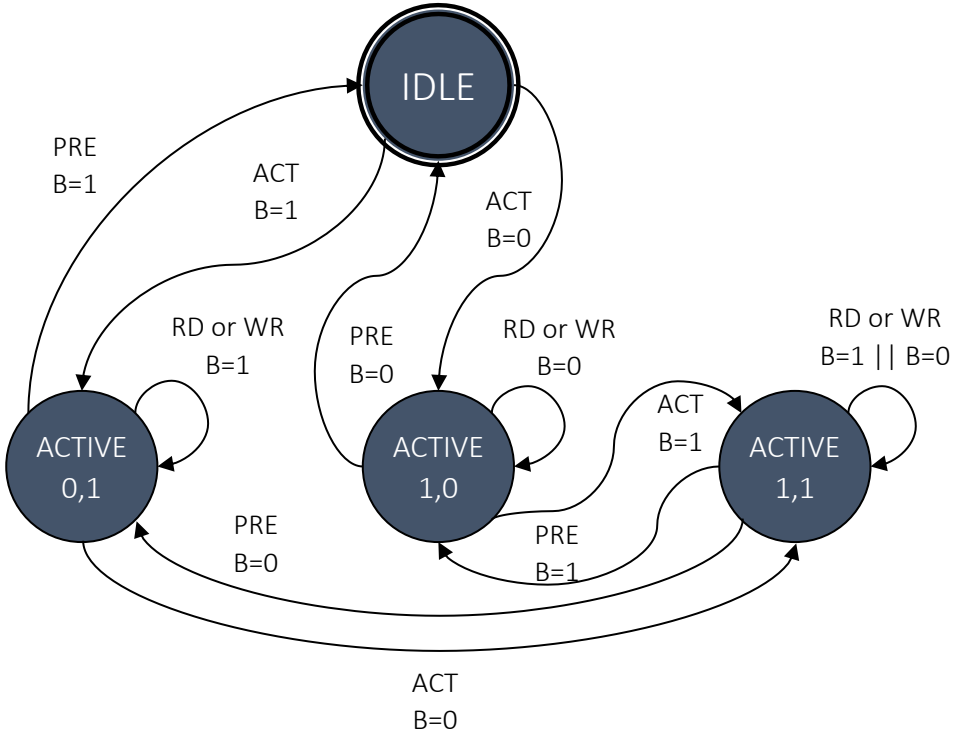
A Single Memory Bank



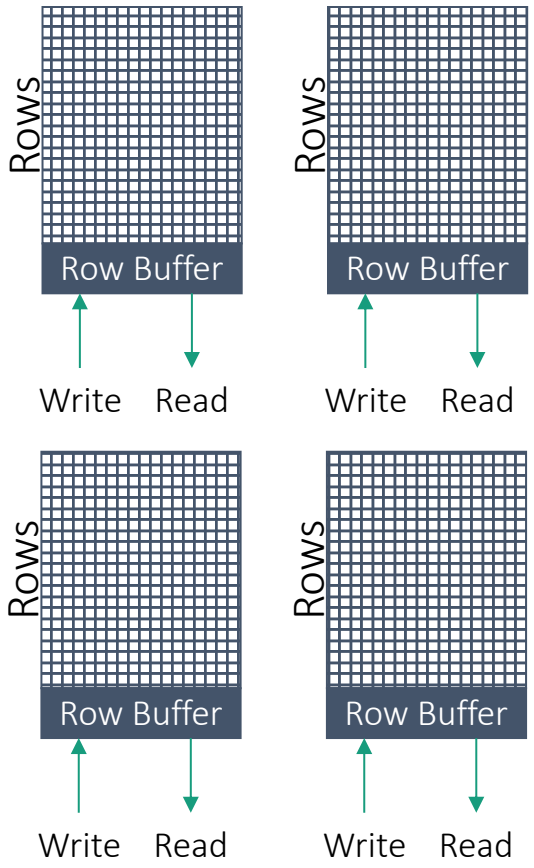
- In the beginning we are IDLE state
- If we want to read or write we have to be in the ACTIVE state.
- A specific row is stored in the bank's row buffer (ACT)
- Then we can perform read (RD) or write (WR) operations
- If we want to read data from another row we have to close the current row (PRE)

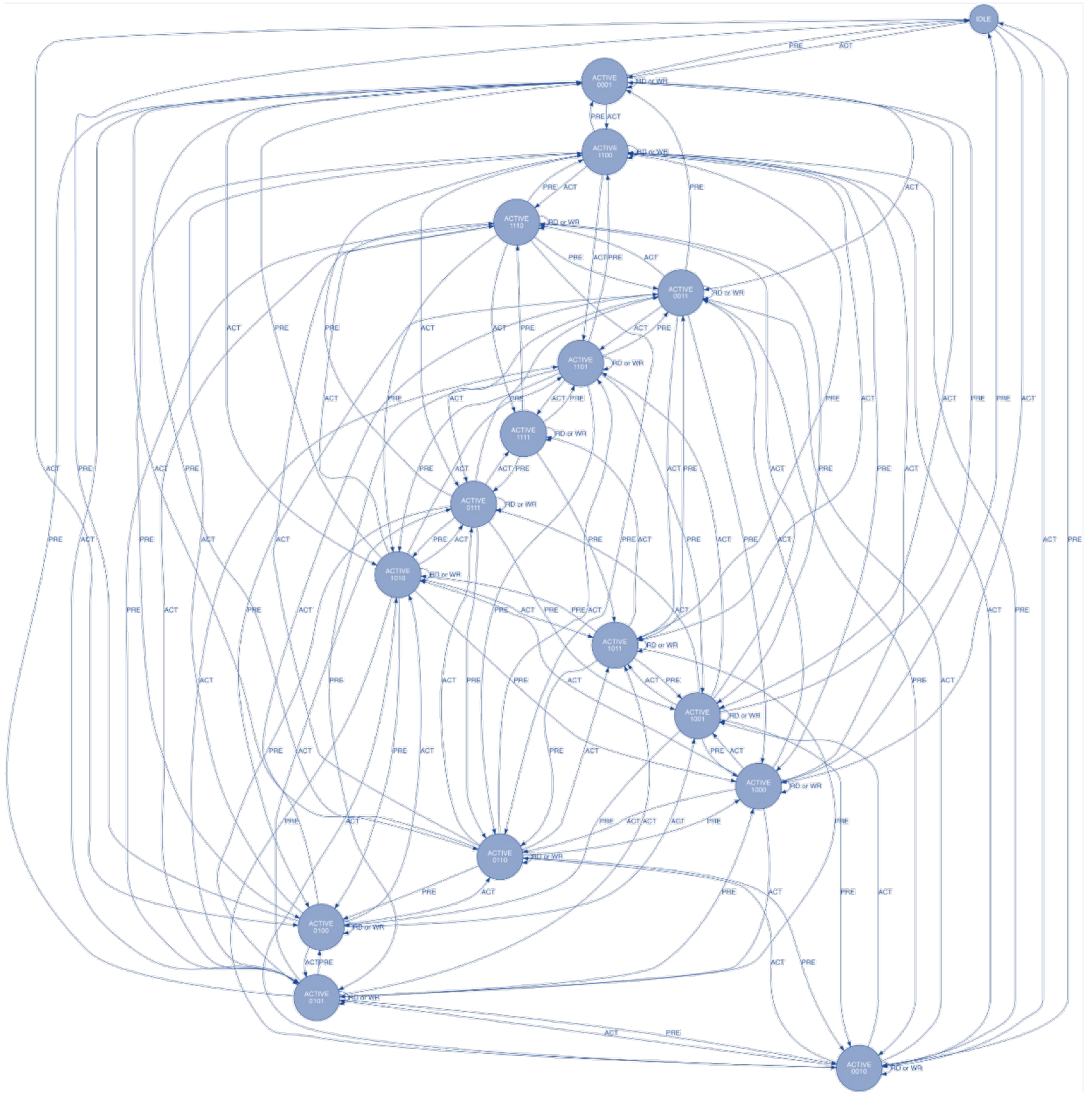
Two Memory Banks



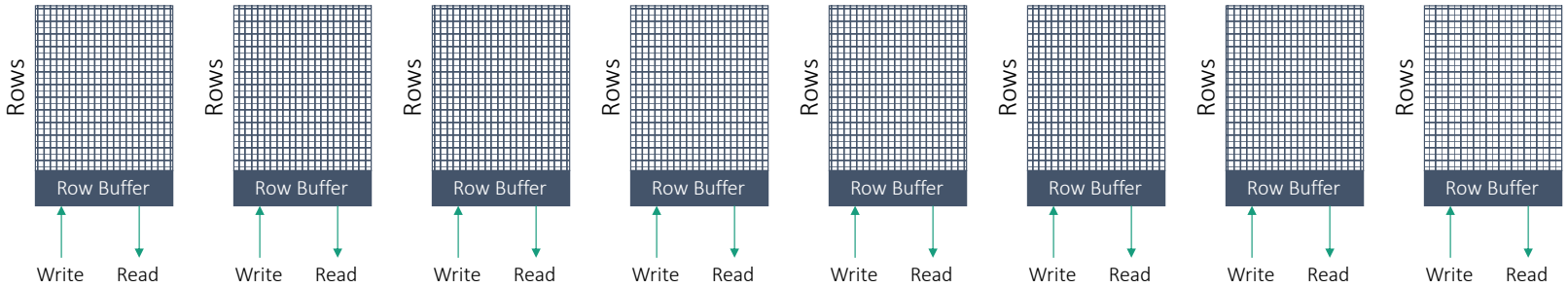


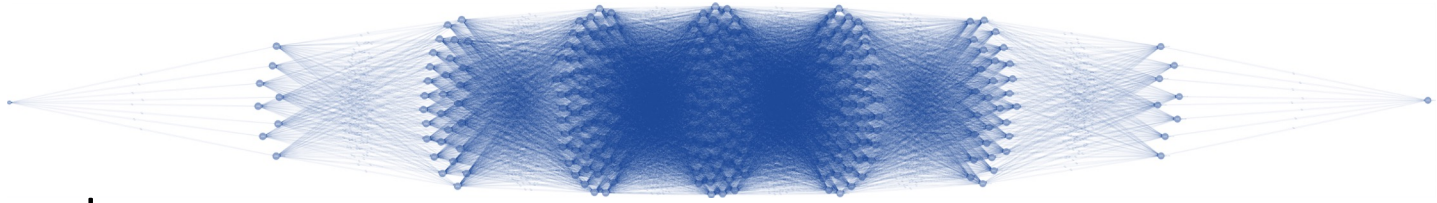
Four Memory Banks





8 Memory Banks





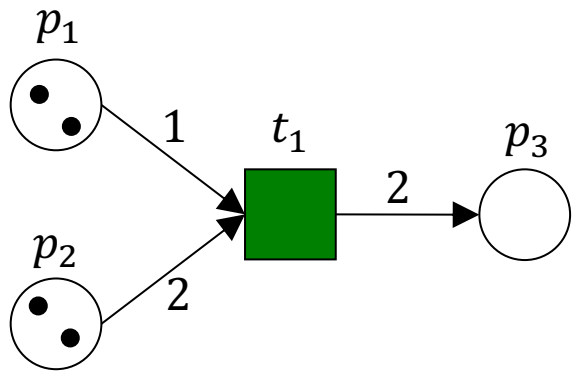
- DDR3 Memory
- State Explosion can happen!
  - Number of banks: B
  - Number of states/nodes: 2^B
  - Number of edges: $2 \cdot \binom{2 \cdot B}{B-1} + 2^B - 1$

Recap: Petri Nets

Petri Nets

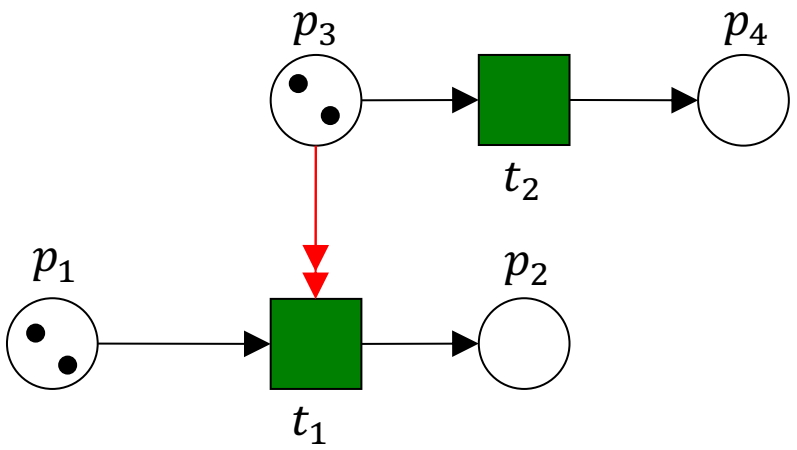
Petri Nets are models for concurrent asynchronous systems:

- Places, which usually represent states
- Transitions, which usually represent events
- Arcs
  - Connect places and transitions
  - Can have a weight
- Tokens
- Firing rules:
  - Enabled if $\#tokens \geq weight$
  - Enabled transition may fire
  - Firing removes and generates new tokens, transition may be disabled

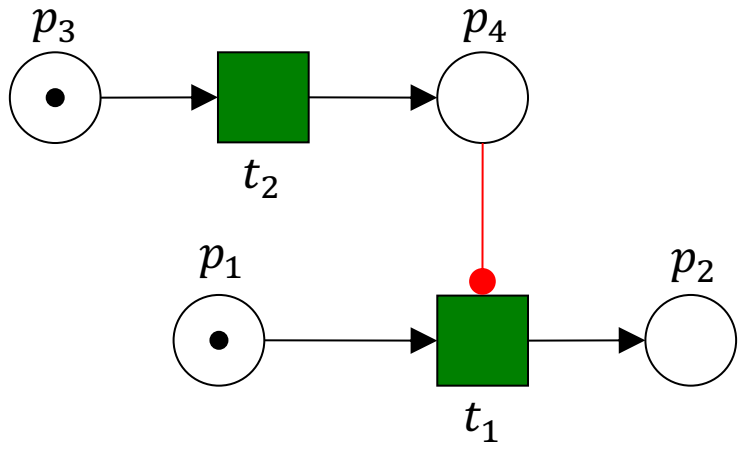


Petri Net Extensions

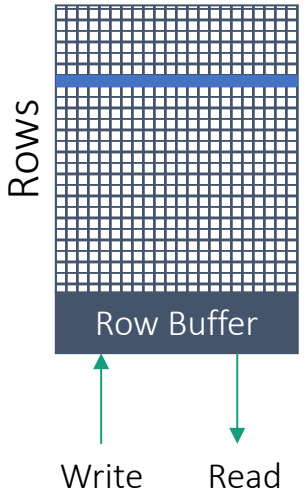
Reset Arcs:

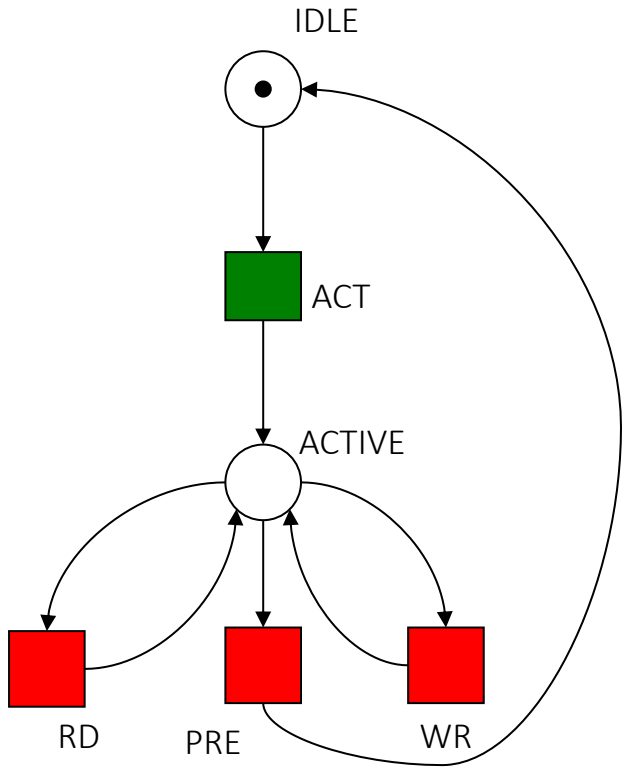


Inhibitor Arcs:



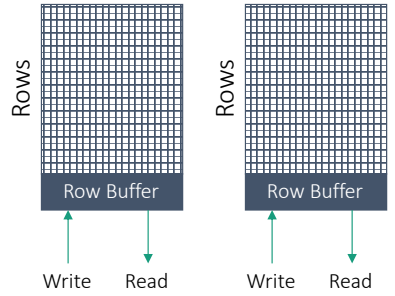
Modeling DRAMs with Petrinets

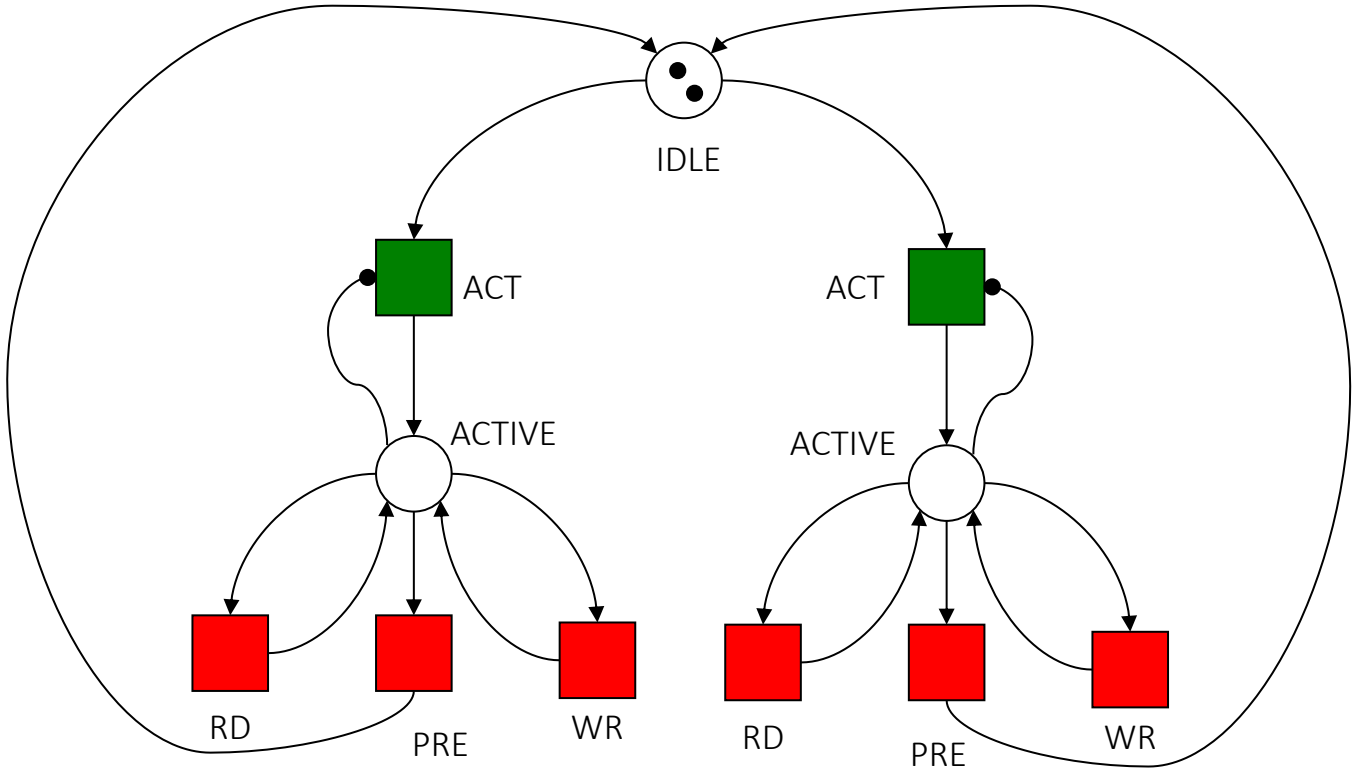




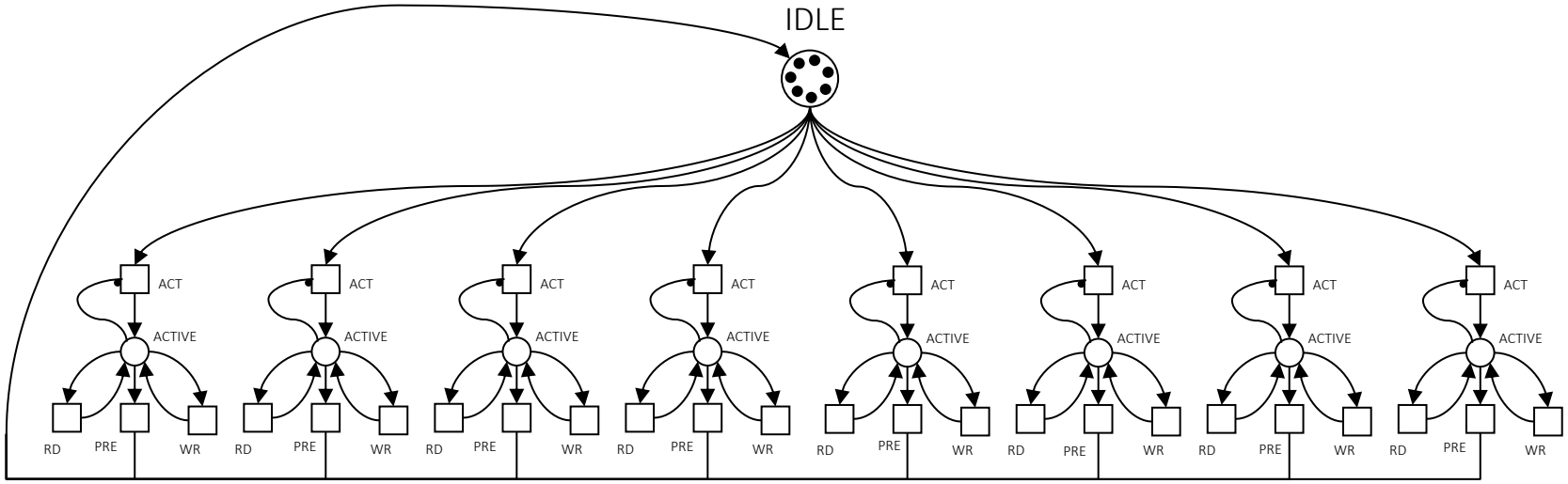
- State modeled as place
- DRAM command modeled as transition
- In the beginning we are IDLE state
- If we want to read or write we must be in the ACTIVE state.
- A specific row is stored in the bank's row buffer (ACT)
- Then we can perform read (RD) or write (WR) operations
- If we want to read data from another row we must close the current row and precharge the bitlines (PRE)

Two Memory Banks

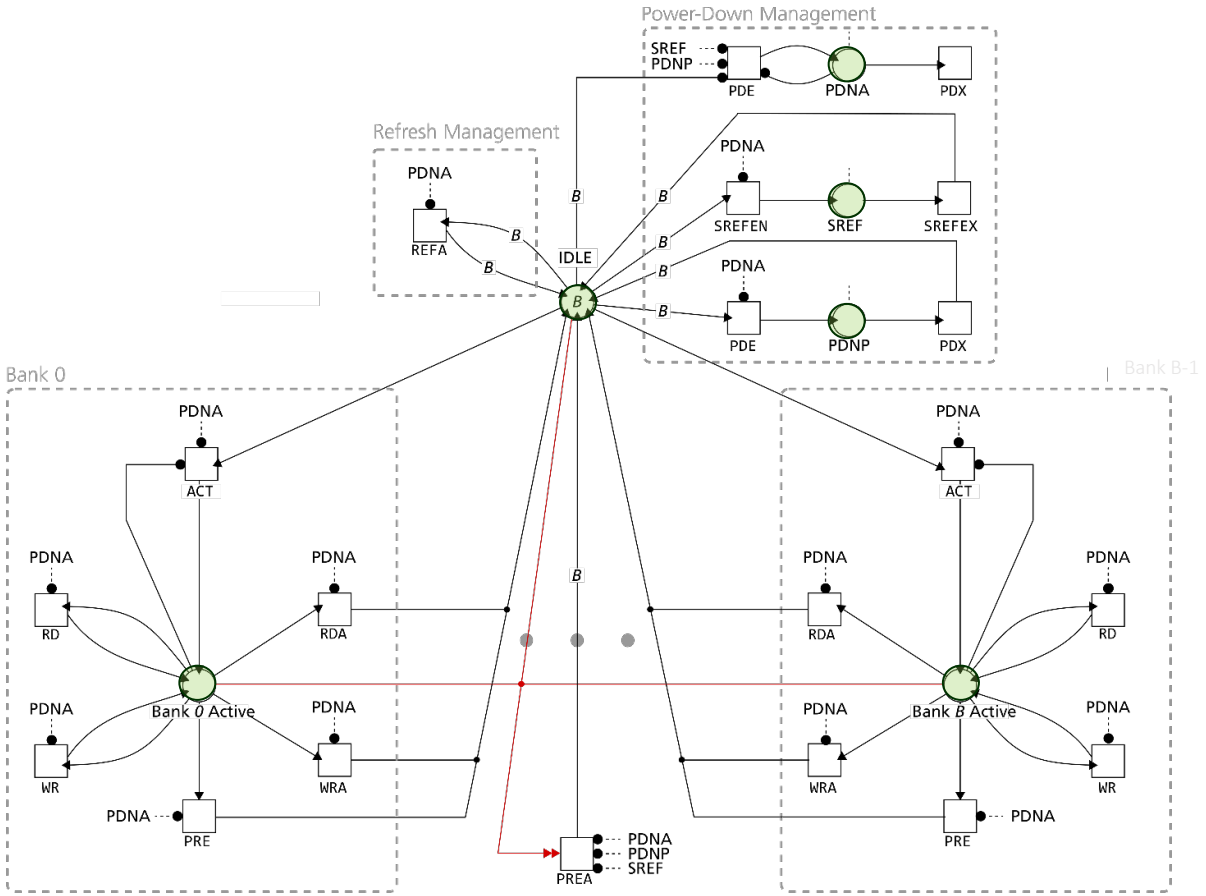




8 Memory Banks

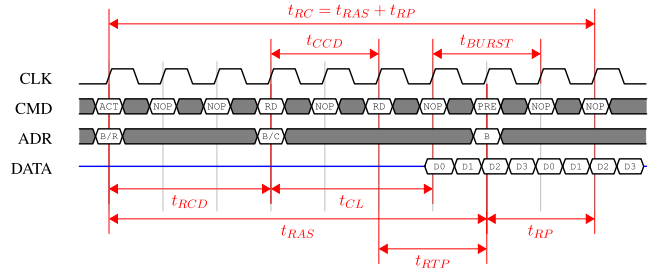


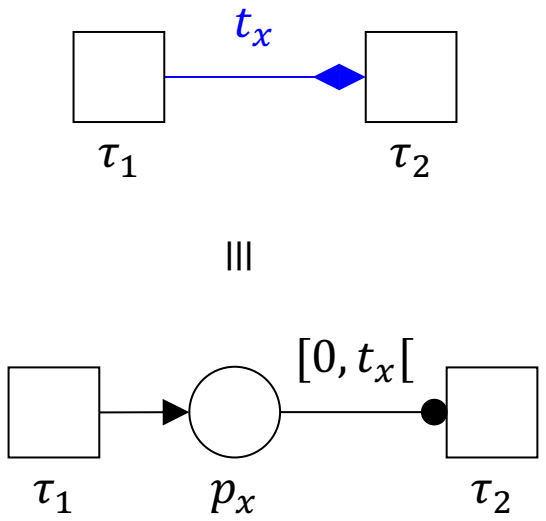
Modeling All DRAM States and Commands



- Comprehensive Model with clear separation between states and commands
- Models only 5 state types
- Support of multiple banks i.e. bank parallelism
- Divided in several subnets:
  - Banks
  - Refresh
  - Power-Down
  - Bankgroups
  - Ranks

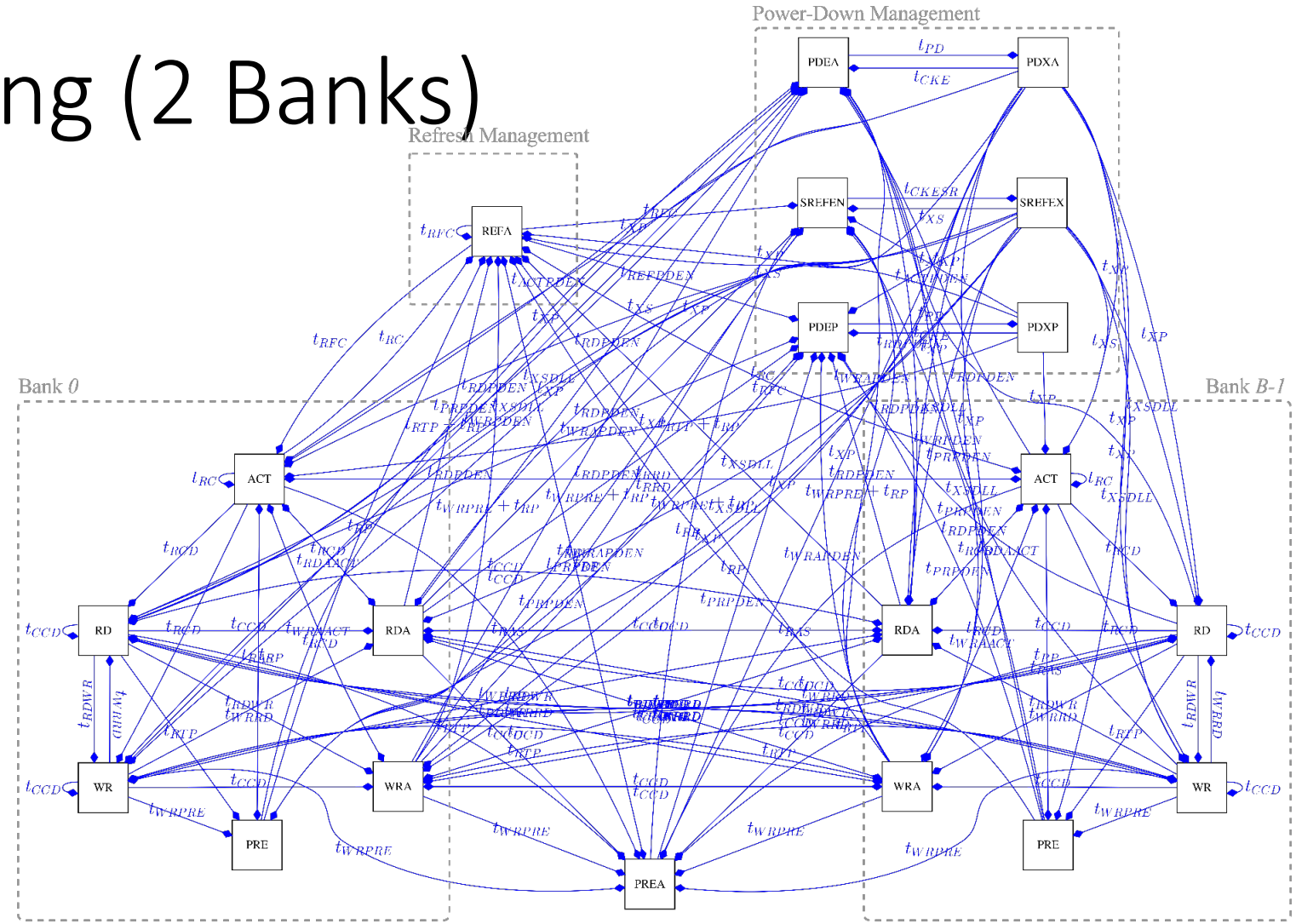
Modeling DRAM Timing



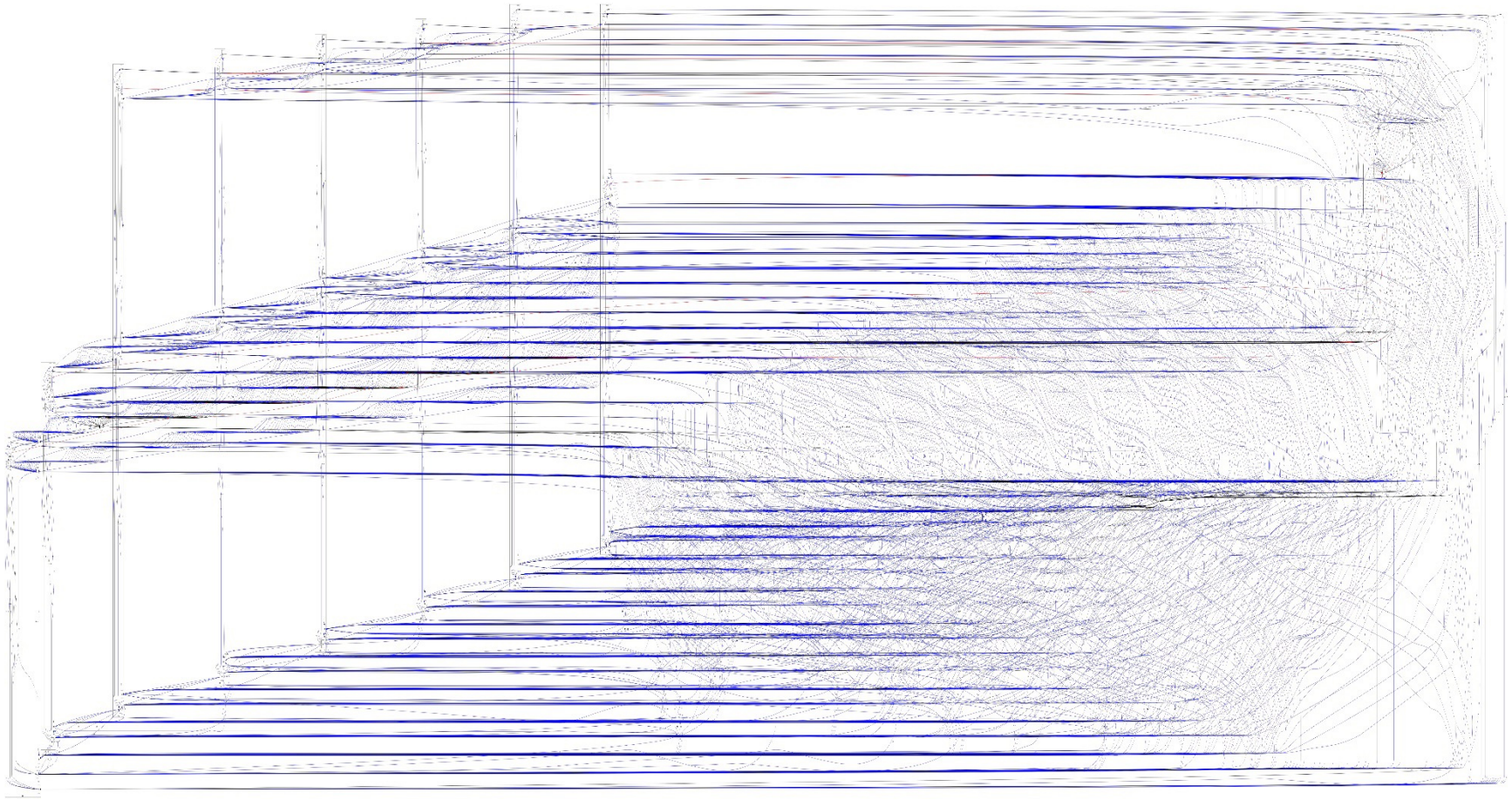


- DRAMs feature a complex timing protocol
- E.g. 90 out of 260 pages of DDR3 standard are showing timing diagrams and explanations for the timings.
- DRAM command timing dependencies can be modeled by a timed inhibitor arc:
- For example, **ACT** to **RD** would be t_{RCD}

Modeling Timing (2 Banks)

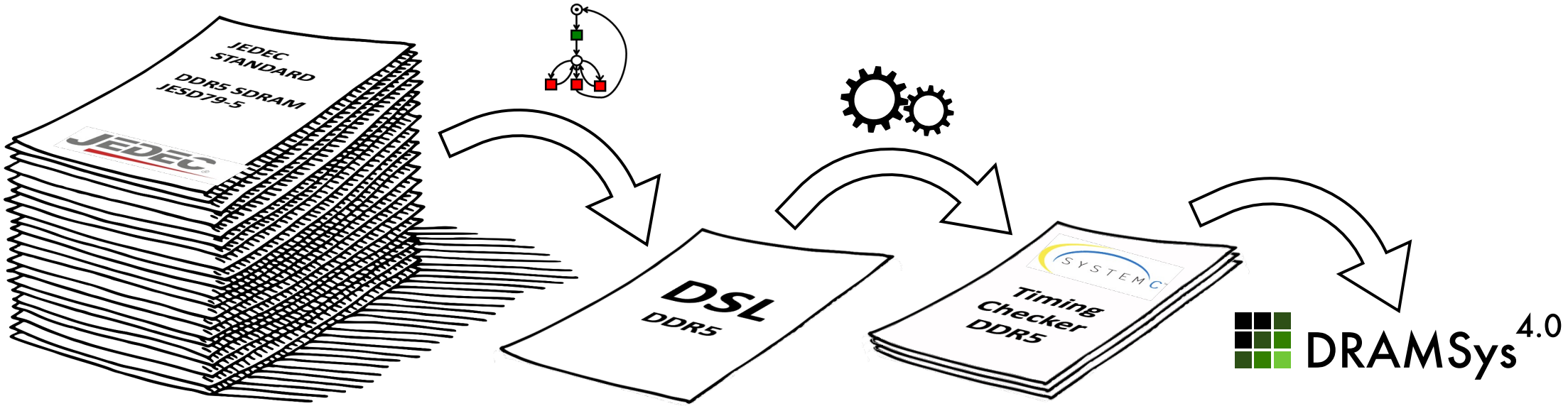


Modeling DRAM Timing (8 Banks)



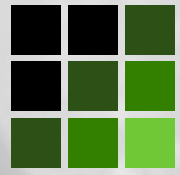
Code Generation and Validation

- Timed Petri nets allow a formal representation of a DRAM protocol, however, no graphical handling possible
- DRAMml is a DSL to describe DRAM's behavior with a petri net semantic
- DSL is as a basis for correct-by-construction DRAMSys TLM code generation
- For example: from DDR5 release it took 2 weeks to implement the model



DDR3-800

```
DRAM {
  Name = "DDR3-800D";
  B = 8;
  N = 4;

  Device {
    Places {
      IDLE (B);
      PDNP;
      SREF;
      PDNA;
    }

    Transitions {
      REFA;
      PREA;
      PDEP;
      PDXP;
      SREFEN;
      SREFEX;
      PDEA;
      PDXA;
    }

    Arcs {
      // Normal Arcs:
      IDLE   -> REFA   (Weight = B);
      REFA   -> IDLE   (Weight = B);
      PREA   -> IDLE   (Weight = B);
      IDLE   -> PDEP   (Weight = B);
      PDEP   -> PDNP;
      PDNP   -> PDXP;
      PDXP   -> IDLE   (Weight = B);
      IDLE   -> SREFEN (Weight = B);
      SREFEN -> SREF;
      SREF   -> SREFEX;
      SREFEX -> IDLE   (Weight = B);
      PDEA   -> PDNA;
      PDNA   -> PDXA;

      IDLE -> ACT;
      RDA  -> IDLE;
      WRA  -> IDLE;
      PRE  -> IDLE;

      // Inhibitor Arcs:
      PDNA -o REFA;
      PDNA -o PREA;
      PDNP -o PREA;
      SREF -o PREA;
      PDNA -o PDEP;
      PDNA -o SREFEN;
      PDNA -o PDEA;
      PDNP -o PDEA;
      SREF -o PDEA;
      IDLE -o PDEA (Weight = B);

      PDNA -o ACT;
      PDNA -o RD;
      PDNA -o WR;
      PDNA -o PRE;
      PDNA -o RDA;
      PDNA -o WRA;


      ACTIVE ->> PREA;
      IDLE   ->> PREA;
    }


    B : Bank {
      Places {
        ACTIVE;
      }

      Transitions {
        ACT;
        RD;
        RDA;
        PRE;
        WR;
        WRA;
      }

      Arcs {

        ACT    -> ACTIVE;
        ACTIVE -> RD;
        RD     -> ACTIVE;
        ACTIVE -> RDA;
        ACTIVE -> WR;
        WR     -> ACTIVE;
        ACTIVE -> WRA;
        ACTIVE -> PRE;


        ACTIVE -o ACT;
      }
    }
  }

  TimingConstraints {
    Timings {
      tCK     = 2.5;
      tCCD    = 10;
      tRCD    = 12.5;
      tRP     = 12.5;
      tRAS    = 37.5;
      tRL     = 5*tCK;
      tWL     = 5*tCK;
      tRTP    = 4*tCK;
      tWTR    = 4*tCK;
      tRRD    = 4*tCK;
      tWR     = 15;
      tFAW    = 40;
      tRFC    = 110;
      tREFI   = 7800;
      tREFMAX = 9*tREFI;

      tRC     = tRAS + tRP;
      tRDWR   = tRL+tCCD+2*tCK-tWL;
      tWRRD   = tWL + tCCD + tWTR;
      tRDAACT = tRTP + tRP;
      tWRPRE  = tWL + tCCD + tWR;
      tWRAACT = tWRPRE + tRP;

      tXP     = 3*tCK;
      tXS     = tRFC + 10;
      tXSDLL  = 512*tCK;
      tCKE    = 3*tCK;
      tCKESR  = tCKE + tCK;
      tPD     = tCKE;
      tRDPDEN = tRL + 5*tCK;
      tWRPDEN = tWL + 4*tCK + tWR;
      tWRAPDEN= tWL + 5*tCK + tWR;
      tREFPDEN= tCK;
      tACTPDEN= tCK;
      tPRPDEN = tCK;
    }

    Places {
      CMD_BUS;
      NAW_POOL (N);
    }

    Arcs {
      ACT -<> PRE  (tRAS,0);
      ACT -<> RD   (tRCD,0);
      ACT -<> WR   (tRCD,0);
      ACT -<> RDA  (tRCD,0);
      ACT -<> WRA  (tRCD,0);
      ACT -<> ACT  (tRC,tRRD);
      ACT -<> PDEA (0,tACTPDEN);
      ACT -<> REFA (0,tRC);
      ACT -<> PREA (0,tRAS);

      RD  -<> PRE    (tRTP,0);
      RD  -<> PREA   (0,tRTP);
      RD  -<> PDEA   (0,tRDPDEN);
      RD  -<> PDEP   (0,tRDPDEN);
      RDA -<> PDEA   (0,tRDPDEN);
      RDA -<> PDEP   (0,tRDPDEN);
      RD  -<> RD     (tCCD,tCCD);
      RD  -<> RDA    (tCCD,tCCD);
      RDA -<> RD     (0,tCCD);
      RDA -<> RDA    (0,tCCD);
      RD  -<> WR     (tRDWR,tRDWR);
      RD  -<> WRA    (tRDWR,tRDWR);
      RDA -<> WR     (0,tRDWR);
      RDA -<> WRA    (0,tRDWR);
      RDA -<> ACT    (tRDAACT,0);
      RDA -<> REFA   (0,tRTP+tRP);
      RDA -<> PREA   (0,tRTP);
      RDA -<> SREFEN (0,tRDPDEN);

      WR  -<> PRE    (tWRPRE,0);
      WR  -<> PREA   (0,tWRPRE);
      WR  -<> PDEA   (0,tWRPDEN);
      WRA -<> PDEA   (0,tWRAPDEN);
      WRA -<> PDEP   (0,tWRAPDEN);
      WR  -<> WR     (tCCD,tCCD);
      WR  -<> WRA    (tCCD,tCCD);
      WRA -<> WR     (0,tCCD);
      WRA -<> WRA    (0,tCCD);
      WR  -<> RD     (tWRRD,tWRRD);
      WR  -<> RDA    (tWRRD,tWRRD);
      WRA -<> RD     (0,tWRRD);
      WRA -<> RDA    (0,tWRRD);
      WRA -<> ACT    (tWRAACT,0);
      WRA -<> REFA   (0,tWRPRE+tRP);
      WRA -<> PREA   (0,tWRPRE);
      WRA -<> SREFEN (0,tWRPRE+tRP);

      PRE  -<> ACT    (tRP,0);
      PRE  -<> REFA   (0,tRP);
      PRE  -<> PDEA   (0,tPRPDEN);
      PRE  -<> PDEP   (0,tPRPDEN);
      PRE  -<> SREFEN (0,tRP);
      PREA -<> ACT    (0,tRP);
      PREA -<> REFA   (0,tRP);
      PREA -<> PDEP   (0,tPRPDEN);
      PREA -<> SREFEN (0,tRP);

      PDEP -<> PDXP   (0,tPD);
      PDEA -<> PDXA   (0,tPD);
      PDXA -<> PDEA   (0,tCKE);
      PDXP -<> PDEP   (0,tCKE);
      PDXP -<> REFA   (0,tXP);
      PDXP -<> SREFEN (0,tXP);
      PDXA -<> ACT    (0,tXP);
      PDXP -<> ACT    (0,tXP);
      PDXA -<> PRE    (0,tXP);
      PDXA -<> PREA   (0,tXP);
      PDXA -<> RD     (0,tXP);
      PDXA -<> RDA    (0,tXP);
      PDXA -<> WR     (0,tXP);
      PDXA -<> WRA    (0,tXP);

      REFA   -<> ACT    (0,tRFC);
      REFA   -<> REFA   (0,tRFC);
      REFA   -<> PREA    (0,tRFC);
      REFA   -<> SREFEN (0,tRFC);
      REFA   -<> PDEP   (0,tREFPDEN);
      SREFEX -<> ACT    (0,tXS);
      SREFEX -<> REFA   (0,tXS);
      SREFEX -<> PDEP   (0,tXS);
      SREFEX -<> SREFEN (0,tXS);
      SREFEX -<> RD     (0,tXSDLL);
      SREFEX -<> RDA    (0,tXSDLL);
      SREFEX -<> WR     (0,tXSDLL);
      SREFEX -<> WRA    (0,tXSDLL);
      SREFEN -<> SREFEX (0,tCKESR);

      CMD_BUS -> * [tCK, inf];
      * -> CMD_BUS;

      NAW_POOL -> ACT [tFAW, inf];
      ACT      -> NAW_POOL;

      REFA   -<> WARNING (0,tREFMAX);
      SREF   -o  WARNING;
      SREFEX -<> WARNING (0,tREFMAX);
    }
  }
}
```

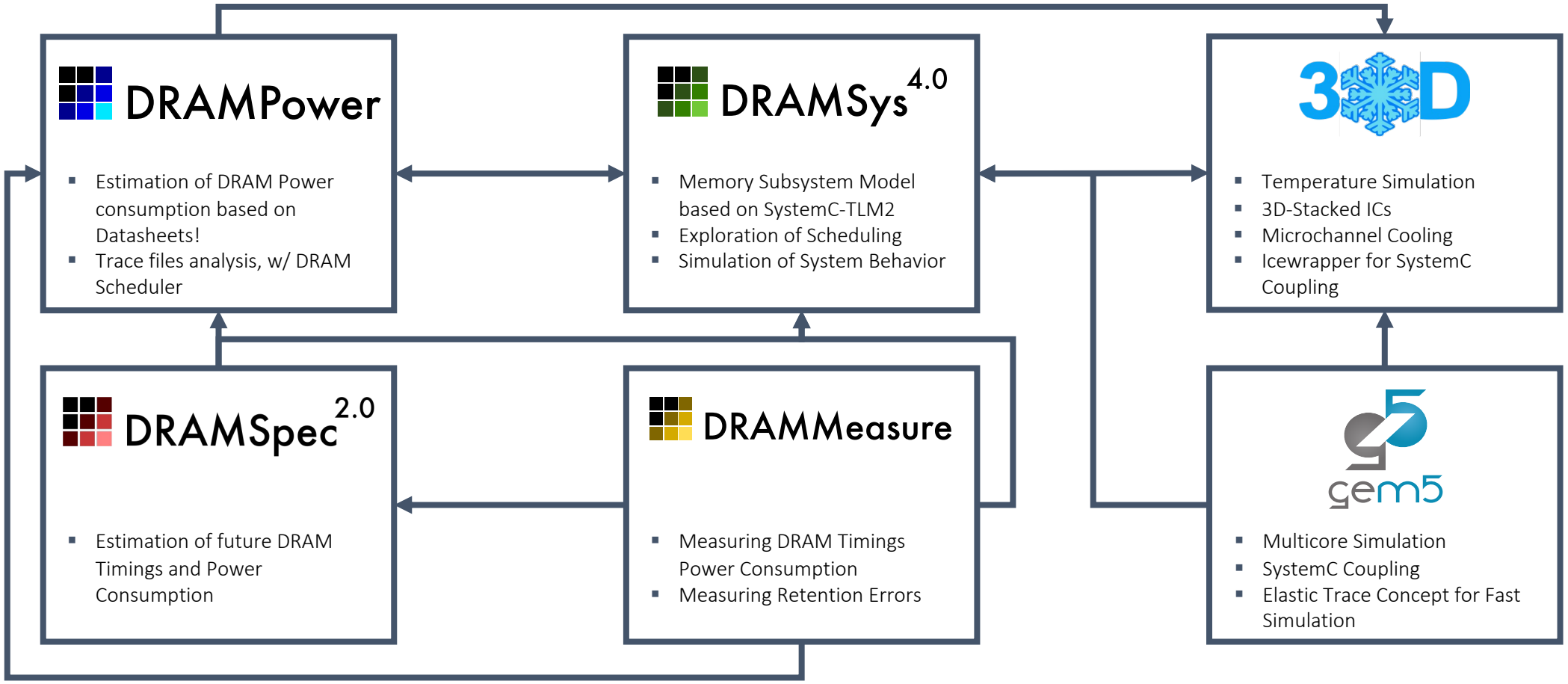


DRAMSys 4.0

Functional Models	Non-Functional	Analysis
TLM DRAMml	Power Thermal Errors	Trace Analyzer

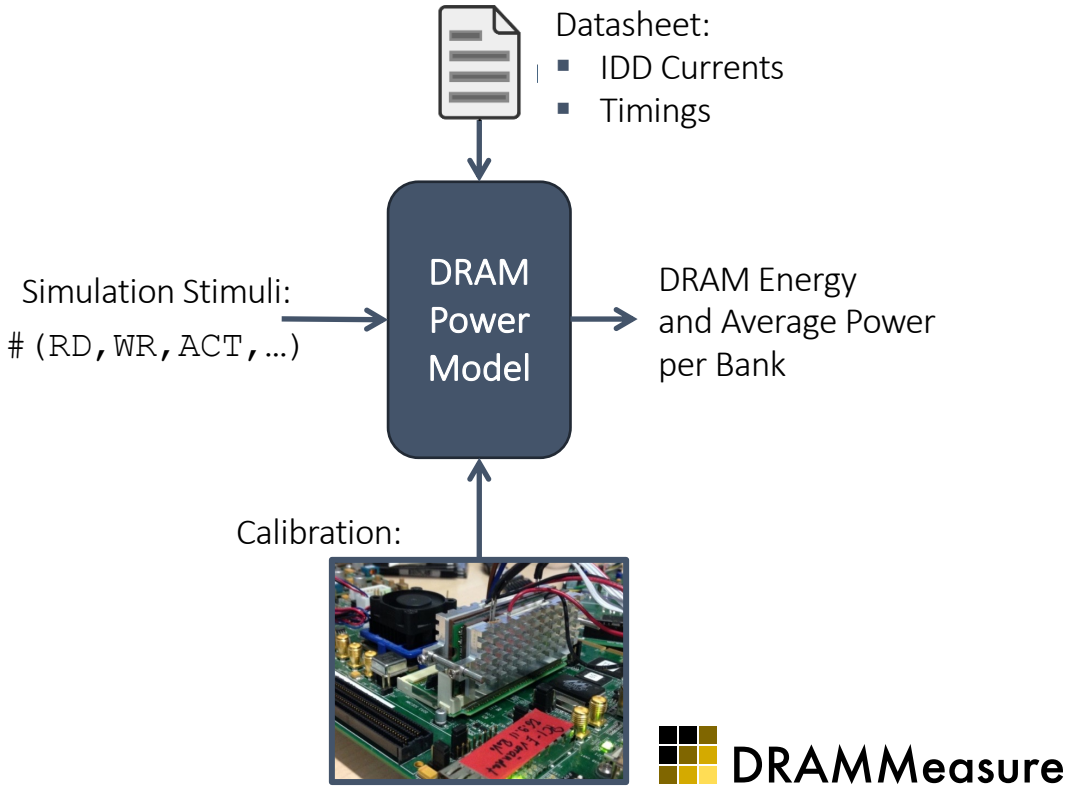
accellera
SYSTEMS INITIATIVE

Overview (Our) Tools



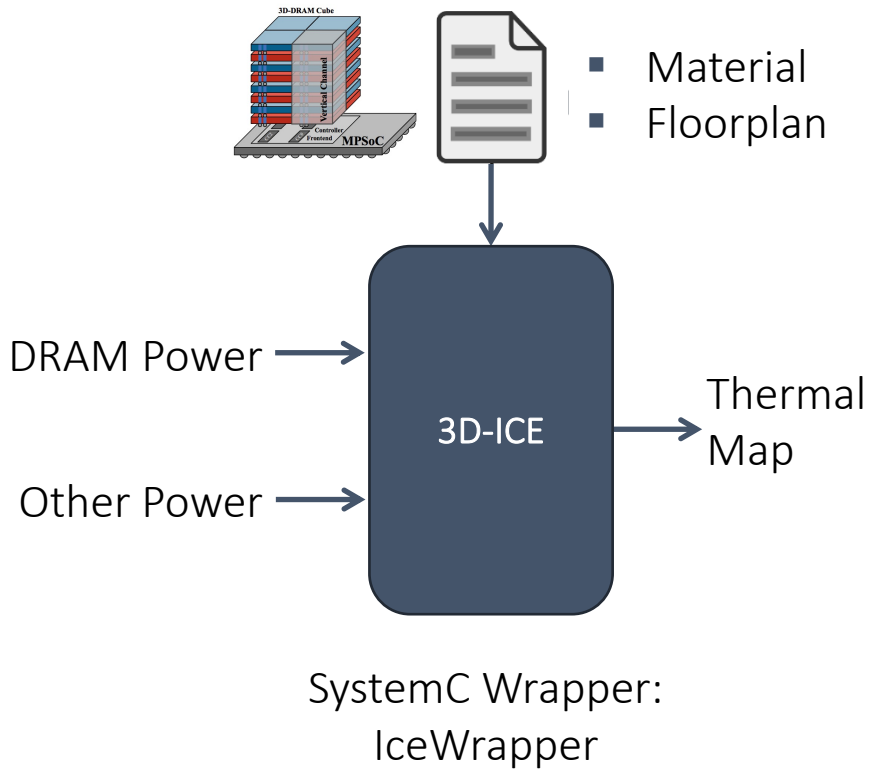
DRAMPower

TU/e DRAMPower

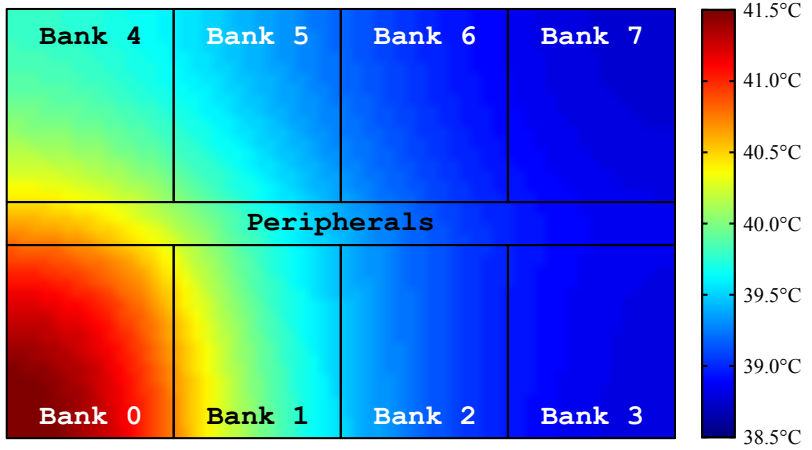


- State-of-the-art: Micron Model (Pessimistic)
- Highly Accurate DRAM power model developed together with TU/e
- Bank-wise calculation
- Temperature dependency
- Widely used by Industry

3D-ICE

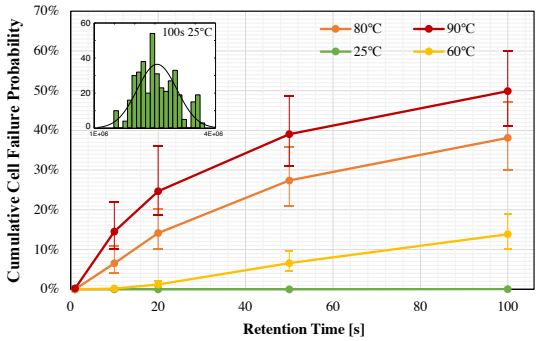


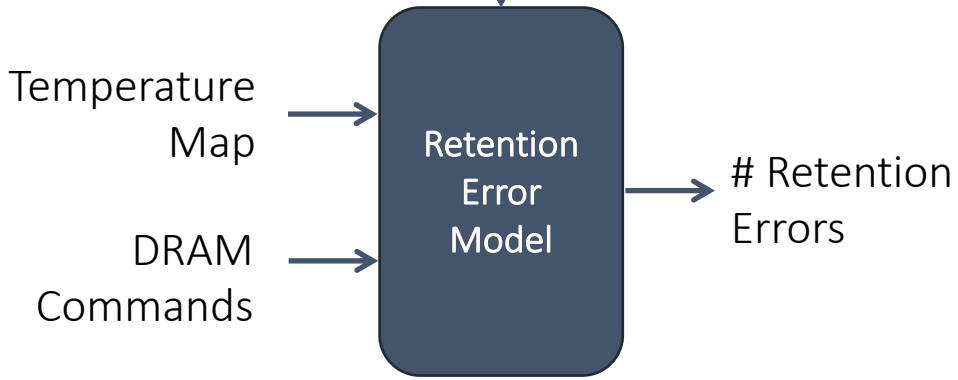
- Thermal RC networks
- Solving differential equations
- Co-simulation with SystemC



Retention Error Model

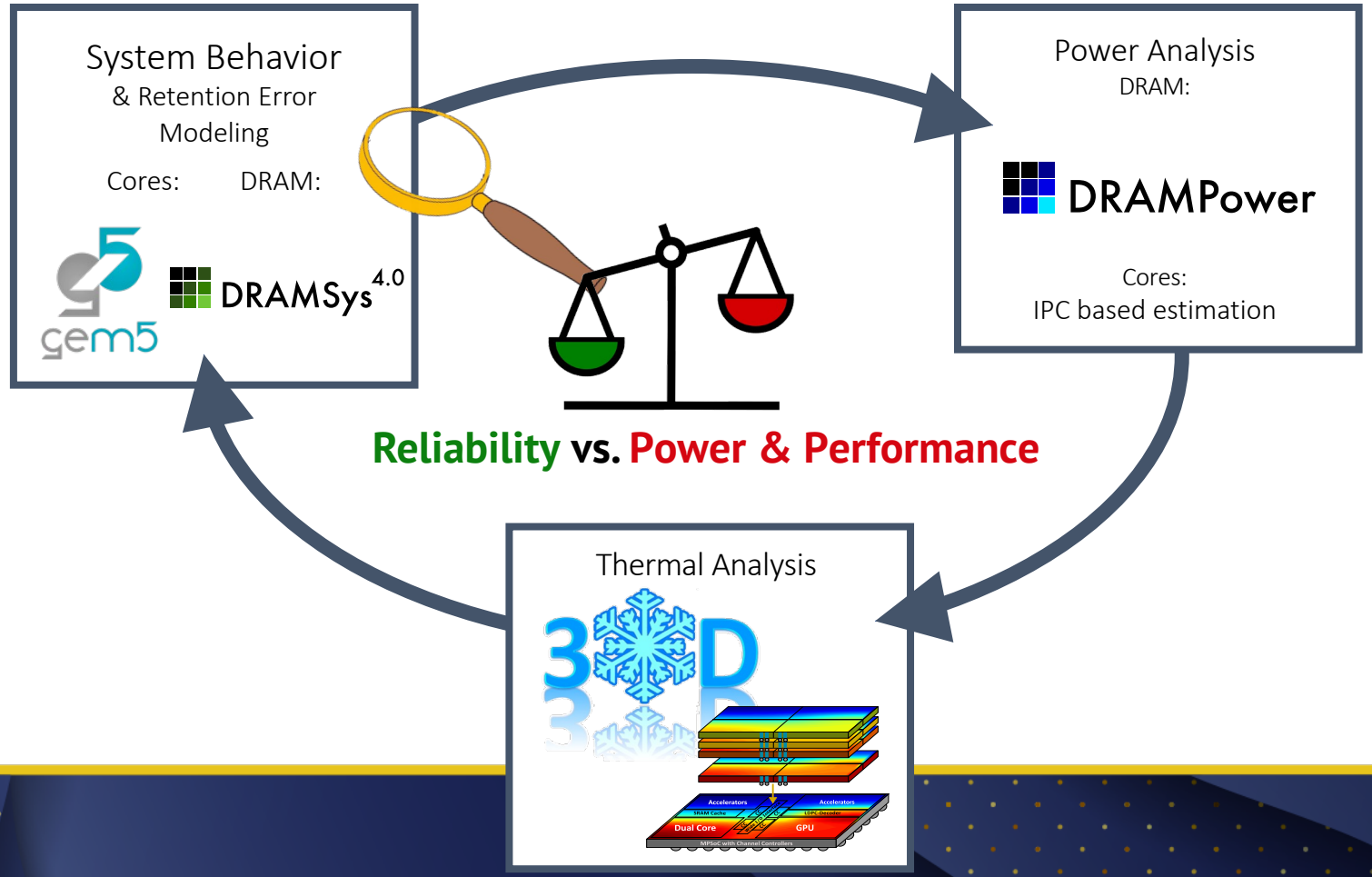
DRAMMeasure





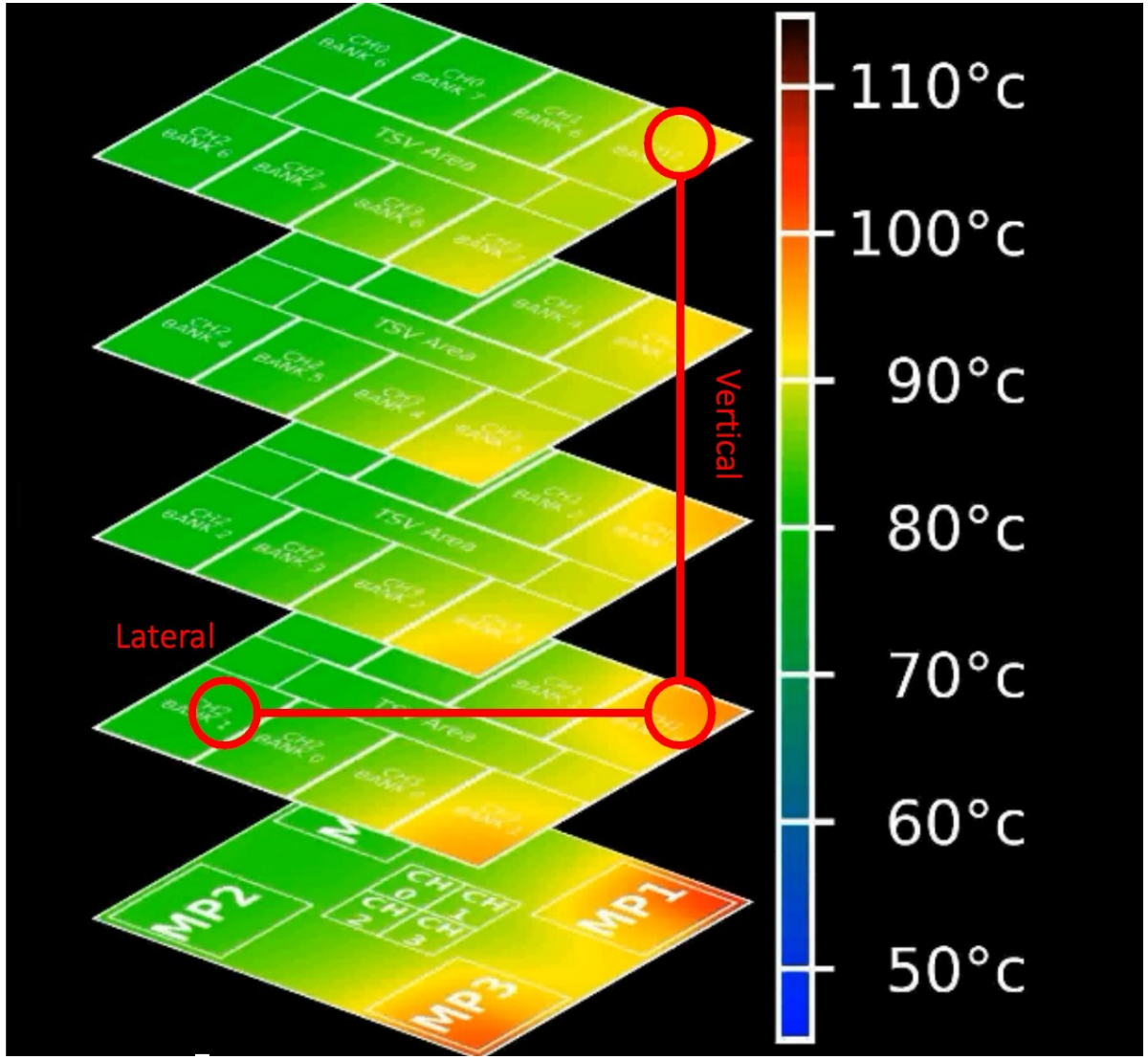
- When not refreshing correctly
- DDR3 & Wide I/O
- Calibrated with Measurements

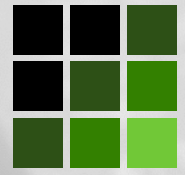
Holistic Simulation



Simulation of an Android Smart Phone

- 3D MPSoC with 4 ARM Cortex A9
- 4 channel stacked Wide I/O DRAM
- Running Android OS
- *Dynamic Voltage and Frequency Scaling* (DVFS) for the Cores
- Closed loop simulation
- Temperature sensors for each bank and core



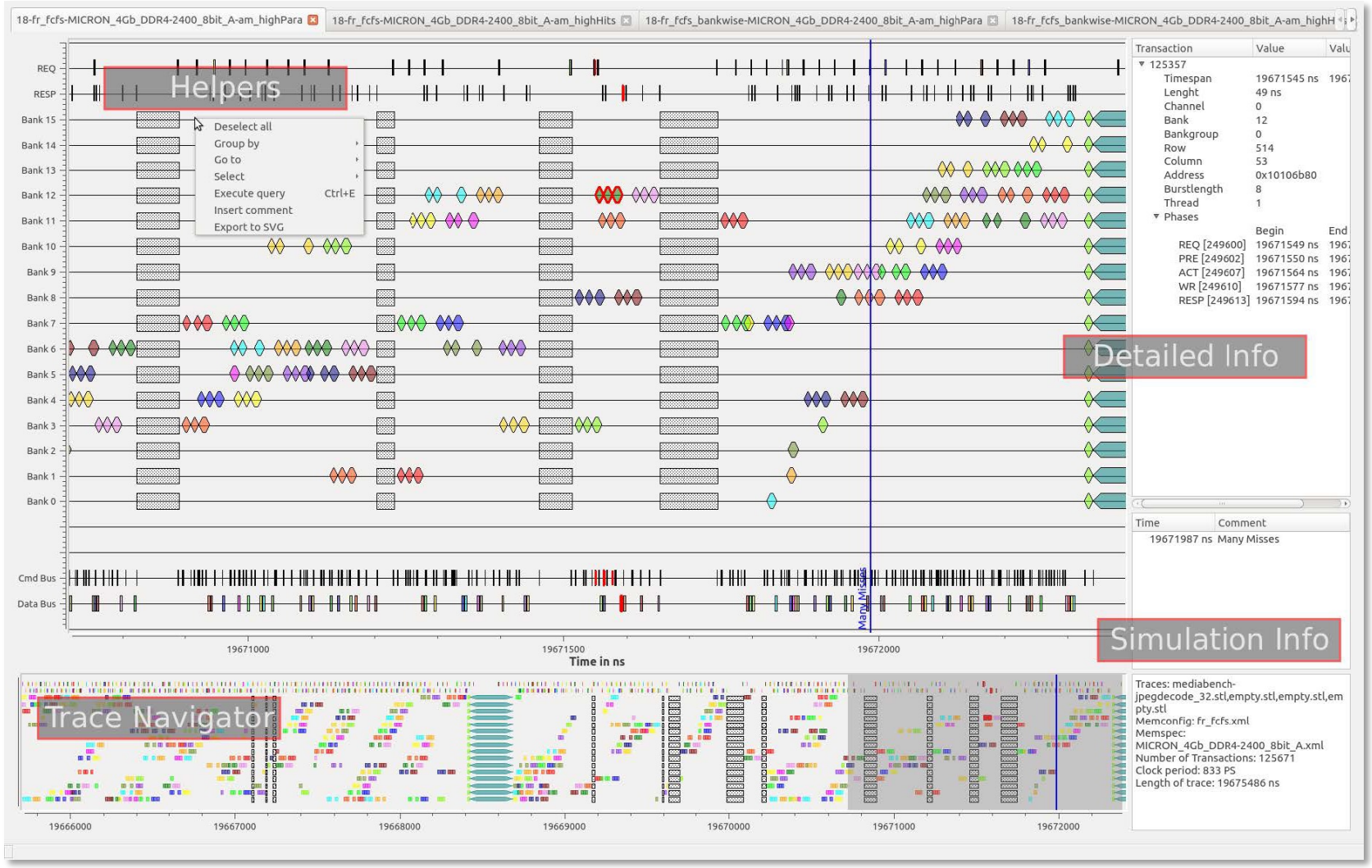


DRAMSys 4.0

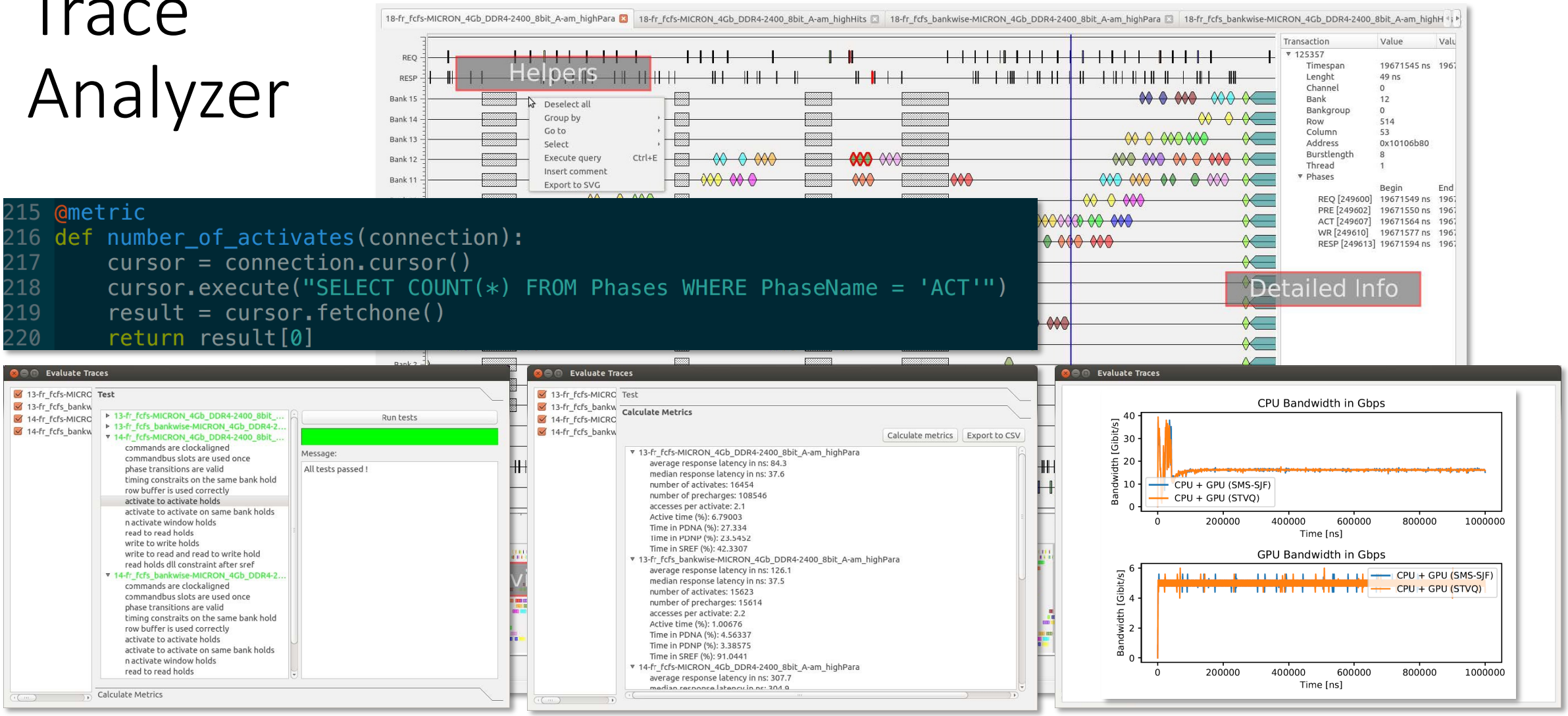
Functional Models	Non-Functional	Analysis
TLM DRAMml	Power Thermal Errors	Trace Analyzer

accellera
SYSTEMS INITIATIVE

Trace Analyzer



Trace Analyzer



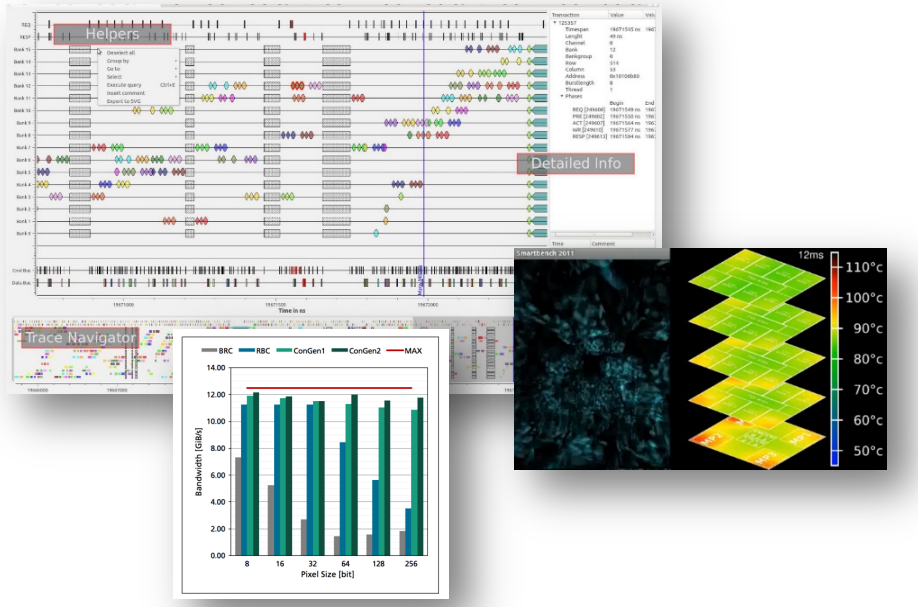
Where to get the tools:

DRAMSys:
https://github.com/tukl-msd/DRAMSys

gem5:
https://gem5.googlesource.com/

DRAMPower:
https://github.com/tukl-msd/DRAMPower

3D-ICE:
https://github.com/esl-epfl/3d-ice



2022
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
EUROPE
MUNICH, GERMANY
DECEMBER 6 - 7, 2022

The Open Source DRAM Simulator DRAMSys4.0

Dr. Matthias Jung, Fraunhofer IESE

Fraunhofer IESE

accellera SYSTEMS INITIATIVE