TECHNISCHE UNIVERSITÄT DRESDEN

Vodafone Chair Mobile Communications Systems, Prof. Gerhard Fettweis

vodafone chair

Data Plane Framework for Software Defined Radio Access Networks

Emil Matus
TU Dresden

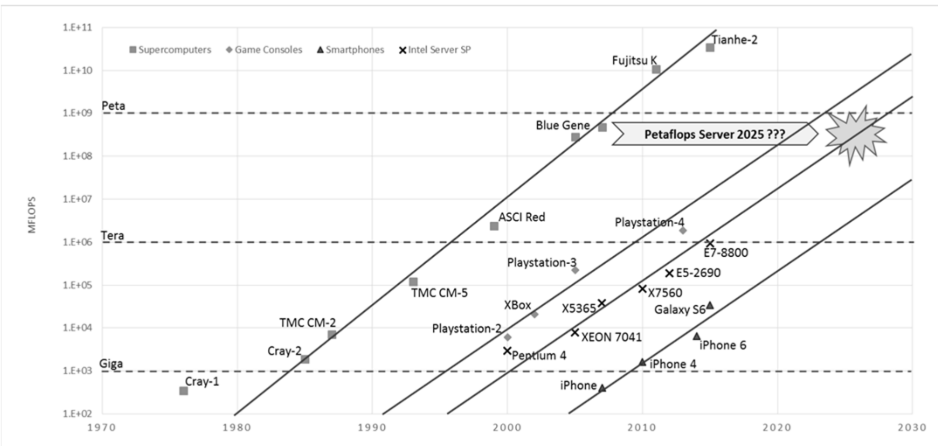
MPSoC Forum
Ventura Beach Marriott, CA, USA
July 13-17, 2015

Processing Power Evolution

TECHNISCHE UNIVERSITÄT DRESDEN

vodafone chair

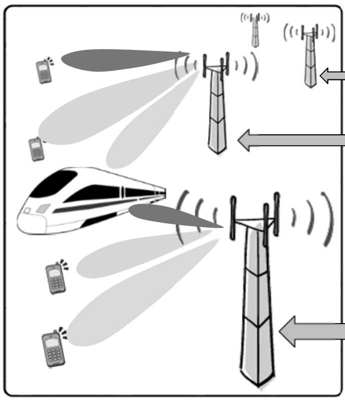
What are the implications for wireless infrastructure if this trend will continue?

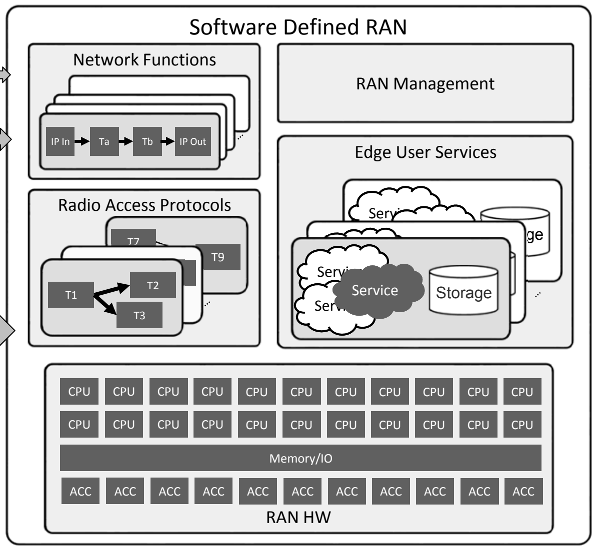


Flexible RAN Architecture

TECHNISCHE UNIVERSITÄT DRESDEN

vodafone chair





Open, software defined RAN:

- Radio (L1-L3)
- Network functions (L4-L6)
- User services (L8)

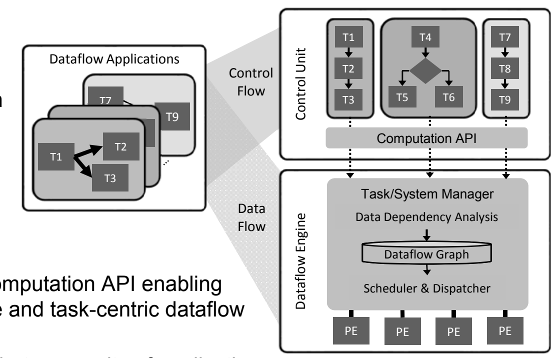
TU Dresden | E.Matus - MPSoC 2015 | Slide 3

Dataflow Computation Model …

TECHNISCHE UNIVERSITÄT DRESDEN

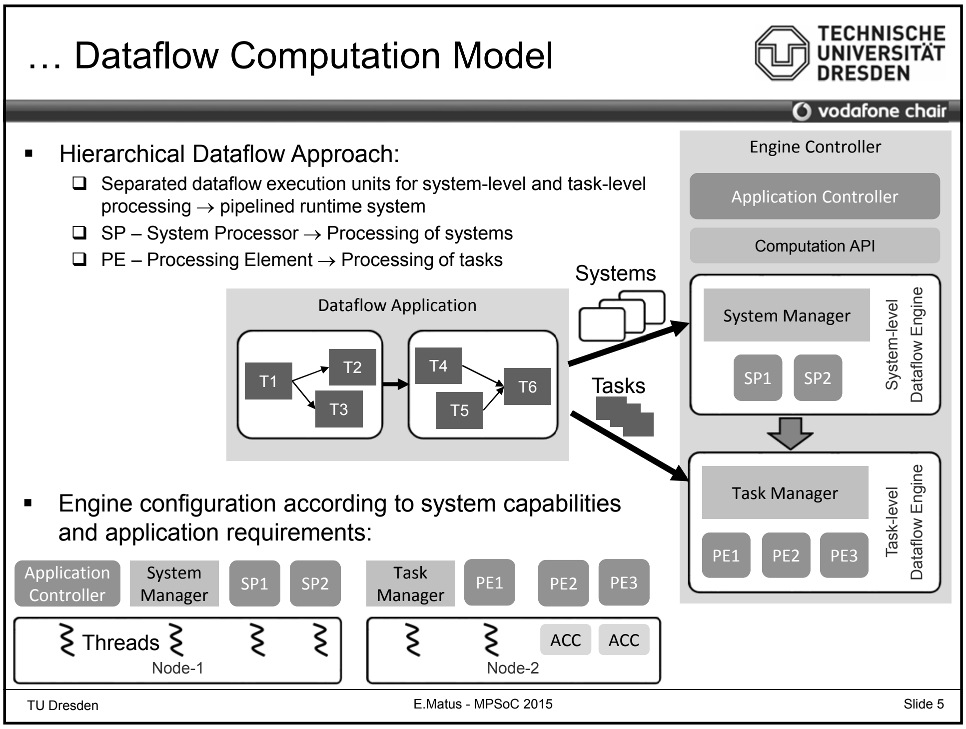
vodafone chair

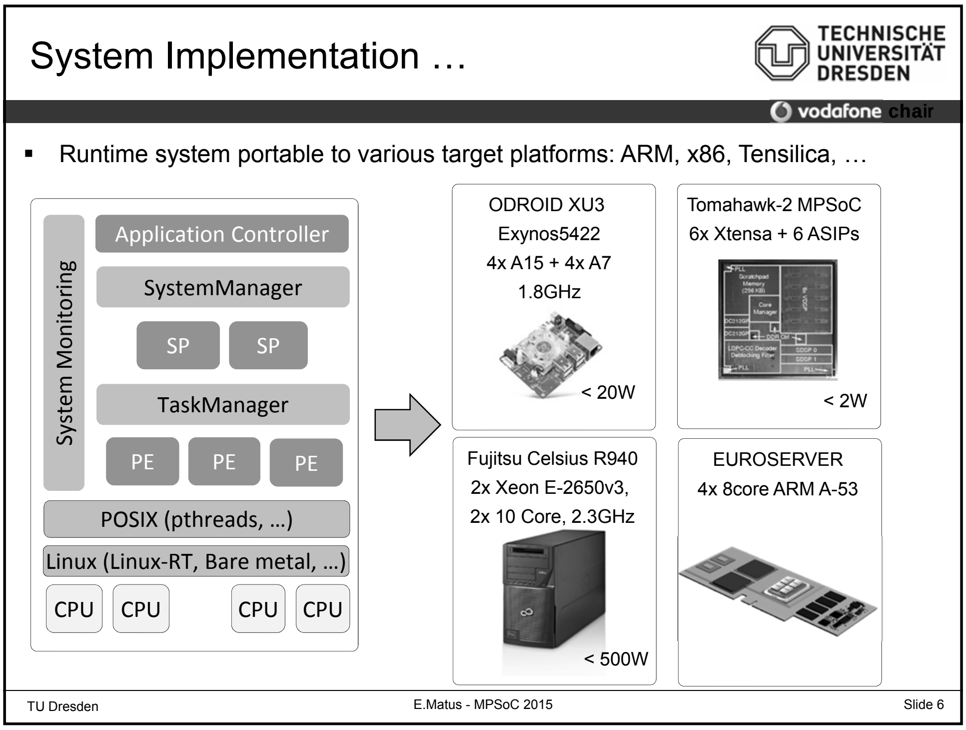
- Dataflow computation model well suited for most challenging parts:
  - Network functions – packet processing
  - Data plane of L2-L3 – transport block processing
  - L1 Baseband - sync, de-/coding, de-/modulation, channel estimation, equalization, …
- Dataflow application:
  - Task: atomic unit of computation
  - System: composition of tasks and systems



- Motivation:
  - Modular SW architecture and computation API enabling flexible design of runtime engine and task-centric dataflow applications
  - Exploit inherent parallelism and heterogeneity of application
  - Support for hierarchy and data/control dependency

TU Dresden | E.Matus - MPSoC 2015 | Slide 4


… Dataflow Computation Model
TECHNISCHE UNIVERSITÄT DRESDEN
vodafone chair
Hierarchical Dataflow Approach:
Separated dataflow execution units for system-level and task-level processing → pipelined runtime system
SP – System Processor → Processing of systems
PE – Processing Element → Processing of tasks
Dataflow Application
T1
T2
T3
T4
T5
T6
Systems
Tasks
Engine Controller
Application Controller
Computation API
System Manager
SP1
SP2
System-level Dataflow Engine
Task Manager
PE1
PE2
PE3
Task-level Dataflow Engine
Engine configuration according to system capabilities and application requirements:
Application Controller
System Manager
SP1
SP2
Task Manager
PE1
PE2
PE3
Threads
Node-1
ACC
ACC
Node-2
TU Dresden
E.Matus - MPSoC 2015
Slide 5

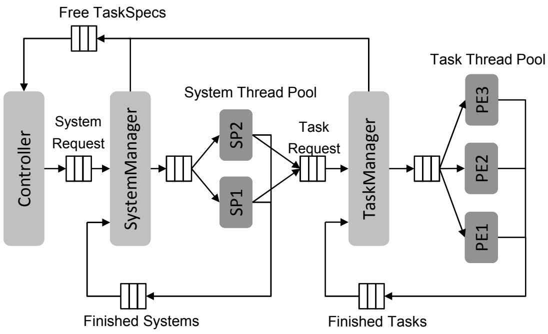

System Implementation …
TECHNISCHE UNIVERSITÄT DRESDEN
vodafone chair
Runtime system portable to various target platforms: ARM, x86, Tensilica, …
System Monitoring
Application Controller
SystemManager
SP
SP
TaskManager
PE
PE
PE
POSIX (pthreads, …)
Linux (Linux-RT, Bare metal, …)
CPU
CPU
CPU
CPU
ODROID XU3
Exynos5422
4x A15 + 4x A7
1.8GHz
< 20W
Tomahawk-2 MPSoC
6x Xtensa + 6 ASIPs
< 2W
Fujitsu Celsius R940
2x Xeon E-2650v3,
2x 10 Core, 2.3GHz
< 500W
EUROSERVER
4x 8core ARM A-53
TU Dresden
E.Matus - MPSoC 2015
Slide 6

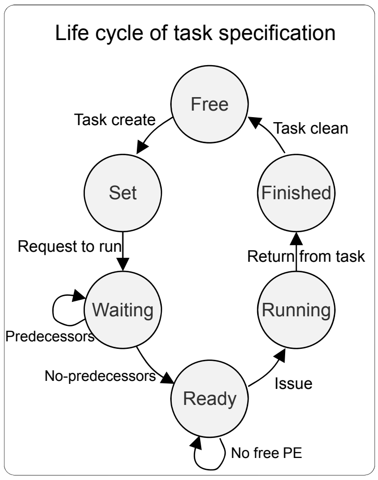
… System Implementation

TECHNISCHE UNIVERSITÄT DRESDEN

vodafone chair

- Pipelined message passing queuing system:
  - Message passing queues between system components (threads) with various arbitration mechanisms
  - Pre-allocated buffers and thread pools
  - Pre-allocated task specification structures
  - Optimized by thread priority and core affinity





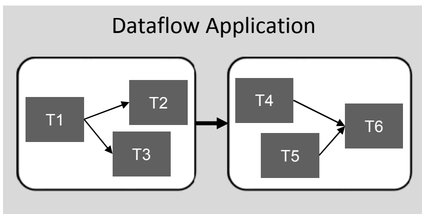
TU Dresden | E.Matus - MPSoC 2015 | Slide 7

Task-Centric Computation API

TECHNISCHE UNIVERSITÄT DRESDEN

vodafone chair

- TaskC - "Simple" and extensible task-centric programming interface enabling specification of:
  - Tasks and systems,
  - IN/OUT arguments,
  - Dependencies,
  - WCET ,
  - Deadline, Priority,
  - …



- Support for dynamic and/or static task-graph generation:
  - Definition by generator function → Dynamic dataflow
  - Definition by data structure → Static dataflow

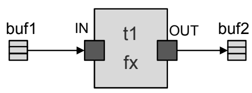
TU Dresden | E.Matus - MPSoC 2015 | Slide 8

Computation API …

TECHNISCHE UNIVERSITÄT DRESDEN

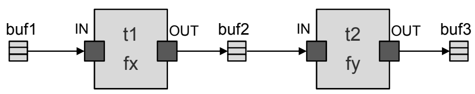
vodafone chair

- Create Task: **`TaskSpec* t = task(fnc, *arg1, *arg2,…);`**




```
TaskSpec* t1 =
   task(fx, IN(buf1, BUF1_SIZE), OUT(buf2, BUF2_SIZE));
set(t1, Property, Value);
push_task(t1);
```

- Task dependency:
  `predecessors(TaskSpec* dest_task, TaskSpec *source_task,…)`




```
TaskSpec* t1 = task(fx, IN(buf1, BUF1_SIZE), OUT(buf2, BUF2_SIZE));
TaskSpec* t2 = task(fy, IN(buf2, BUF2_SIZE), OUT(buf3, BUF3_SIZE));
predecessors(t2, t1);
push_task(t1); push_task(t2);
```

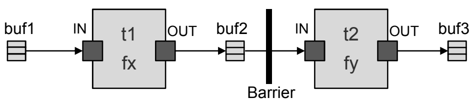
TU Dresden E.Matus - MPSoC 2015 Slide 9

… Computation API …

TECHNISCHE UNIVERSITÄT DRESDEN

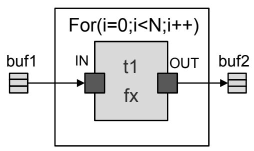
vodafone chair

- Task synchronization: **`synchronize();`**




```
TaskSpec* t1 = task(fx, IN(buf1, BUF1_SIZE), OUT(buf2, BUF2_SIZE));
push_task(t1);
synchronize();
TaskSpec* t2 = task(fy, IN(buf2, BUF2_SIZE), OUT(buf3, BUF3_SIZE));
push_task(t2);
```

- For/While loops: **`for(init; condition; update){ task(…); … }`**




```
for(i=0;i<N;i++){
   TaskSpec* t1 = task((fx, IN(&buf1[i], SIZE1),
   OUT(&buf2[i], SIZE2));
   push_task(1);
}
Synchronize();
```

TU Dresden E.Matus - MPSoC 2015 Slide 10

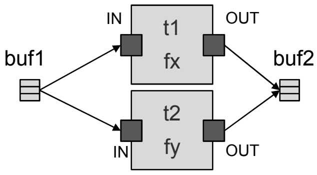
… Computation API

TECHNISCHE UNIVERSITÄT DRESDEN

vodafone chair

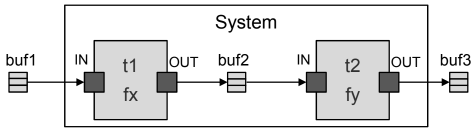
- Create Task:

```
if (condition){
        task(TASK1, …);
else{
        task(TASK2, …);
}
```




```
if(cond){
        TaskSpec* t1 = task(fx, IN((buf1,  BUF1_SIZE), OUT((buf2, BUF2_SIZE));
        push_task(t1);
}
else{
        TaskSpec* t2 = task(fy, IN(buf2, BUF2_SIZE), OUT(buf3, BUF3_SIZE));
        push_task(t2);
}
```

- Hierarchy:



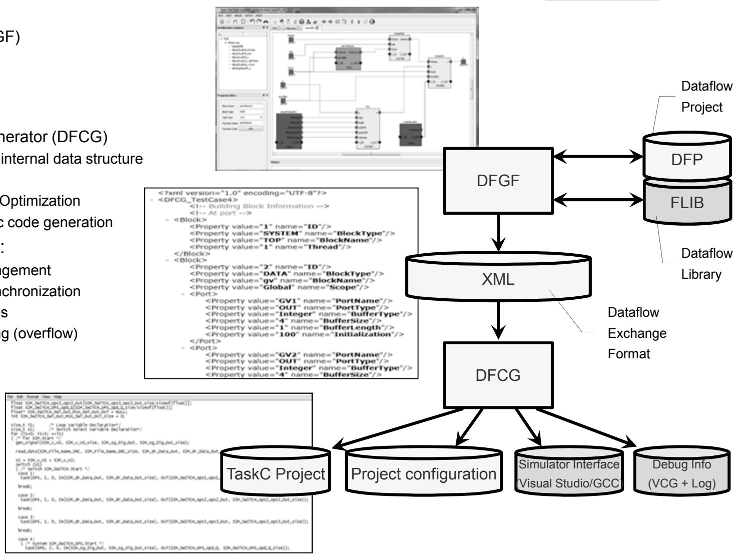
TU Dresden E.Matus - MPSoC 2015 Slide 11

Modeling and Code Generation Toolflow

TECHNISCHE UNIVERSITÄT DRESDEN

vodafone chair

- Model definition:
  - Dataflow GUI (DFGF)
  - XML description
  - C++ API
- Code generation:
  - Dataflow Code Generator (DFCG)
    - Generation of internal data structure
    - Flattening
    - Scheduling & Optimization
    - Target specific code generation
  - Generated code:
    - Memory management
    - Task/Data synchronization
    - Task interfaces
    - Buffer checking (overflow)
    - Debug info
- Debugging:
  - Visual Studio/ GCC project
  - Autoverification vs. MATLAB reference



TU Dresden E.Matus - MPSoC 2015 Slide 12