

FOSDEM'16

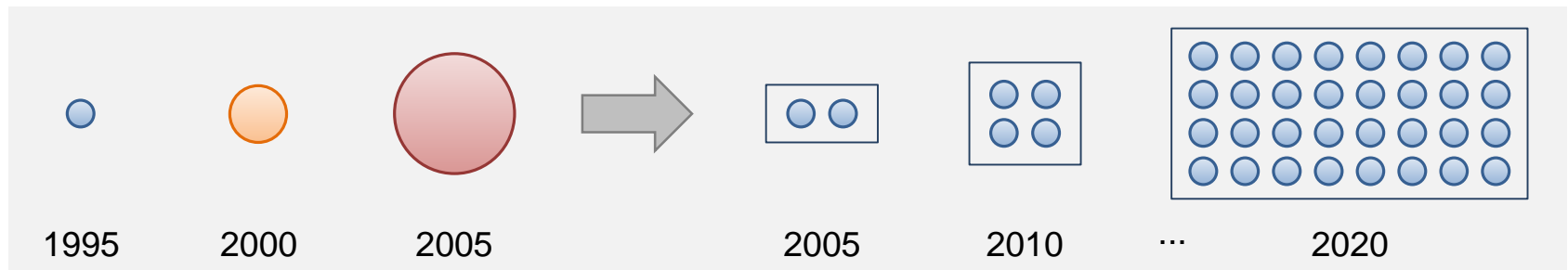
Embedded Multicore Building Blocks (EMB²)

Easy and Efficient Parallel Programming of Embedded Systems

Tobias Schüle
Siemens Corporate Technology

Introduction

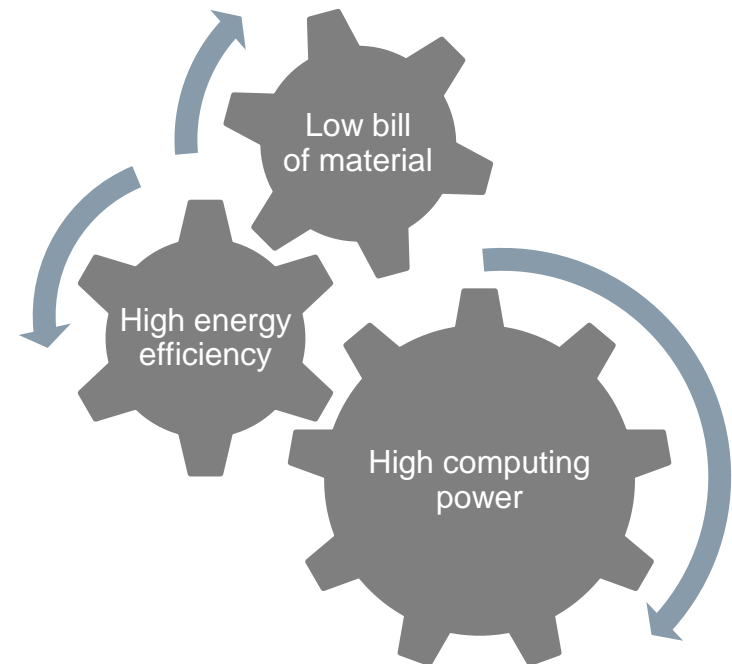
“The free lunch is over!”



- **No further increase of clock frequencies**
due to excessive heat dissipation
- **No significant performance improvements**
of single processor cores

⇒ **Multicore processors are here to stay**

⇒ **Applications need to leverage parallelism**



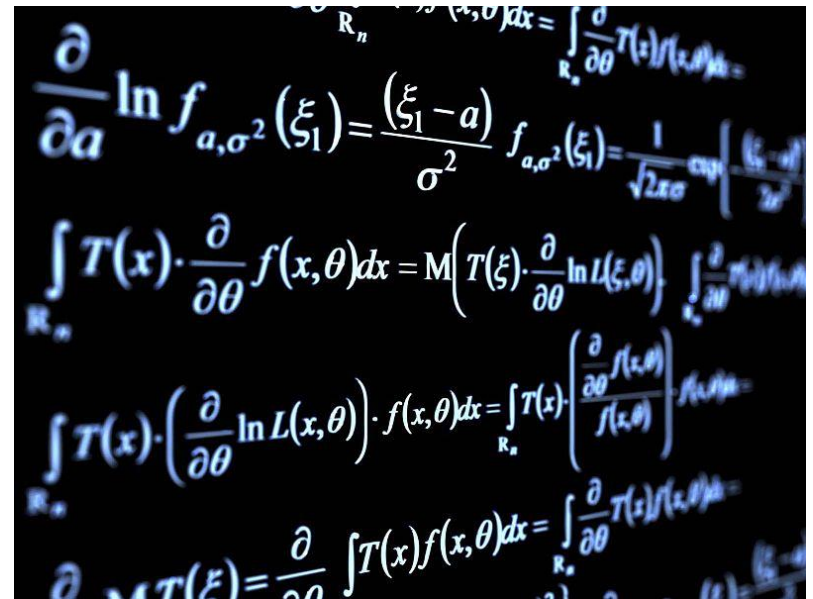
Introduction

Sequential programming is easy (sometimes) ...

Dot product (sequential)

```
#define SIZE 1000
```

```
main() {  
    double a[SIZE], b[SIZE];  
    // Compute a and b ...  
    double sum = 0.0;  
    for(int i = 0; i < SIZE; i++)  
        sum += a[i] * b[i];  
    // Use sum ...  
}
```



The image displays several mathematical formulas related to probability theory and statistics, specifically focusing on the Fisher information and the score function. The formulas are written in a stylized, handwritten-like font on a dark background.

- Top formula: $\frac{\partial}{\partial a} \ln f_{a, \sigma^2}(\xi_1) = \frac{(\xi_1 - a)}{\sigma^2} f_{a, \sigma^2}(\xi_1) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left\{-\frac{(\xi_1 - a)^2}{2\sigma^2}\right\}$
- Second formula: $\int_{\mathbb{R}_n} T(x) \cdot \frac{\partial}{\partial \theta} f(x, \theta) dx = M\left(T(\xi) \cdot \frac{\partial}{\partial \theta} \ln L(\xi, \theta)\right)$
- Third formula: $\int_{\mathbb{R}_n} T(x) \cdot \left(\frac{\partial}{\partial \theta} \ln L(x, \theta)\right) \cdot f(x, \theta) dx = \int_{\mathbb{R}_n} T(x) \cdot \left(\frac{\frac{\partial}{\partial \theta} f(x, \theta)}{f(x, \theta)}\right) \cdot f(x, \theta) dx$
- Bottom formula: $\frac{\partial}{\partial \theta} \ln L(\xi) = \frac{\partial}{\partial \theta} \int_{\mathbb{R}_n} T(x) f(x, \theta) dx = \int_{\mathbb{R}_n} \frac{\partial}{\partial \theta} T(x) f(x, \theta) dx$

<http://wallpaper.com/wallpaper/formula-mathematics-255330>

Public Domain

Introduction

... but multithreaded programming is tedious!

Dot product (POSIX threads)

```
#include <iostream>
● #include <pthread.h>

● #define THREADS 4
  #define SIZE 1000

using namespace std;

double a[SIZE], b[SIZE], sum;

● pthread_mutex_t mutex_sum;

void *dotprod(void *arg) {
●   int my_id = (int)arg;
●   int my_first = my_id * SIZE/THREADS;
●   int my_last = (my_id + 1) * SIZE/THREADS;

   double partial_sum = 0;
   for(int i = my_first; i < my_last && i < SIZE; i++)
       partial_sum += a[i] * b[i];

●   pthread_mutex_lock(&mutex_sum);
   sum += partial_sum;
●   pthread_mutex_unlock(&mutex_sum);

●   pthread_exit((void*)0);
}

int main(int argc, char *argv[]) {
    // Compute a and b ...

●   pthread_attr_t attr;
●   pthread_t threads[THREADS];

●   pthread_mutex_init(&mutex_sum, NULL);
●   pthread_attr_init(&attr);
●   pthread_attr_setdetachstate(&attr,
●       PTHREAD_CREATE_JOINABLE);

    sum = 0;
●   for(int i = 0; i < THREADS; i++)
●       pthread_create(&threads[i], &attr, dotprod,
●           (void*)i);

●   pthread_attr_destroy(&attr);

●   int status;
●   for(int i = 0; i < THREADS; i++)
●       pthread_join(threads[i], (void**)&status);

    // Use sum ...

●   pthread_mutex_destroy(&mutex_sum);
●   pthread_exit(NULL);
}
```

Barbara Chapman, Gabriele Jost, Ruud van der Pas. *Using OpenMP: Portable Shared Memory Parallel Programming*. MIT Press, 2007.

Introduction

“In 2022, multicore will be everywhere.” (IEEE Computer Society)

Various libraries and language extensions for parallel programming available:

- OpenMP
- Intel’s Threading Building Blocks
- Apple’s Grand Central Dispatch
- ...

Target desktop/server applications
Not suitable for embedded systems

Top challenges for multicore (IEEE CS 2022 Report)¹

- Low-power scalable **homogeneous** and **heterogeneous architectures**
- Hard **real-time architectures** with local memory and their programming
- ...

¹ H. Alkhatib, P. Faraboschi, E. Frachtenberg, H. Kasahara, D. Lange, P. Laplante, A. Merchant, D. Milojicic, and K. Schwan. *IEEE CS 2022 Report*. IEEE Computer Society, 2014. www.computer.org/cms/Computer.org/ComputingNow/2022Report.pdf

“Multicore has attracted wide attention from the **embedded systems community** [...]. So far, such parallelization has been performed by application programmers, but it is **very difficult, takes a long time, and has a high cost.**”

Introduction

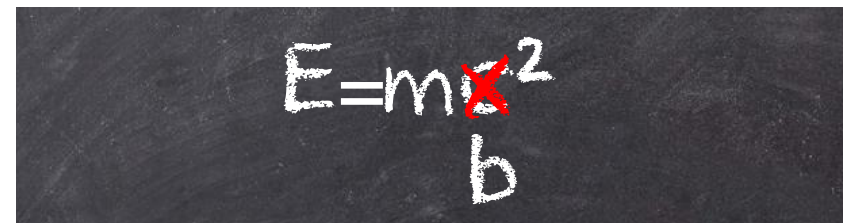
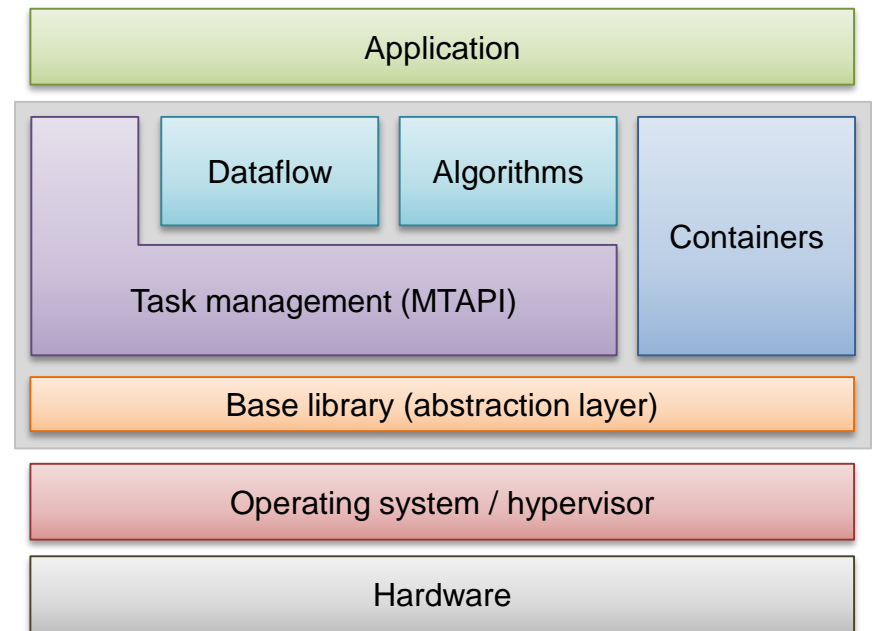
Embedded Multicore Building Blocks (EMB²)

EMB² at a glance

Domain-independent **C/C++ library and runtime platform** for embedded multicore systems.

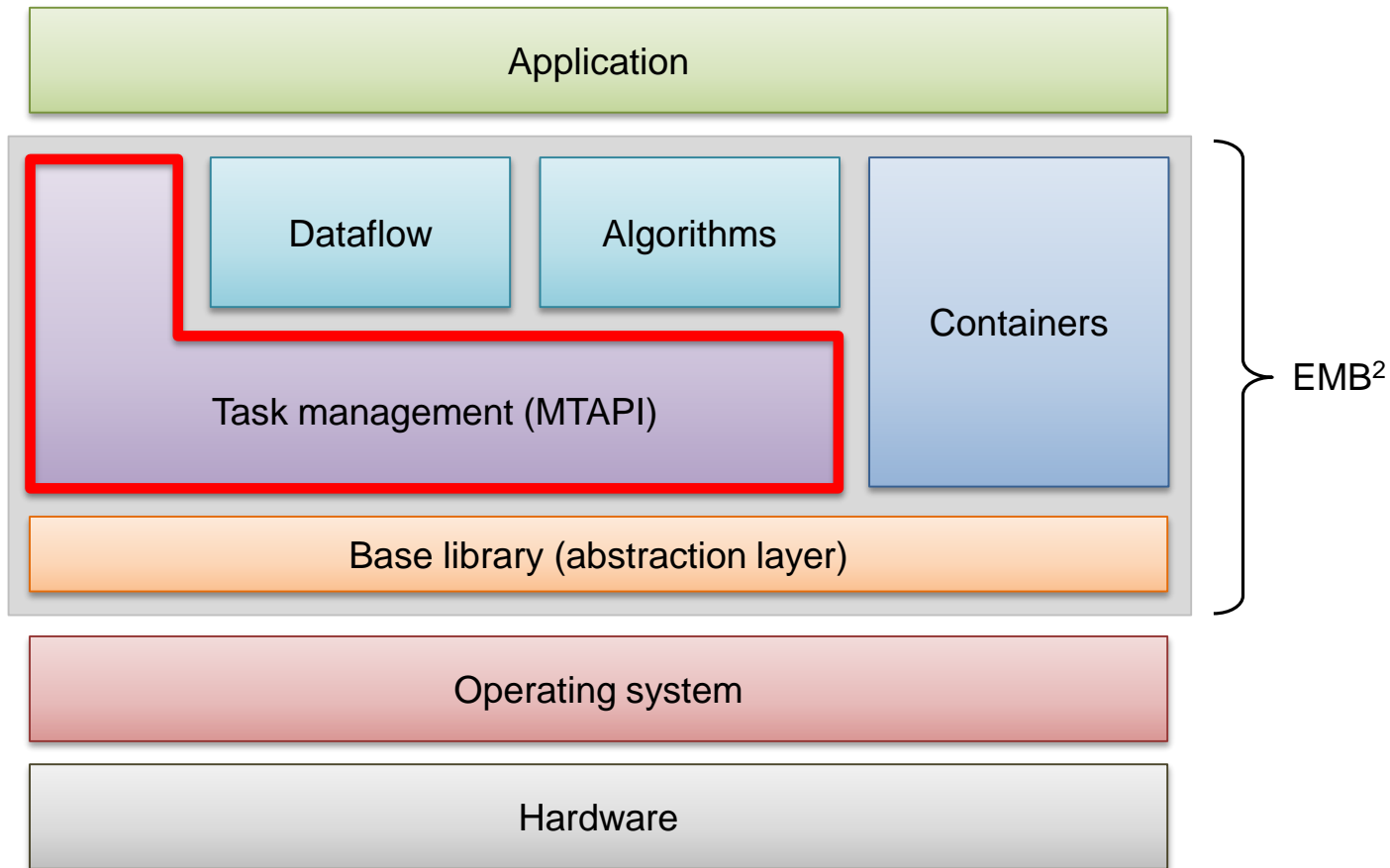
Key features:

- Easy parallelization of existing code
- Real-time capability, resource awareness
- Fine-grained control over core usage (task priorities, affinities)
- Lock-/wait-free implementation
- Support for heterogeneous systems
- Independence of hardware architecture (x86, ARM, ...)



Embedded Multicore Building Blocks

Components

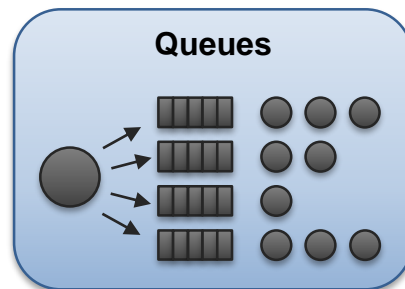
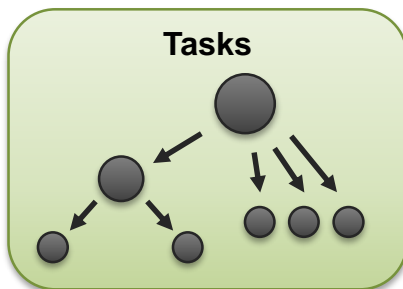


Embedded Multicore Building Blocks

Multicore Task Management API (MTAPI)

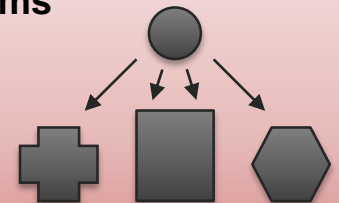
MTAPI in a nut shell

- **Standardized API** for task-parallel programming on a wide range of hardware architectures
- Developed and driven by practitioners of **market-leading companies**
- Part of Multicore Association's **ecosystem** (MRAPI, MCAPI, ...)



Heterogeneous Systems

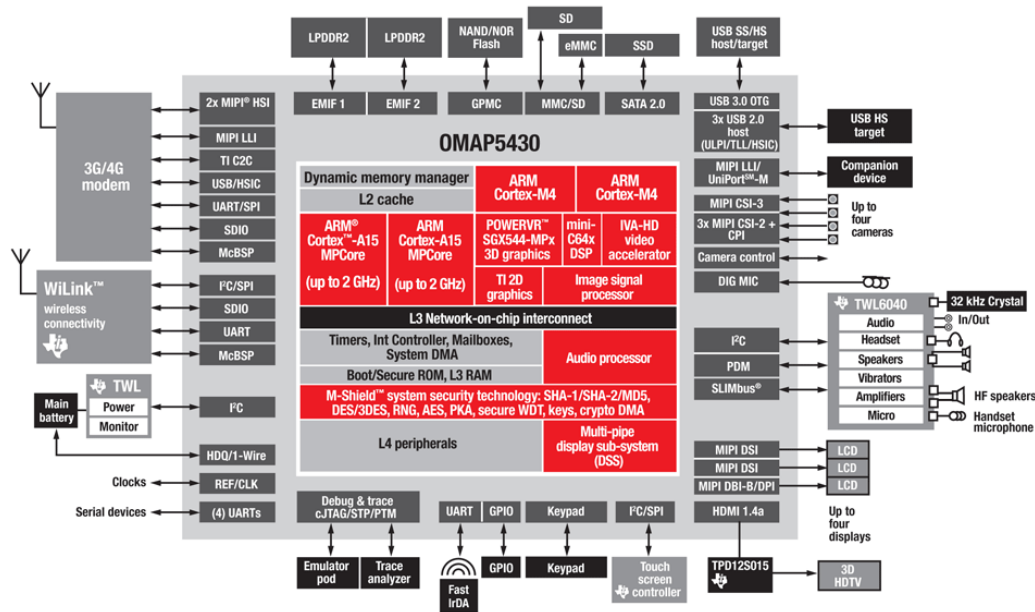
- Shared memory
- Distributed memory
- Different instruction set architectures



Embedded Multicore Building Blocks

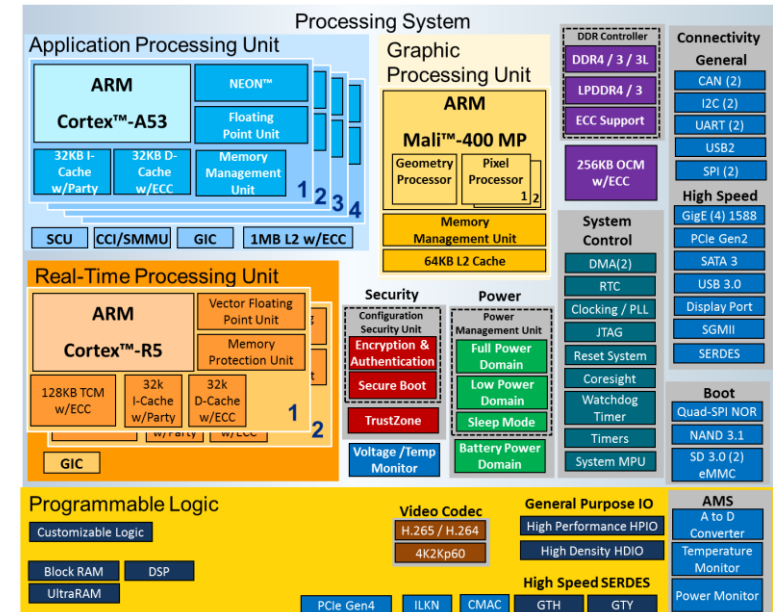
Heterogeneous systems

TI OMAP5430



© Texas Instruments

Xilinx Zynq UltraScale MPSoC



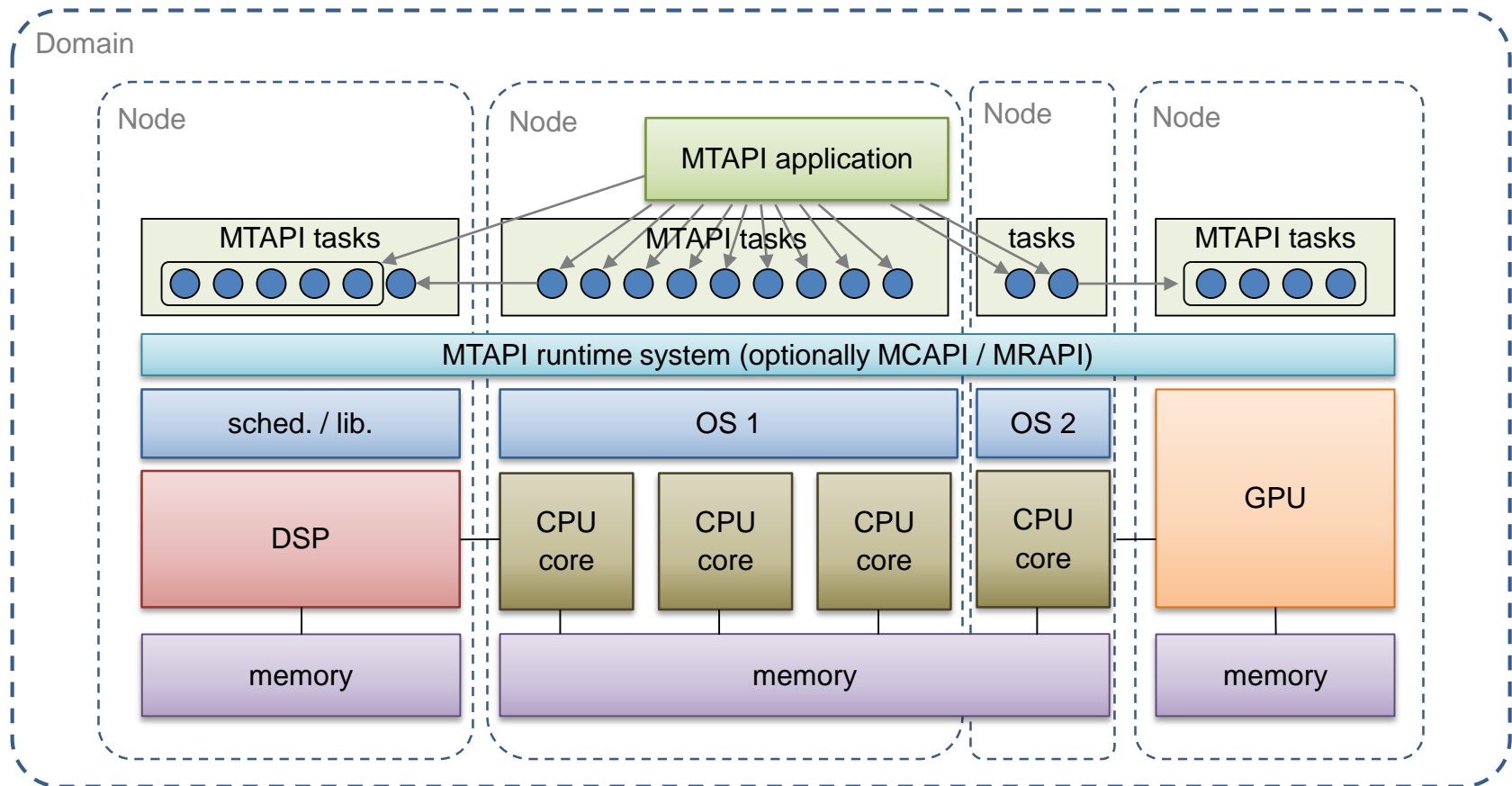
© Xilinx

Key characteristics

- High performance using specialized hardware (DSPs, FPGAs, etc.)
- Low power consumption ⇒ reduced heat dissipation
- Complex programming due to different architectures (lack of abstraction)

Embedded Multicore Building Blocks

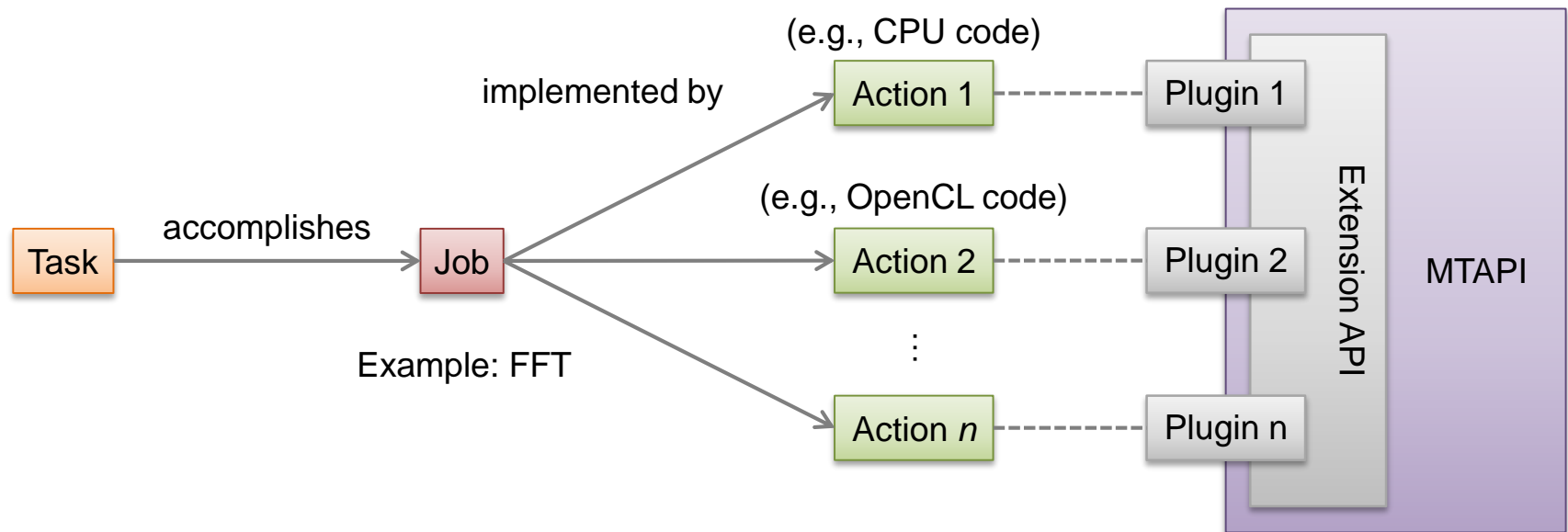
MTAPI for Heterogeneous Systems (1)



Embedded Multicore Building Blocks

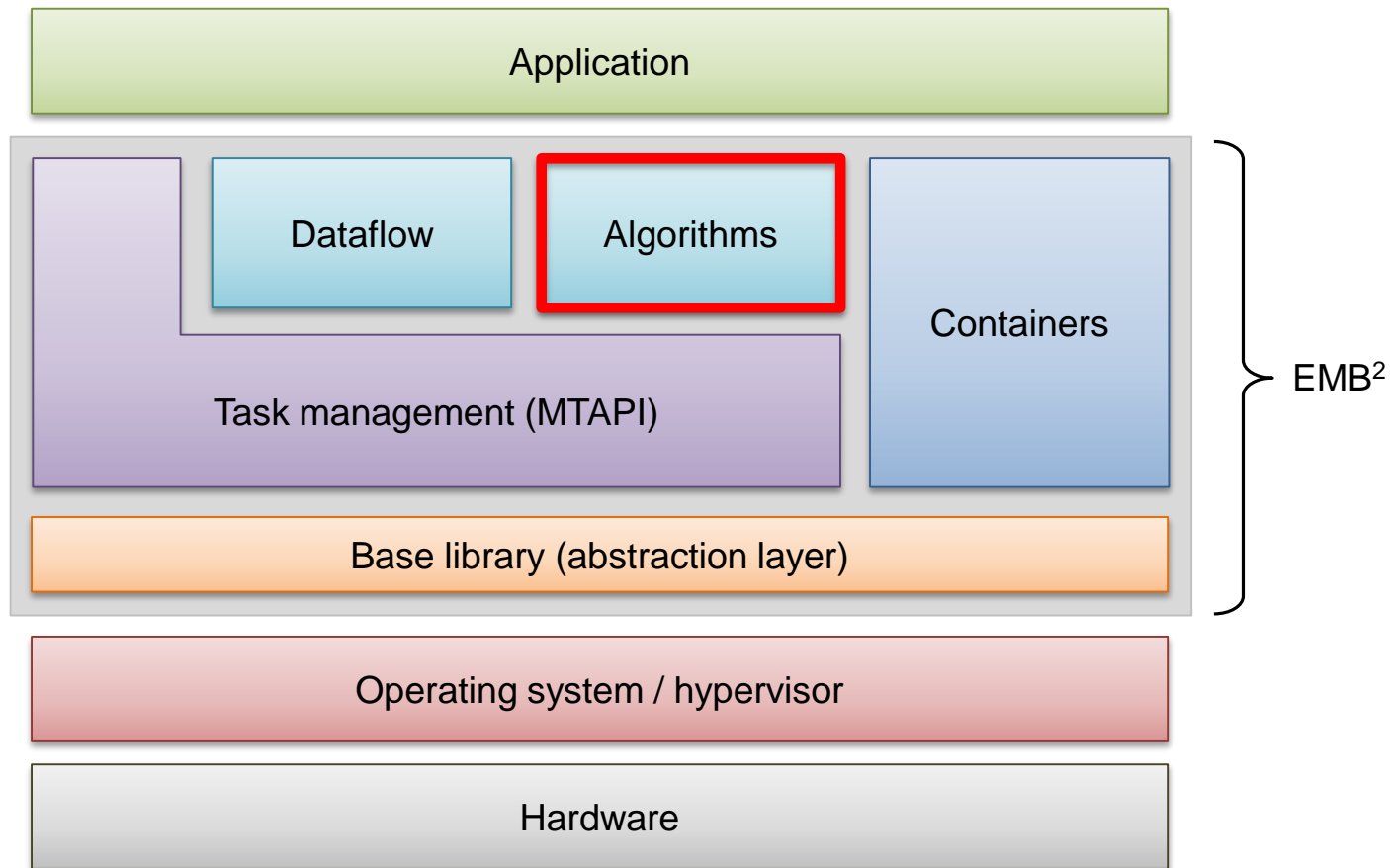
MTAPI for Heterogeneous Systems (2)

- **Job:** A piece of processing implemented by an action. Each job has a unique identifier.
- **Action:** Implementation of a job, may be hardware or software-defined.
- **Task:** Execution of a job resulting in the invocation of an action implementing the job associated with some data to be processed.



Embedded Multicore Building Blocks

Components



Embedded Multicore Building Blocks

Algorithms and Task Affinities / Priorities

Parallel for-each loop

```
std::vector<int> v;  
// initialize v ...  
embb::algorithms::ForEach(v.begin(), v.end(),  
    [] (int& x) {x *= 2;}  
);
```

No need to care of

- task creation and management
- number of processor cores
- load balancing and scheduling
- ...

Function invocation

```
// Create execution policy  
ExecutionPolicy policy(true, 0);  
// Remove worker thread 0 from affinity set  
policy.RemoveWorker(0);  
// Start high priority tasks in parallel on  
// specified worker threads (cores)  
Invoke([=]() {HighPrioFun1();},  
    [=]() {HighPrioFun2();},  
    policy);
```

1st argument: affinity set (true = all)

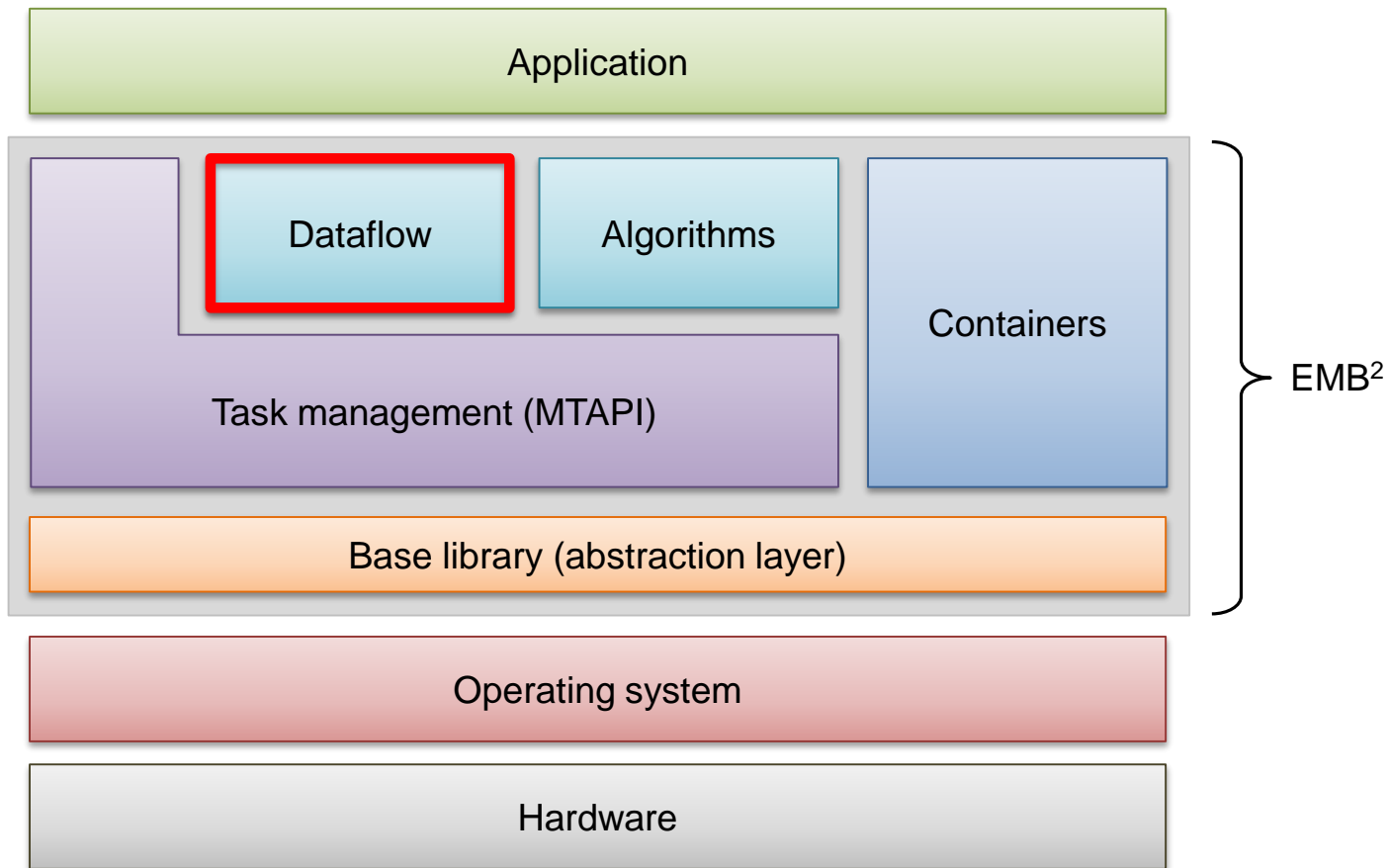
2nd argument: priority (0 = highest)

Example: worker thread (core) 0 is reserved for special tasks

Pass policy as optional parameter

Embedded Multicore Building Blocks

Components

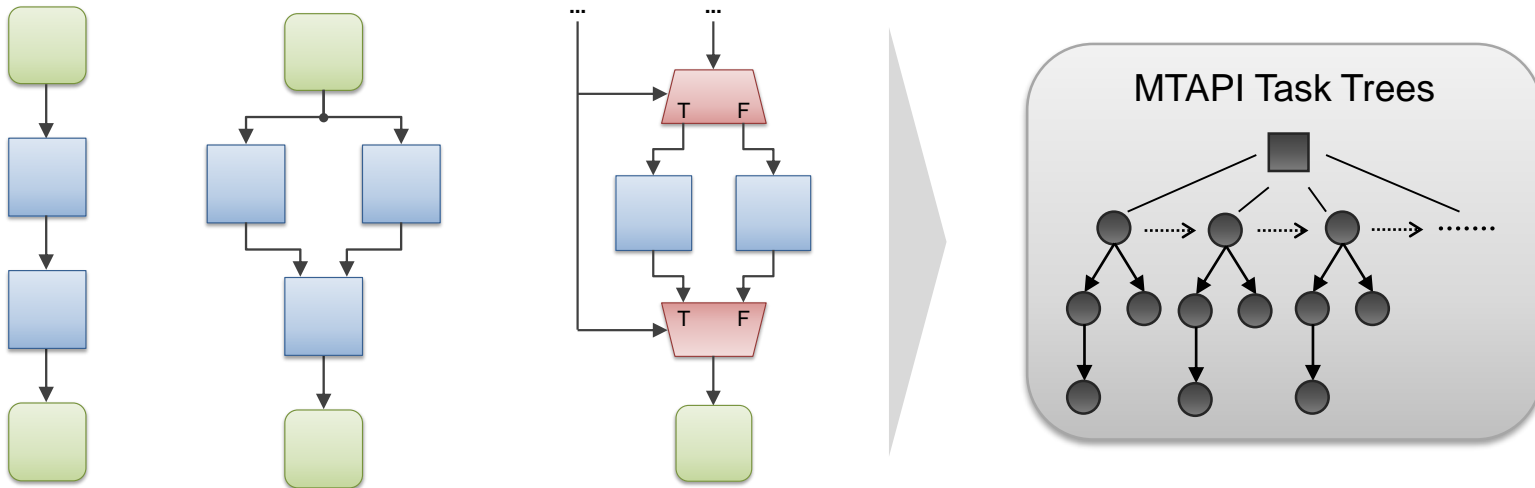


Embedded Multicore Building Blocks

Dataflow Framework

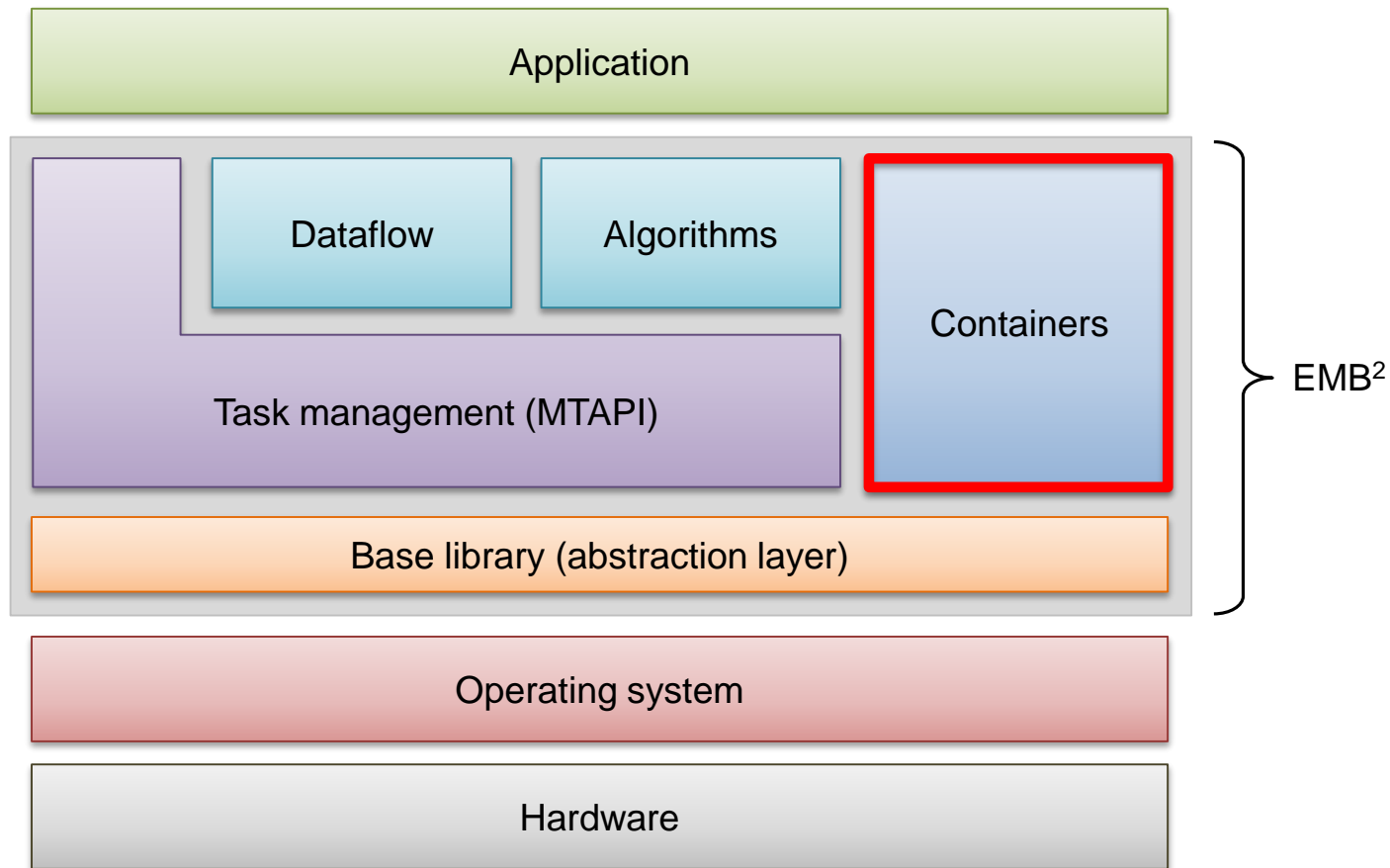
Stream processing

- Embedded systems frequently process **continuous streams of data** such as
 - sensor and actuator data
 - network packets
 - images
 - ...
- Such applications can be modeled using **dataflow networks** and executed in parallel



Embedded Multicore Building Blocks

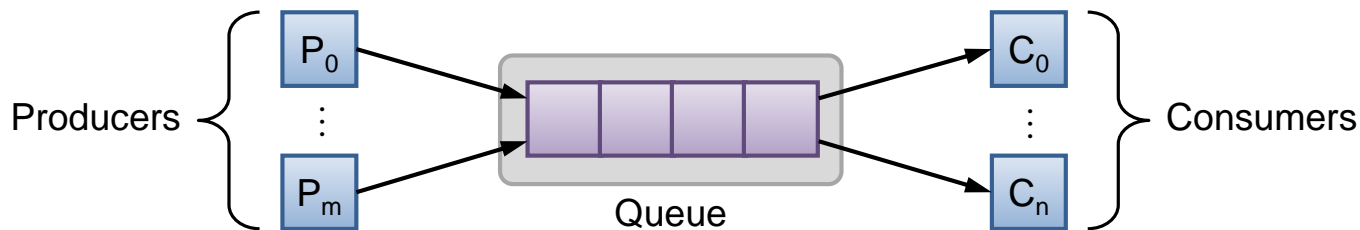
Components



Embedded Multicore Building Blocks

Container Requirements (The Three Commandments)

1. **No race conditions** in case of concurrent accesses \Rightarrow **Thread safety**
2. **No unpredictable delays** in case of contention \Rightarrow **Progress guarantee**
3. **No dynamic memory allocation** after startup \Rightarrow **Preallocated memory**



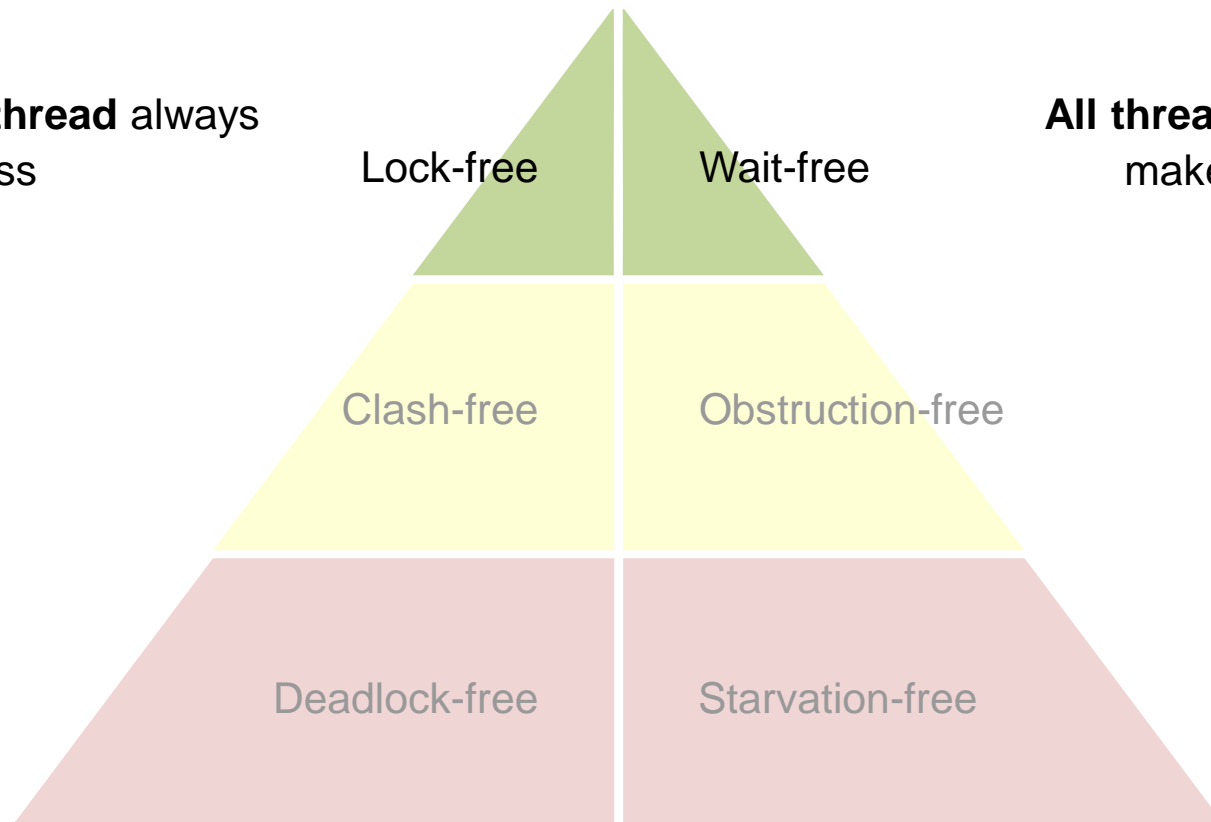
| Implementation | Thread safety | Progress guarantee | Preallocated memory |
|--|---------------|--------------------|---------------------|
| <code>std::queue</code> <code>QQueue (Qt)</code> | ✗ | — | ✗ |
| <code>std::queue</code> <code>QQueue (Qt)</code> + Mutex | ✓ | ✗ | ✗ |
| <code>boost::lockfree::queue</code> <code>tbb::concurrent_queue</code> | ✓ | ✓ / ? | ✗ / ? |
| <code>embb::LockFreeMPMCQueue</code> <code>embb::WaitFreeSPSCQueue</code> | ✓ | ✓ | ✓ |

Embedded Multicore Building Blocks

Progress Guarantees

At least one thread always
makes progress

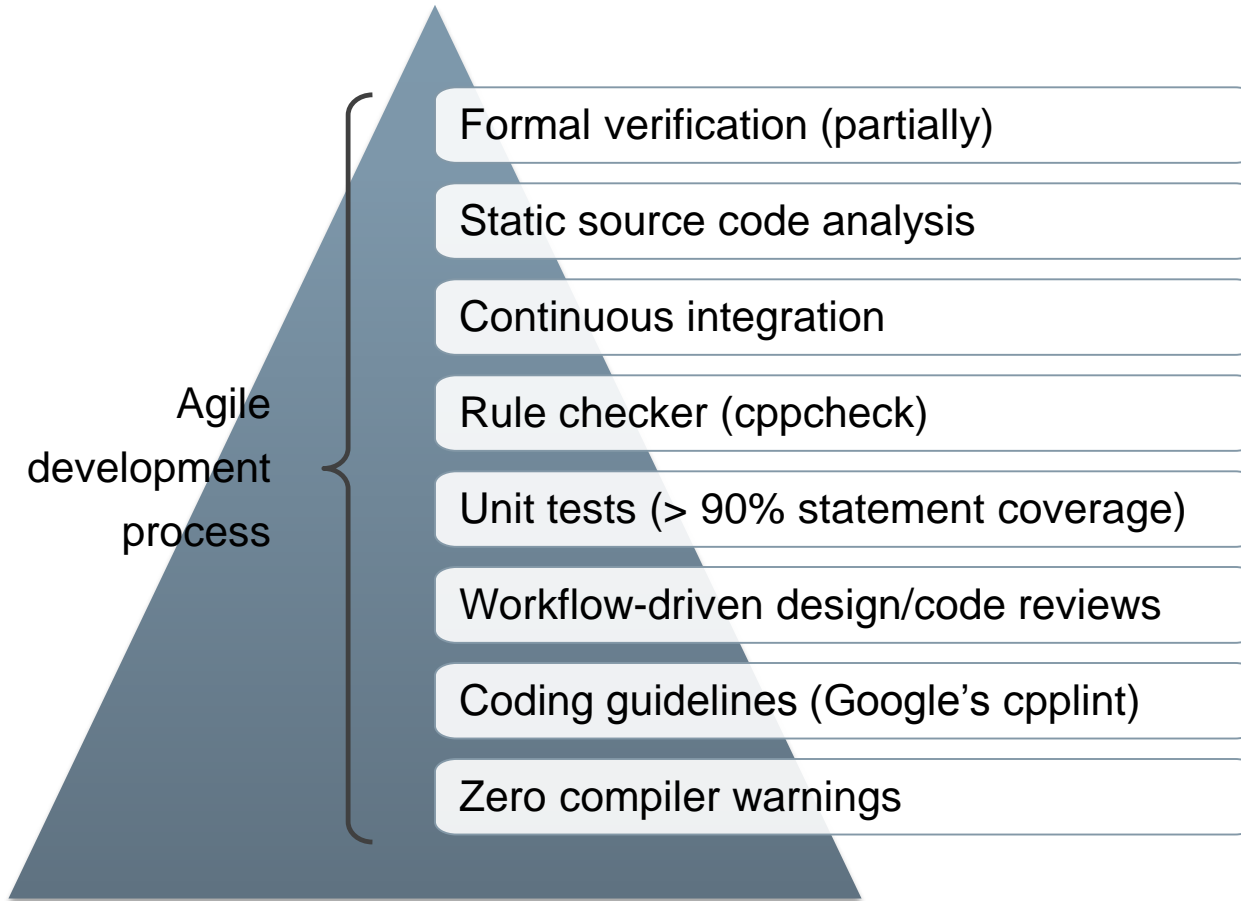
All threads always
make progress



M. Herlihy and N. Shavit. "On the nature of progress". International conference on Principles of Distributed Systems (OPODIS'11), Springer, 2011.

Embedded Multicore Building Blocks

Code Quality



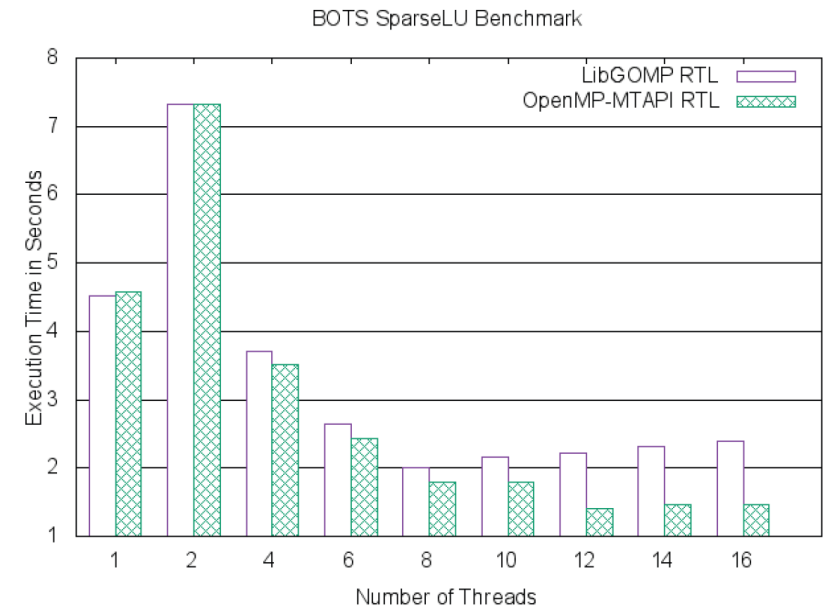
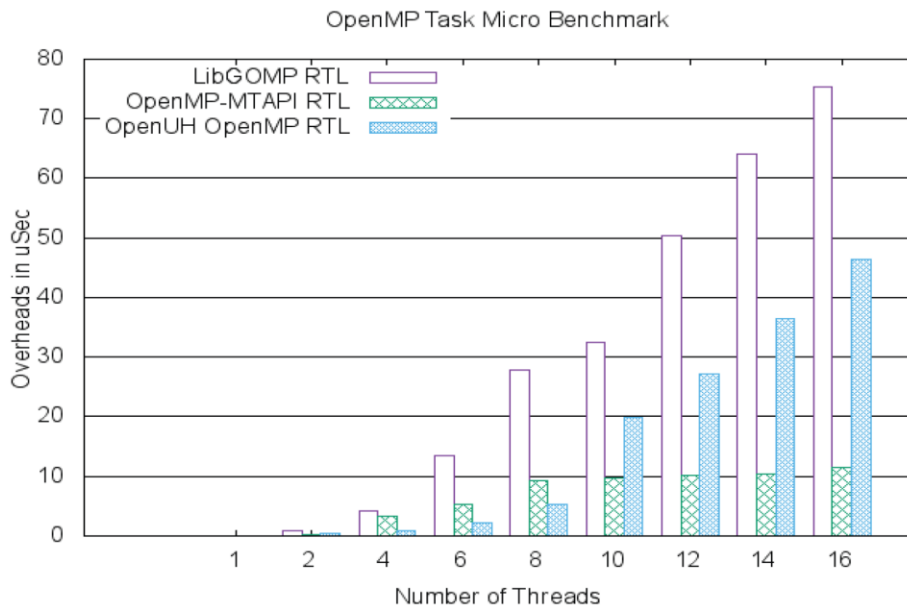
| Category | 2019 | 2020 | 2021 | 2022 | 2023 |
|---------------------------------------|------|------|------|------|------|
| 1. Overall | 100% | 100% | 100% | 100% | 100% |
| 2. Business | 100% | 100% | 100% | 100% | 100% |
| 3. Finance | 100% | 100% | 100% | 100% | 100% |
| 4. Marketing | 100% | 100% | 100% | 100% | 100% |
| 5. Operations | 100% | 100% | 100% | 100% | 100% |
| 6. Human Resources | 100% | 100% | 100% | 100% | 100% |
| 7. Information Technology | 100% | 100% | 100% | 100% | 100% |
| 8. Legal | 100% | 100% | 100% | 100% | 100% |
| 9. Compliance | 100% | 100% | 100% | 100% | 100% |
| 10. Customer Service | 100% | 100% | 100% | 100% | 100% |
| 11. Product Development | 100% | 100% | 100% | 100% | 100% |
| 12. Supply Chain | 100% | 100% | 100% | 100% | 100% |
| 13. Manufacturing | 100% | 100% | 100% | 100% | 100% |
| 14. Quality Control | 100% | 100% | 100% | 100% | 100% |
| 15. Research & Development | 100% | 100% | 100% | 100% | 100% |
| 16. Environmental | 100% | 100% | 100% | 100% | 100% |
| 17. Social | 100% | 100% | 100% | 100% | 100% |
| 18. Governance | 100% | 100% | 100% | 100% | 100% |
| 19. Stakeholder | 100% | 100% | 100% | 100% | 100% |
| 20. Community | 100% | 100% | 100% | 100% | 100% |
| 21. Environment | 100% | 100% | 100% | 100% | 100% |
| 22. Social | 100% | 100% | 100% | 100% | 100% |
| 23. Governance | 100% | 100% | 100% | 100% | 100% |
| 24. Stakeholder | 100% | 100% | 100% | 100% | 100% |
| 25. Community | 100% | 100% | 100% | 100% | 100% |
| 26. Environment | 100% | 100% | 100% | 100% | 100% |
| 27. Social | 100% | 100% | 100% | 100% | 100% |
| 28. Governance | 100% | 100% | 100% | 100% | 100% |
| 29. Stakeholder | 100% | 100% | 100% | 100% | 100% |
| 30. Community | 100% | 100% | 100% | 100% | 100% |
| 31. Environment | 100% | 100% | 100% | 100% | 100% |
| 32. Social | 100% | 100% | 100% | 100% | 100% |
| 33. Governance | 100% | 100% | 100% | 100% | 100% |
| 34. Stakeholder | 100% | 100% | 100% | 100% | 100% |
| 35. Community | 100% | 100% | 100% | 100% | 100% |
| 36. Environment | 100% | 100% | 100% | 100% | 100% |
| 37. Social | 100% | 100% | 100% | 100% | 100% |
| 38. Governance | 100% | 100% | 100% | 100% | 100% |
| 39. Stakeholder | 100% | 100% | 100% | 100% | 100% |
| 40. Community | 100% | 100% | 100% | 100% | 100% |
| 41. Environment | 100% | 100% | 100% | 100% | 100% |
| 42. Social | 100% | 100% | 100% | 100% | 100% |
| 43. Governance | 100% | 100% | 100% | 100% | 100% |
| 44. Stakeholder | 100% | 100% | 100% | 100% | 100% |
| 45. Community | 100% | 100% | 100% | 100% | 100% |
| 46. Environment | 100% | 100% | 100% | 100% | 100% |
| 47. Social | 100% | 100% | 100% | 100% | 100% |
| 48. Governance | 100% | 100% | 100% | 100% | 100% |
| 49. Stakeholder | 100% | 100% | 100% | 100% | 100% |
| 50. Community | 100% | 100% | 100% | 100% | 100% |
| 51. Environment | 100% | 100% | 100% | 100% | 100% |
| 52. Social | 100% | 100% | 100% | 100% | 100% |
| 53. Governance | 100% | 100% | 100% | 100% | 100% |
| 54. Stakeholder | 100% | 100% | 100% | 100% | 100% |
| 55. Community | 100% | 100% | 100% | 100% | 100% |
| 56. Environment | 100% | 100% | 100% | 100% | 100% |
| 57. Social | 100% | 100% | 100% | 100% | 100% |
| 58. Governance | 100% | 100% | 100% | 100% | 100% |
| 59. Stakeholder | 100% | 100% | 100% | 100% | 100% |
| 60. Community | 100% | 100% | 100% | 100% | 100% |
| 61. Environment | 100% | 100% | 100% | 100% | 100% |
| 62. Social | 100% | 100% | 100% | 100% | 100% |
| 63. Governance | 100% | 100% | 100% | 100% | 100% |
| 64. Stakeholder | 100% | 100% | 100% | 100% | 100% |
| 65. Community | 100% | 100% | 100% | 100% | 100% |
| 66. Environment | 100% | 100% | 100% | 100% | 100% |
| 67. Social | 100% | 100% | 100% | 100% | 100% |
| 68. Governance | 100% | 100% | 100% | 100% | 100% |
| 69. Stakeholder | 100% | 100% | 100% | 100% | 100% |
| 70. Community | 100% | 100% | 100% | 100% | 100% |
| 71. Environment | 100% | 100% | 100% | 100% | 100% |
| 72. Social | 100% | 100% | 100% | 100% | 100% |
| 73. Governance | 100% | 100% | 100% | 100% | 100% |
| 74. Stakeholder | 100% | 100% | 100% | 100% | 100% |
| 75. Community | 100% | 100% | 100% | 100% | 100% |
| 76. Environment | 100% | 100% | 100% | 100% | 100% |
| 77. Social | 100% | 100% | 100% | 100% | 100% |
| 78. Governance | 100% | 100% | 100% | 100% | 100% |
| 79. Stakeholder | 100% | 100% | 100% | 100% | 100% |
| 80. Community | 100% | 100% | 100% | 100% | 100% |
| 81. Environment | 100% | 100% | 100% | 100% | 100% |
| 82. Social | 100% | 100% | 100% | 100% | 100% |
| 83. Governance | 100% | 100% | 100% | 100% | 100% |
| 84. Stakeholder | 100% | 100% | 100% | 100% | 100% |
| 85. Community | 100% | 100% | 100% | 100% | 100% |
| 86. Environment | 100% | 100% | 100% | 100% | 100% |
| 87. Social | 100% | 100% | 100% | 100% | 100% |
| 88. Governance | 100% | 100% | 100% | 100% | 100% |
| 89. Stakeholder | 100% | 100% | 100% | 100% | 100% |
| 90. Community | 100% | 100% | 100% | 100% | 100% |
| 91. Environment | 100% | 100% | 100% | 100% | 100% |
| 92. Social | 100% | 100% | 100% | 100% | 100% |
| 93. Governance | 100% | 100% | 100% | 100% | 100% |
| 94. Stakeholder | 100% | 100% | 100% | 100% | 100% |
| 95. Community | 100% | 100% | 100% | 100% | 100% |
| 96. Environment | 100% | 100% | 100% | 100% | 100% |
| 97. Social | 100% | 100% | 100% | 100% | 100% |
| 98. Governance | 100% | 100% | 100% | 100% | 100% |
| 99. Stakeholder | 100% | 100% | 100% | 100% | 100% |
| 100. Community | 100% | 100% | 100% | 100% | 100% |



Embedded Multicore Building Blocks

Performance Comparison

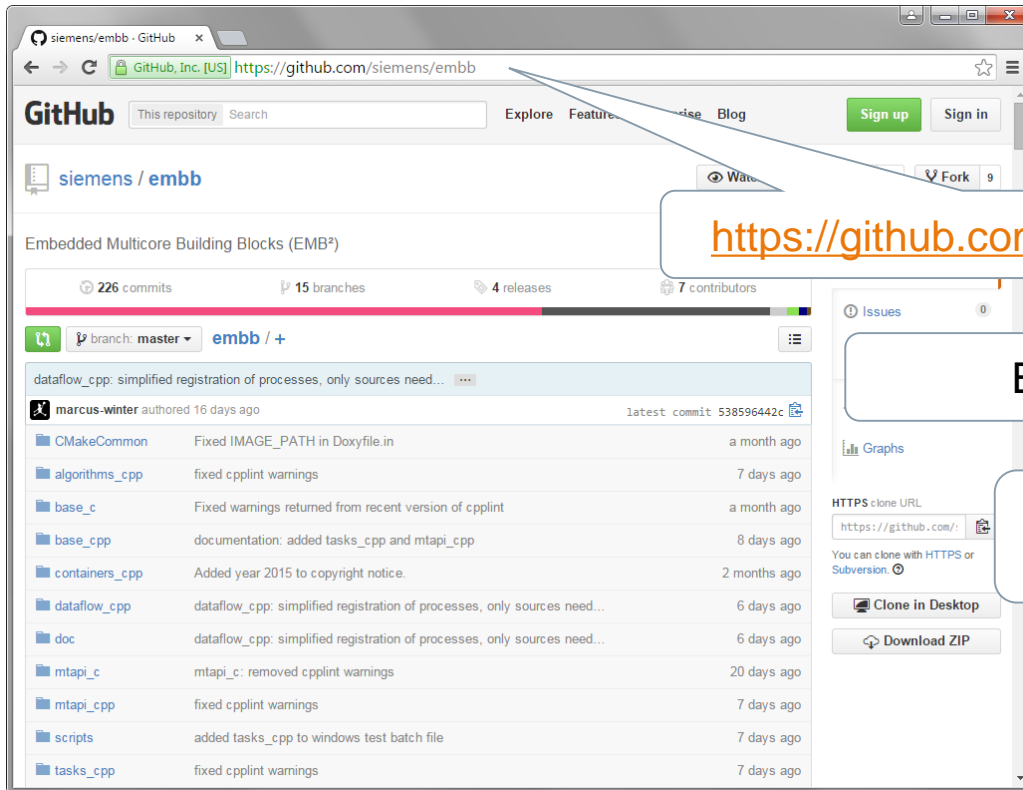
Measurements from the University of Houston show the efficiency of EMB² (green bars):



P. Sun, S. Chandrasekaran, S. Zhu, and B. Chapman. *Deploying OpenMP Task Parallelism on Multicore Embedded Systems with MCA Task APIs*. International Conference on High Performance Computing and Communications (HPCC), IEEE, 2015.

Embedded Multicore Building Blocks

Open Source Software



<https://github.com/siemens/embb/>

BSD 2-clause license

Feedback and contributions
are very welcome!

Thank you!