

Abstract System-on-Chip modeling in SystemC

**Kashif Virk, Shankar Mahadevan,
Jan Madsen**

Informatics and Mathematical Modeling
Technical University of Denmark
Richard Petersens Plads, Building 321
DK2800 Lyngby, Denmark

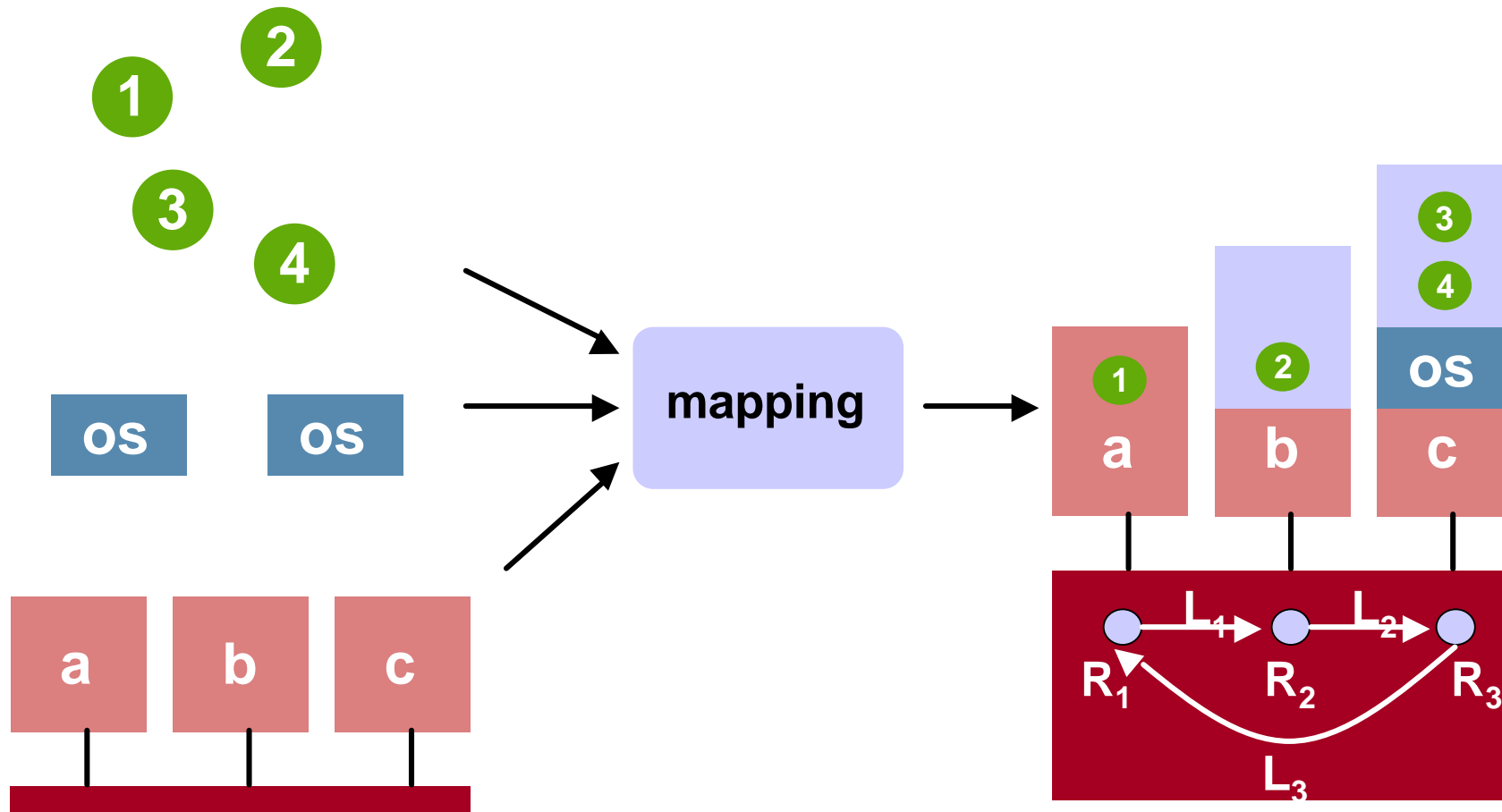


SoC-Mobinet



Funded by SoC-Mobinet (IST 2000-30094)

Motivation



❖ System-level analysis

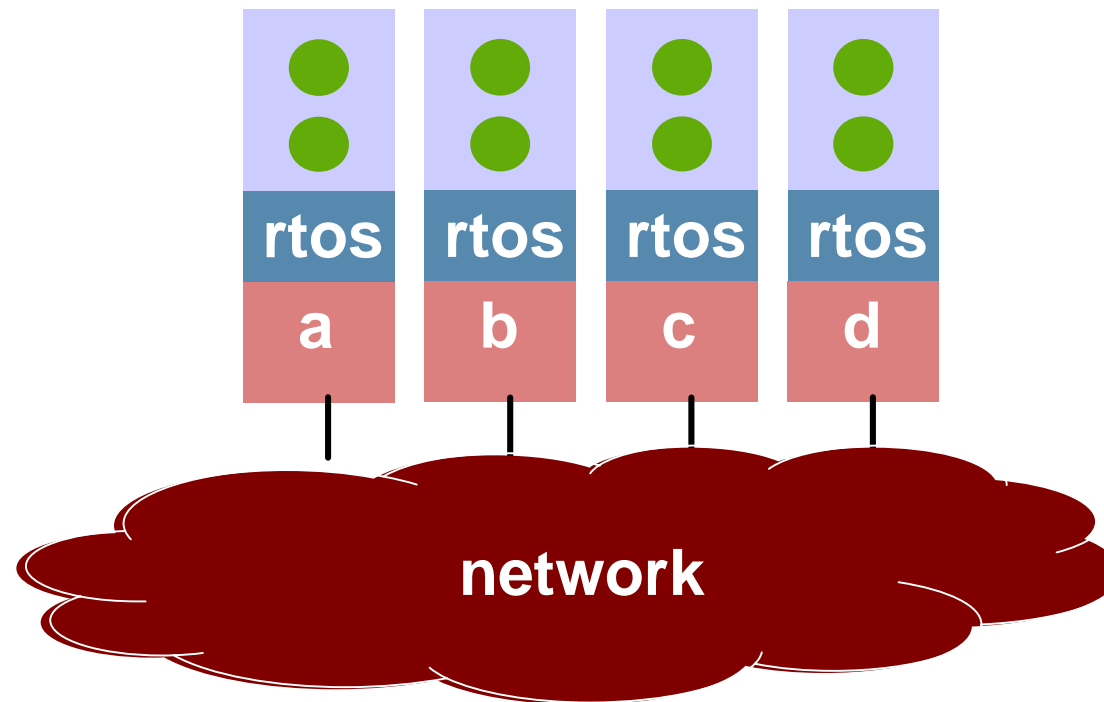
- Consequences of different **mappings of tasks** to processors – software or hardware
- Effects of different **RTOS selections** – scheduling, synchronization and resource allocation policies
- Effects of different **Network** topologies and communication protocols.



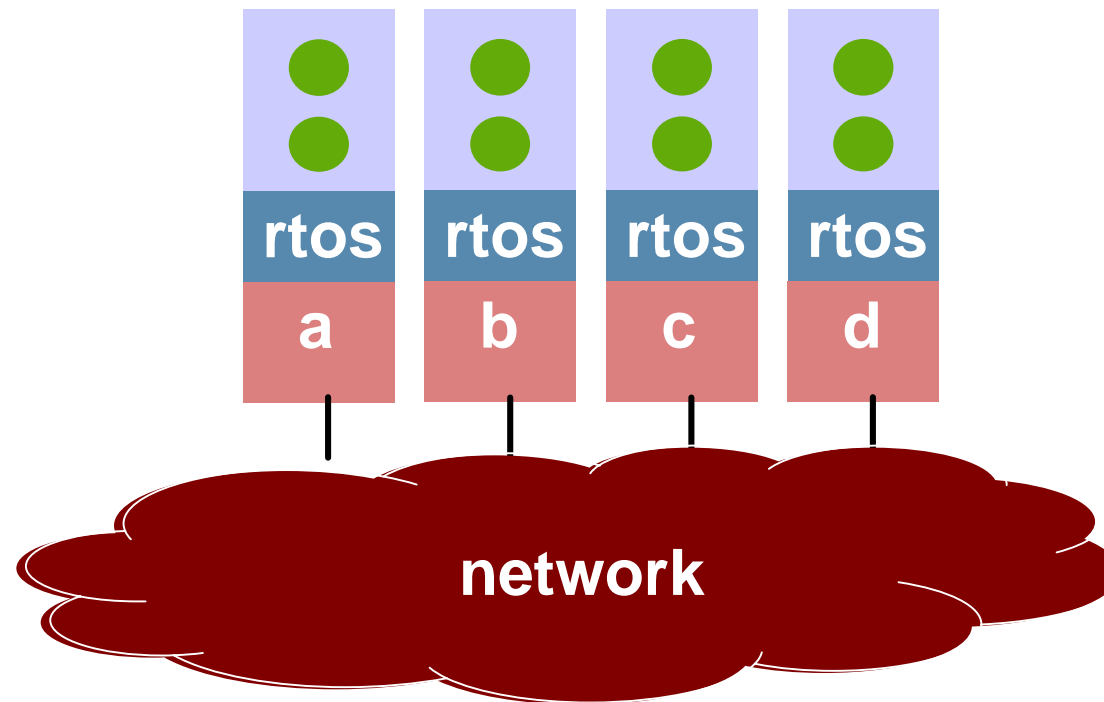
Outline

- Motivation
- Abstract RTOS model
- Extension to handle networks
- Example
- Design space exploration
- Conclusions

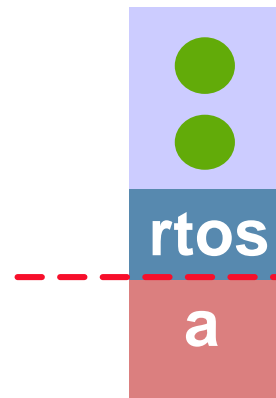
❖ Framework – abstract RTOS modelling



❖ Framework – abstract RTOS modelling



❖ Uni-processor ...



Framework to experiment with different RTOS strategies

Focus on analysis of **timing** and resource sharing

Abstract software model, i.e. no behavior/functionality

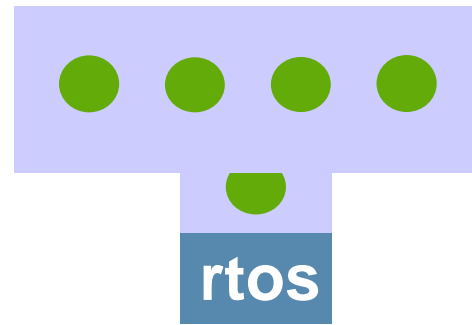
Easy to create tasks and implement RTOS models

Based on **SystemC**

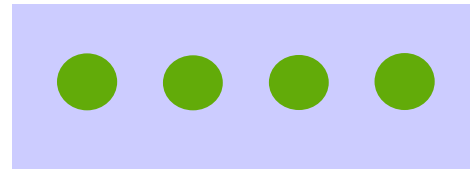
❖ System model



❖ System model

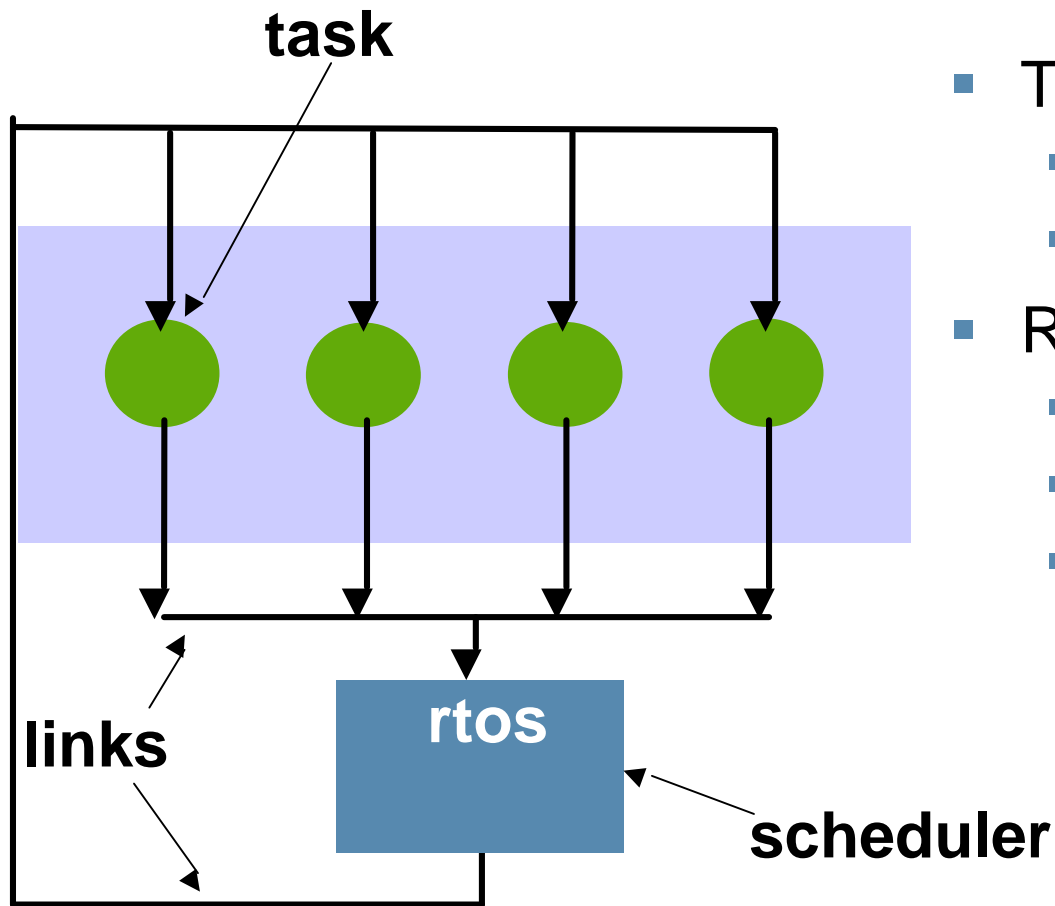


❖ System model



rtos

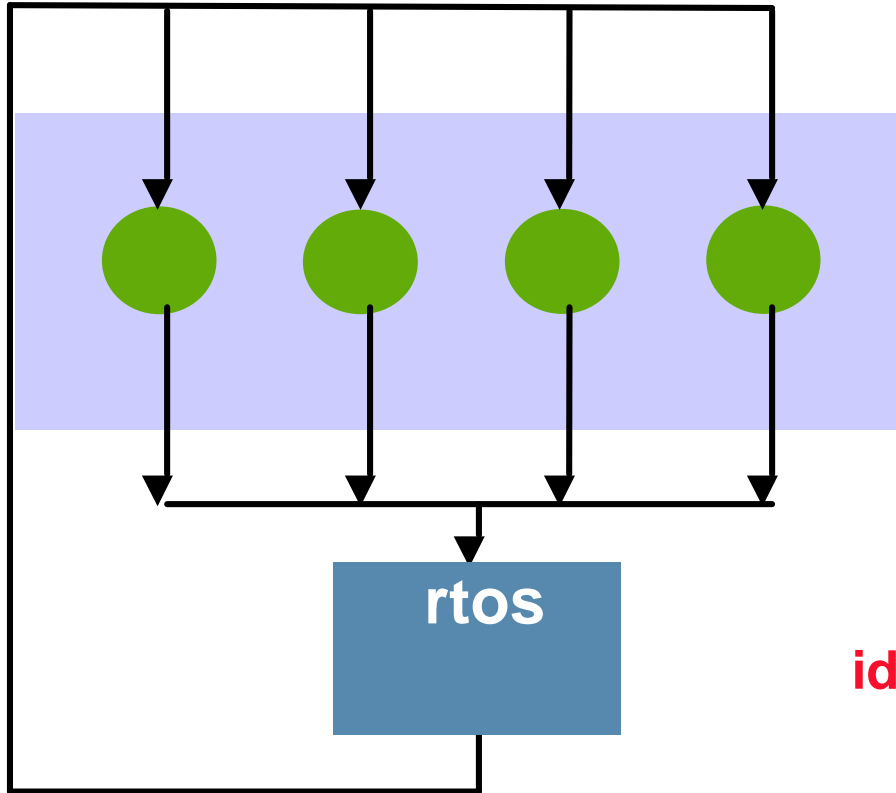
❖ System model



- Task messages:
 - ready
 - finished
- RTOS commands:
 - run
 - preempt
 - Resume



System model - SystemC



```
pa = new  
    task("task_a",1,50,3,12,0,ready);  
registerTask(pa,cpu1);
```

```
pb = new  
    task("task_b",2,40,2,10,0,ready);  
registerTask(pb,cpu1);
```

```
pc = new  
    task("task_c",3,30,1,10,0,ready);  
registerTask(pc,cpu1);
```

identifier

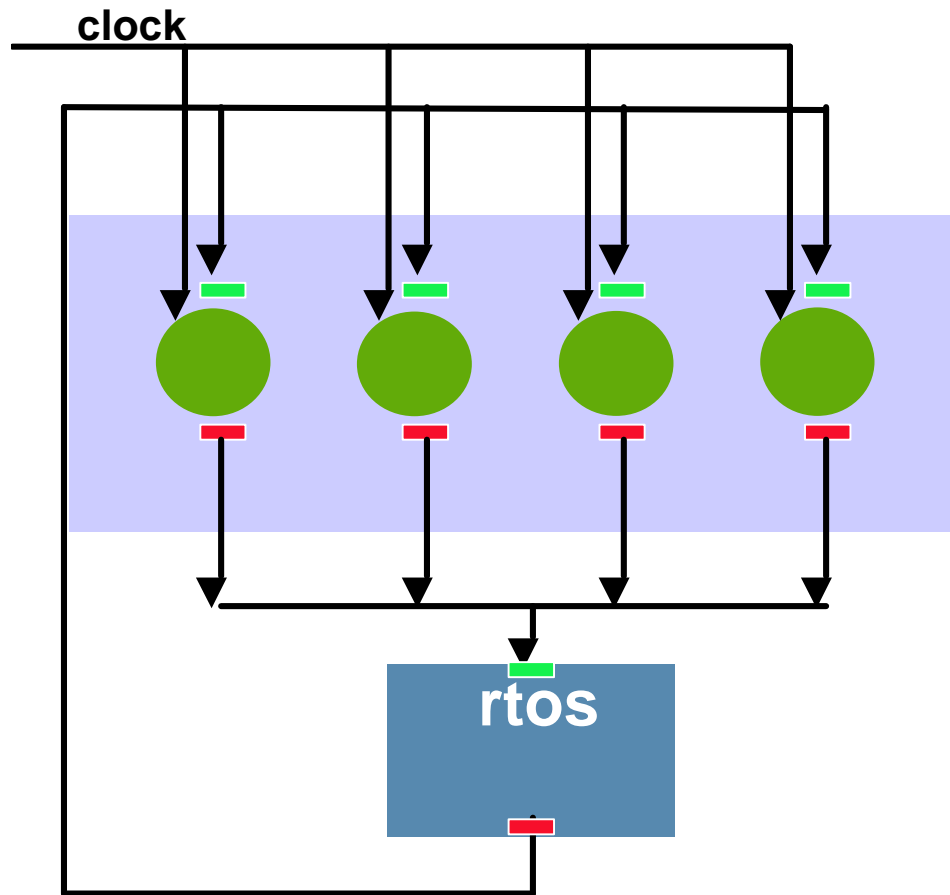
period

priority

WCET

offset

❖ Link model



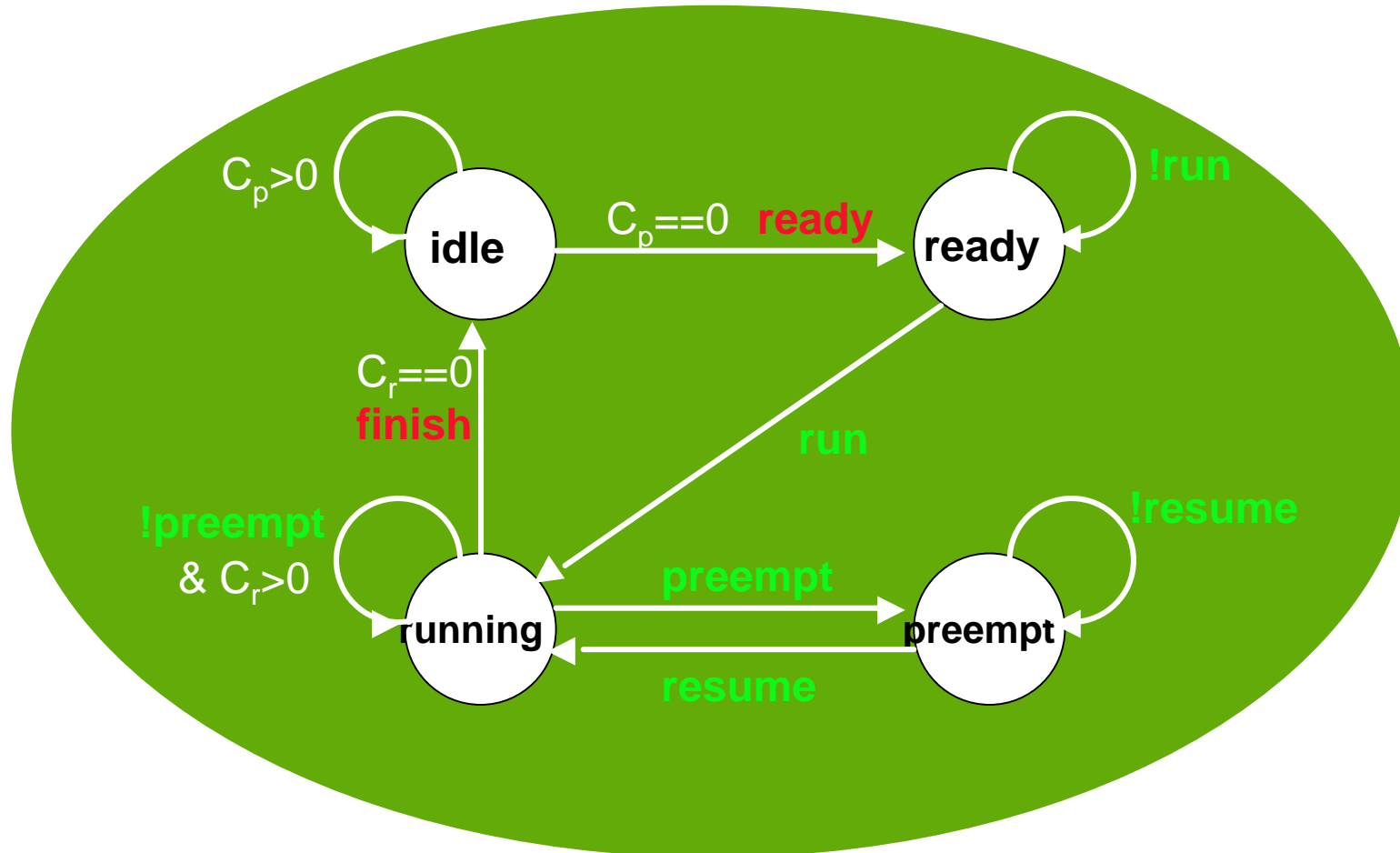
- Aim: Adding tasks without having to create separate communication links
- Uses the SystemC **master-slave** library
- If two tasks send a message at the same time – they are executed in sequence, but in undefined order
- Global "clock" is used to keep track of time

Task model

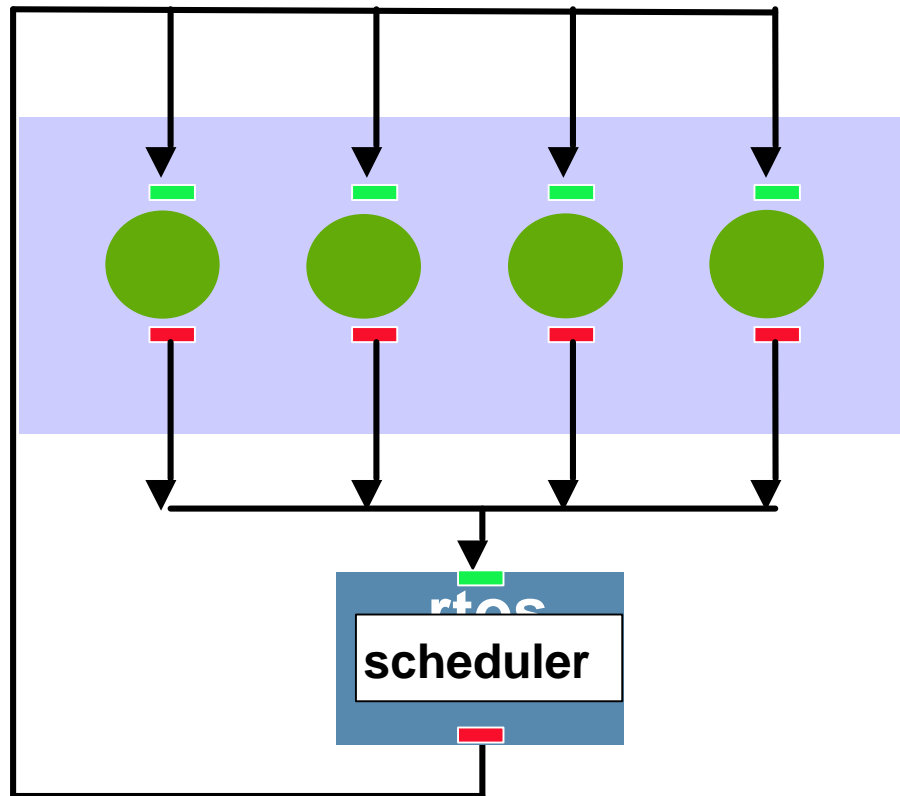
- Periodic task
- Aperiodic task
- Sporadic task



❖ Task model



❖ Scheduling model



- Aim: Simple way to describe the scheduling algorithm
- Scheduler should only handle one message at a time
- Schedulers:
 - RM
 - DM
 - EDF
 - static



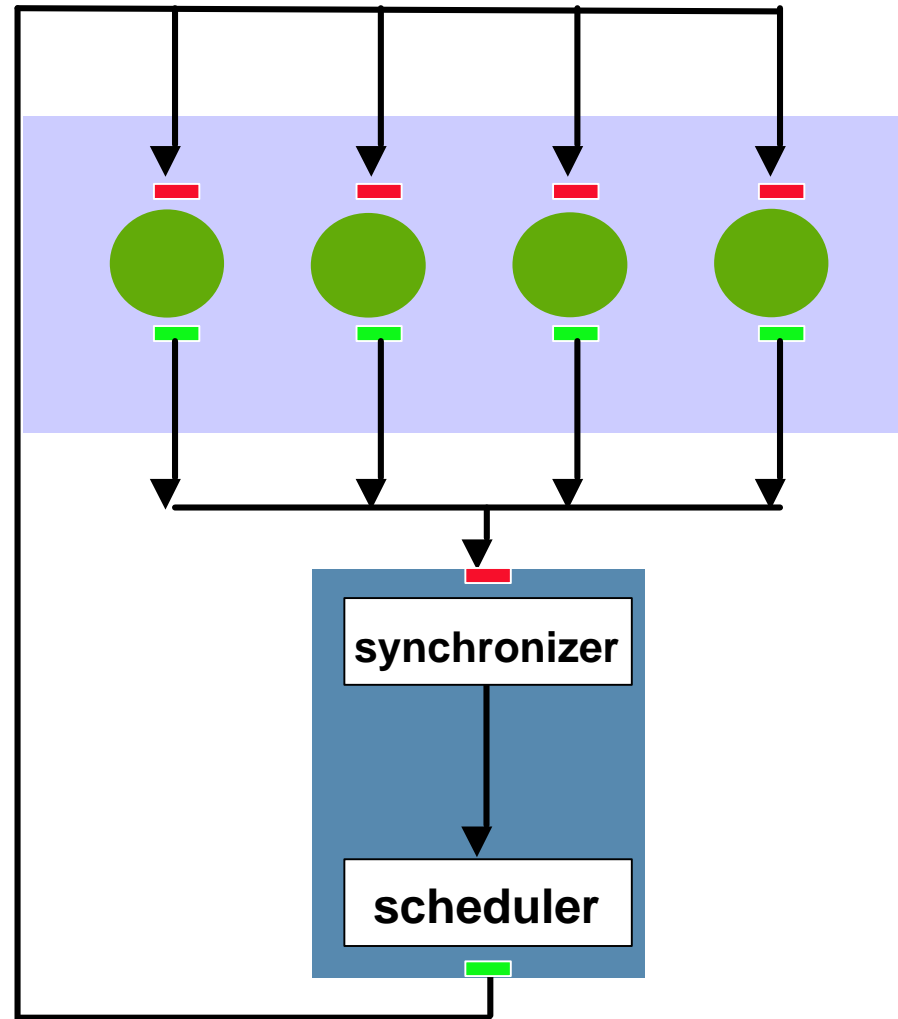
Scheduling model - SystemC

Rate monotonic scheduling

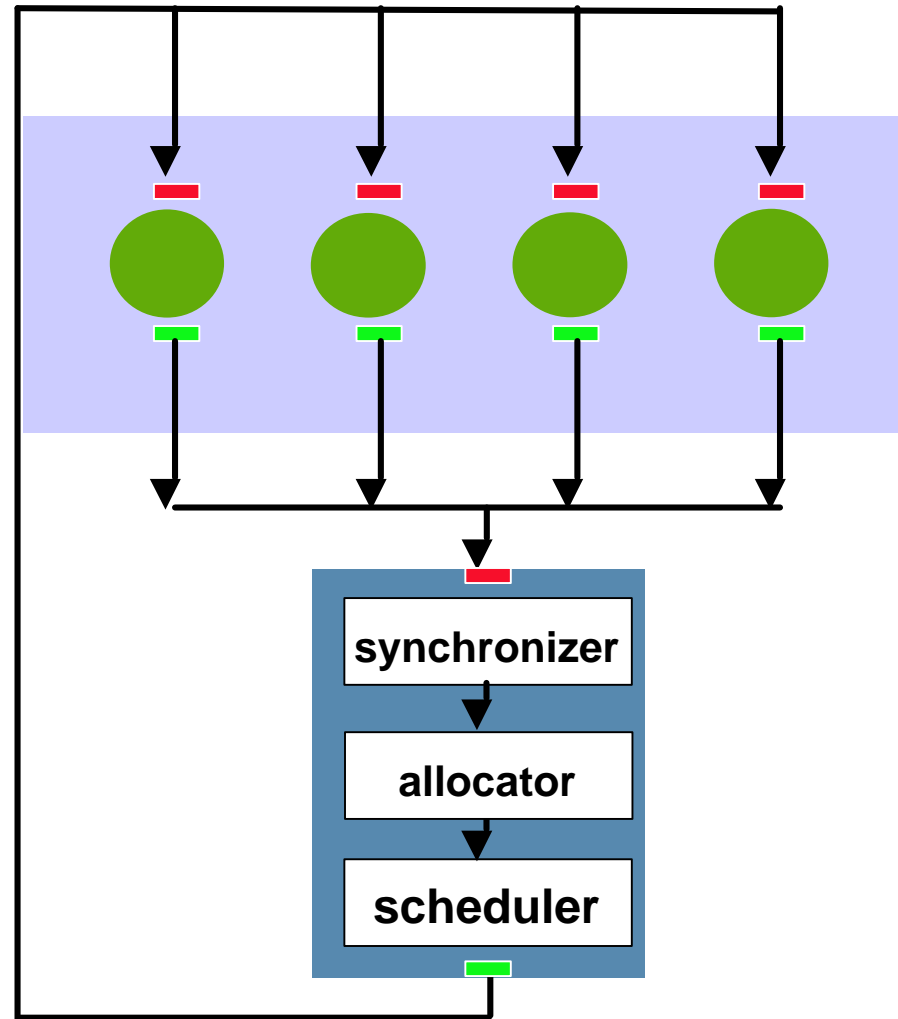
```
void RM_scheduler::doSchedule() {
    ti = in_message;
    if (ti.comm==ready) {
        Q.push(ti);
    } else {
        tk.id = 0;
    }
    tj = Q.top();
    if (tj.id != 0) {
        if (tk.id != 0) {
            if (tk < tj) {
                out_command = *(preemptTask(&tk, &Q));
                tk = tj;
                out_command = *(runTask(&tj, &Q));
            } else {
            }
        } else {
            tk = tj;
            out_command = *(runTask(&tj, &Q));
        }
    } else {
    }
}
```

- t_i : message received from from task i
- t_j : message from task j with highest priority from ready list
- t_k : message of the currently running task

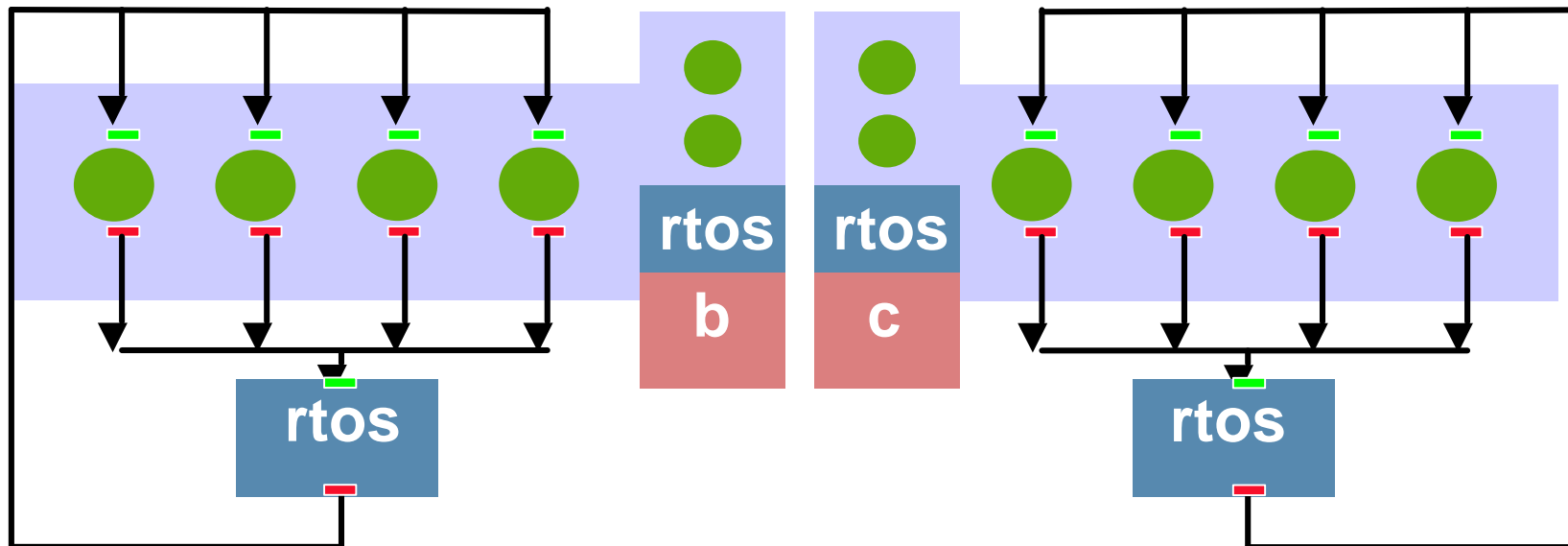
❖ Data dependencies



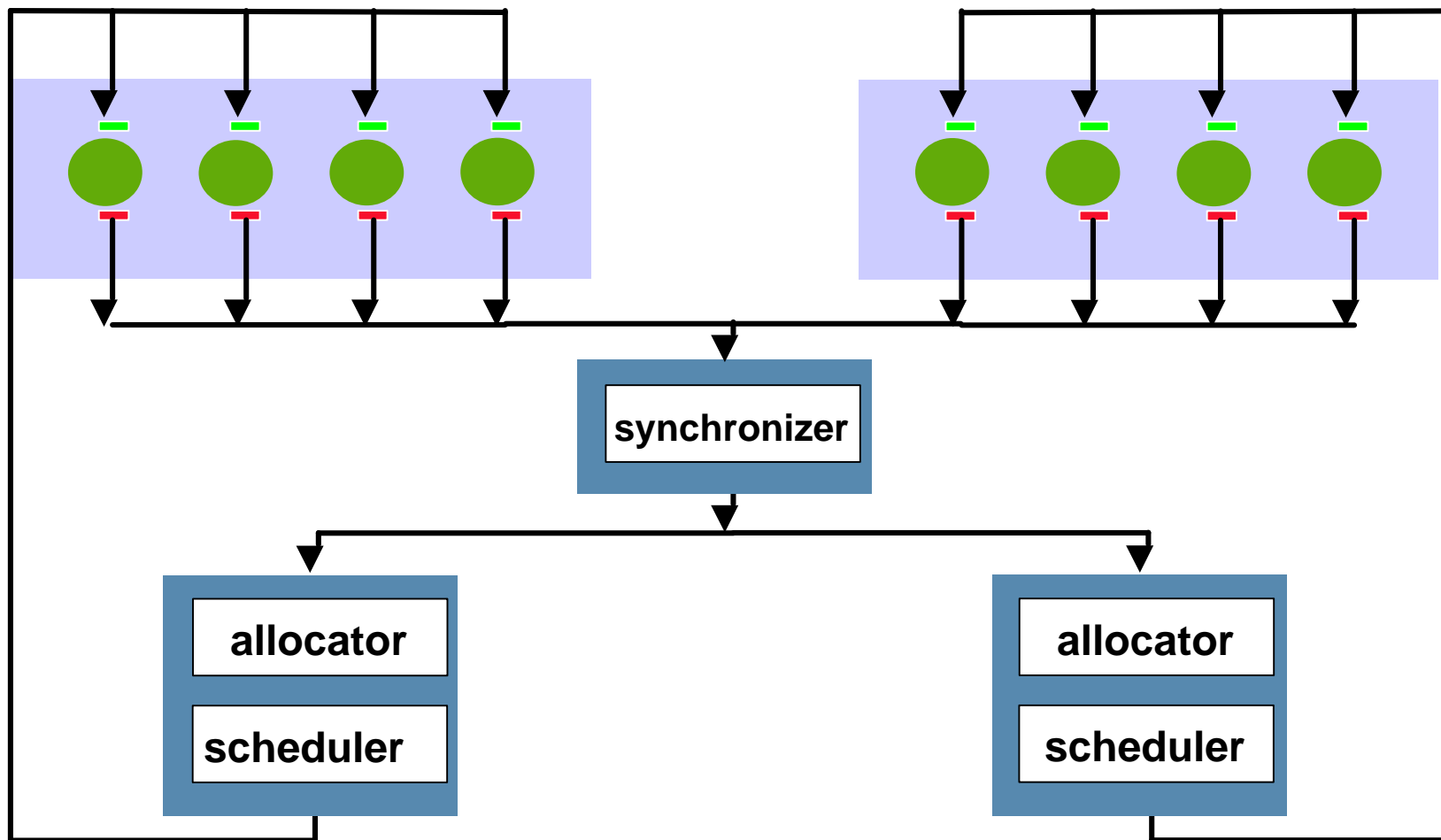
Resource sharing



Multi-processors



Multi-processors

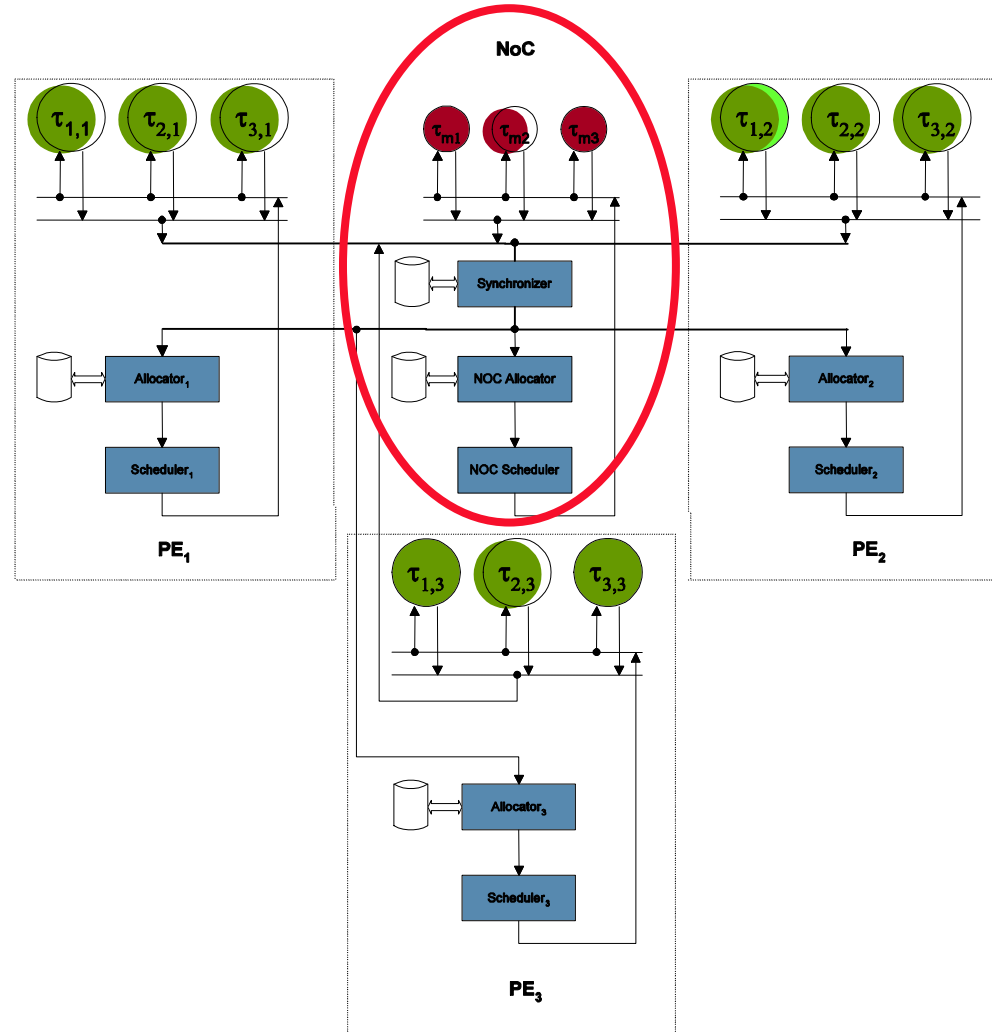


❖ Networked multi-processors

- Data transfers between processors are considered as **message tasks**
- The network can be considered as a **communication processor** on which message tasks are scheduled
- The network provides,
 - Topology = resource allocator
 - Protocol = scheduler



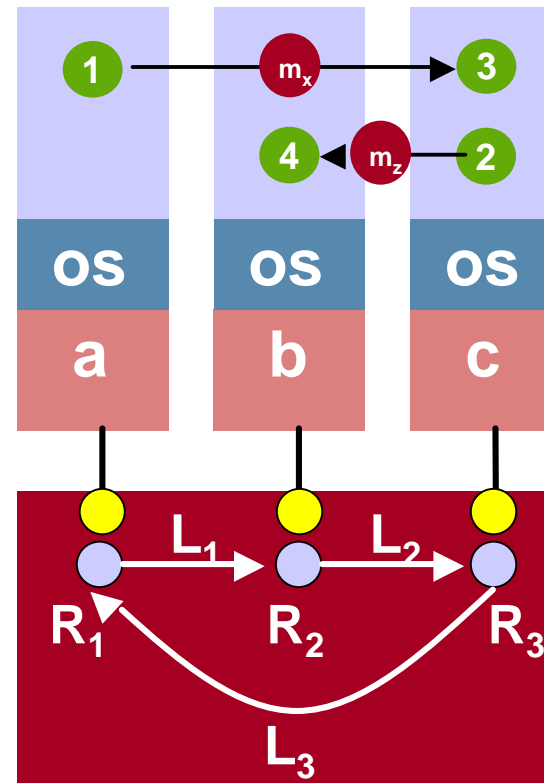
Network-on-Chip model



Simple MPSoC example

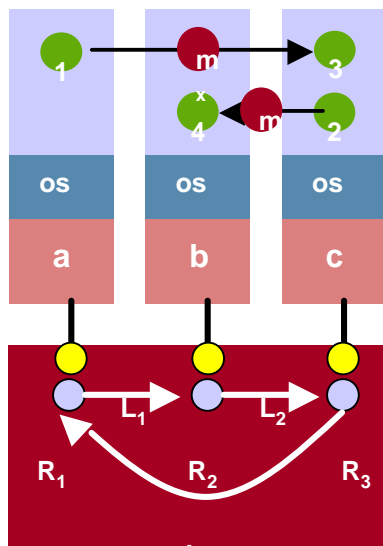
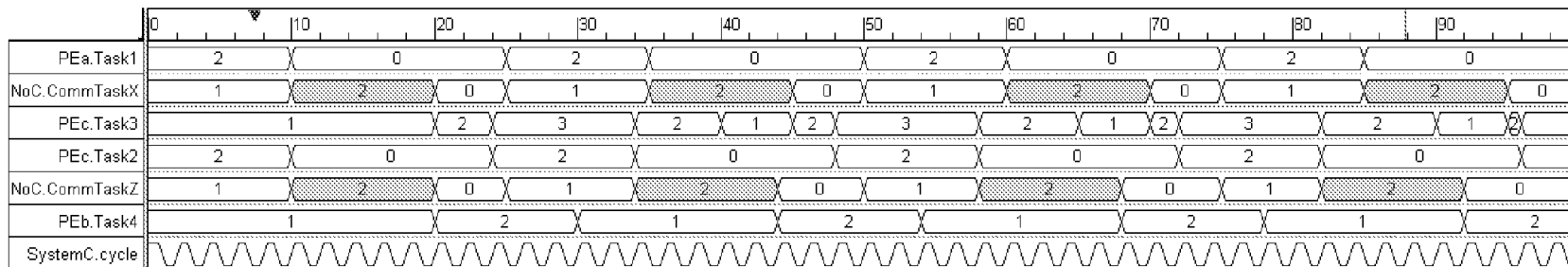
m_x : R_1, L_1, R_2, L_2, R_3

m_z : R_3, L_3, R_1, L_1, R_2

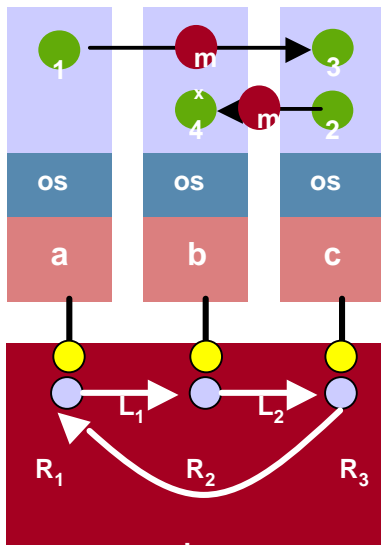
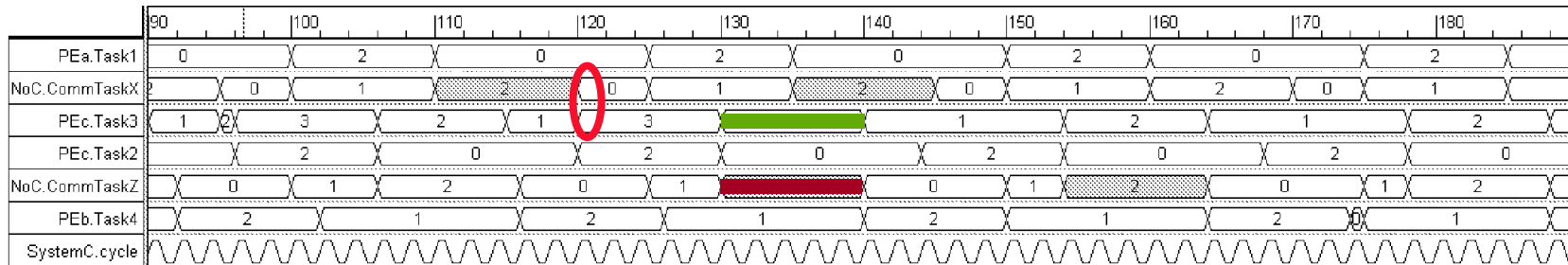




Example: concurrent communication



❖ Example: system deadlock!



Assuming buffers in the routers, there is no problem

But, assuming wormhole routing, message task x would be stalled until task 3 is ready

However, task 3 has to wait for task 2, which cannot complete as it requires the network (L1) to deliver its result

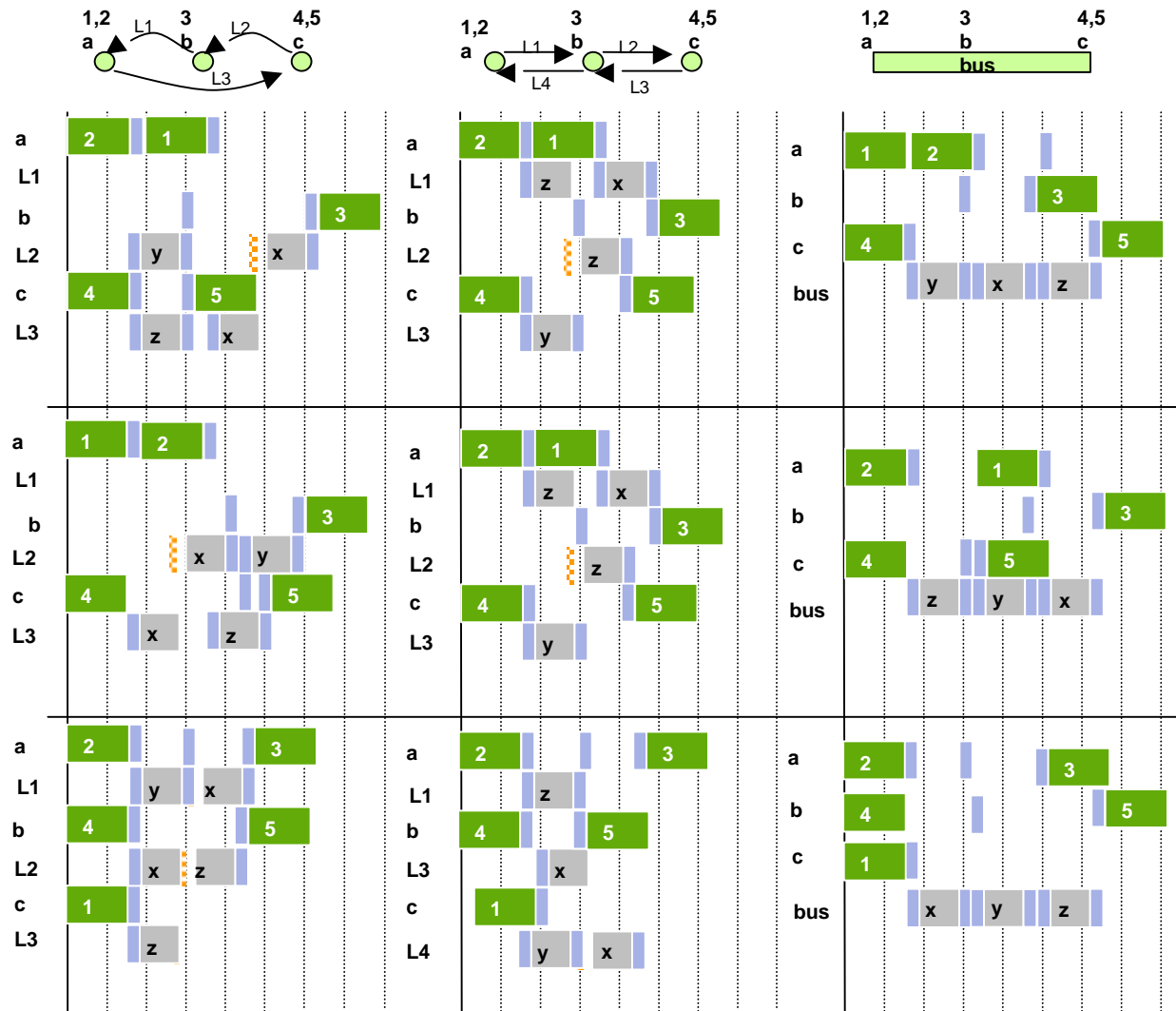


Design space exploration

Timing Aware

QoS Aware
Any traffic from **a**
has higher priority

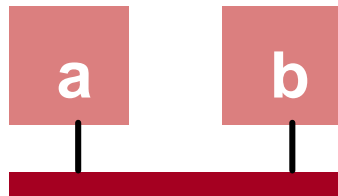
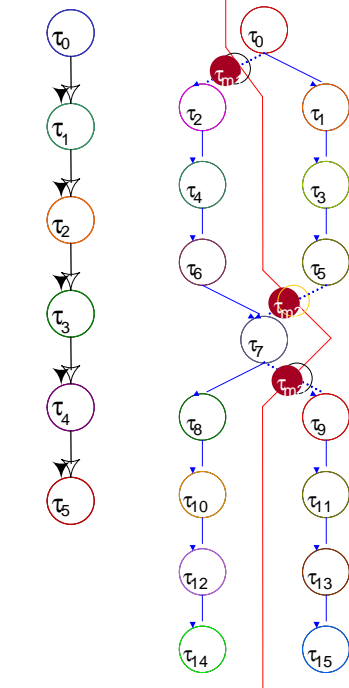
Allocation Aware
Swap task on PEs
to: 2,3 | 4,5 | 1





Example

JPEG Decoder MP3 Decoder



	0us	10us	20us	30us	40us	50us	60us	70us
SystemC.task_mp3_0[7:0]	00				00			00
SystemC.task_mp3_1[7:0]	00	X				01		
SystemC.MessageTask1[7:0]	00				00	X		01
SystemC.task_mp3_2[7:0]	00	X				01		
SystemC.task_mp3_3[7:0]	00	X				01		
SystemC.task_mp3_4[7:0]	00	X				01		
SystemC.task_mp3_5[7:0]	00	X				01		
SystemC.task_mp3_6[7:0]	00	X				01		
SystemC.MessageTask2[7:0]	01	00	X			01		
SystemC.task_mp3_7[7:0]	01	02	00	X	02	00	X	01
SystemC.task_mp3_8[7:0]	01	00	X			01		
SystemC.MessageTask3[7:0]	01	00	X	01	00	X		01
SystemC.task_mp3_9[7:0]	01	00	X			01		
SystemC.task_mp3_10[7:0]	01	X	00	X		01		
SystemC.task_mp3_11[7:0]	01	X	00	X		01		
SystemC.task_mp3_12[7:0]	01	X	02	X	00	X		01
SystemC.task_mp3_13[7:0]	01	X	02	X	00	X		01
SystemC.task_mp3_14[7:0]	01	X	02	X	00	X		01
SystemC.task_mp3_15[7:0]	01	X	02	X	00	X		01
SystemC.task_jpge_0[7:0]	03	03	03	X	02			00
SystemC.task_jpge_1[7:0]	01		X	02	X			00
SystemC.task_jpge_2[7:0]	01			03	02	X		00
SystemC.task_jpge_3[7:0]	01			X	02	X		00
SystemC.task_jpge_4[7:0]	01			X	02	X		00
SystemC.cycle								

Conclusions

- System-level modeling framework which combines application, RTOS and execution platform
- Extension to model network-on-chip
- Example
 - System-level analysis
 - Early design space exploration
- Work in progress
 - Power modeling
 - Reconfigurable platforms
 - Modeling mobile sensor networks