A time-triggered implementation model for real-time distributed systems

Virginia Papailiopoulou
Dumitru Potop-Butucaru
Yves Sorel

INRIA Paris-Rocquencourt

SYNCHRON 2011
Dammarie-les-Lys, France

Outline

- Motivation
  - Avionics embedded computing systems
  - Integrated Modular Avionics (IMA)
- ARINC 653 overview
  - Focus on temporal aspects
- Time-triggered implementation model
  - Reservation/Scheduling tables
- Proposition: time-triggered IMA
  - Time-triggered IMA implementation

Outline

- Motivation
  - Avionics embedded computing systems
  - Integrated Modular Avionics (IMA)
- ARINC 653 overview
  - Focus on temporal aspects
- Time-triggered implementation model
  - Reservation/Scheduling tables
- Proposition: time-triggered IMA
  - Time-triggered IMA implementation

Integrated Modular Avionics (IMA)

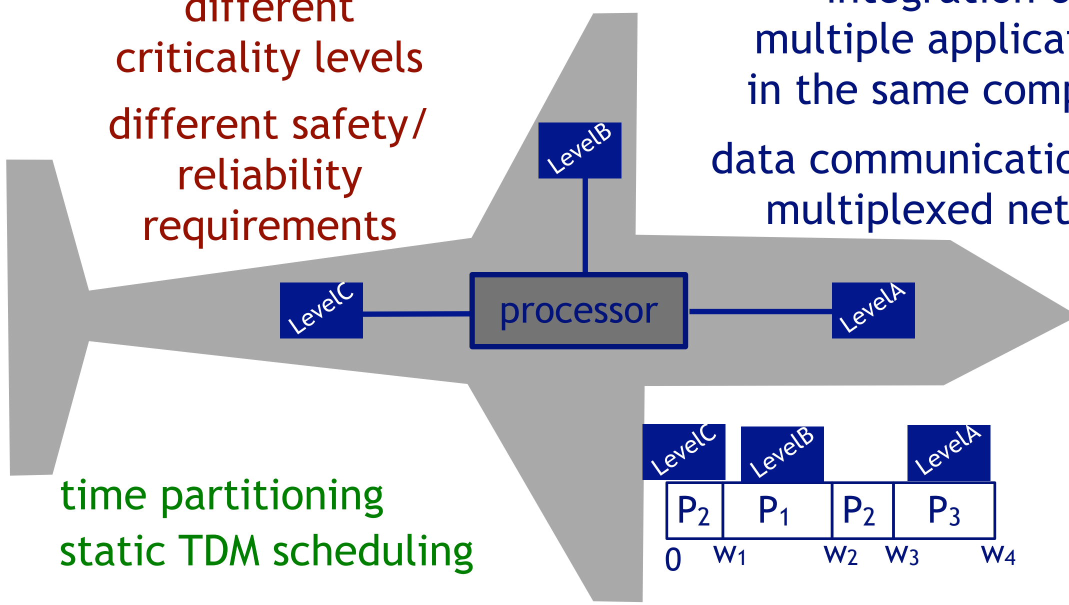
- Better use of hardware resources
- Lower design/maintenance costs

different
criticality levels

different safety/
reliability
requirements

integration of
multiple applications
in the same computer

data communications with
multiplexed network



time partitioning
static TDM scheduling

space partitioning
no unspecified communication
no side-effects

ARINC 653
robust partitioning
2-level scheduling

Integrated Modular Avionics (IMA)

- Inside each partition:
  - Partition-level scheduler (L1) within TDM slots allocated by the static scheduler (L0)
  - Any scheduling policy can be used (RR, EDF, ...)
  - ARINC 653
    - priority-preemptive L1 scheduler
      - easy porting of existing software

Motivation

- Inside each partition:
  - Partition-level scheduler (L1) within TDM slots allocated by the static scheduler (L0)
  - Any scheduling policy can be used (RR, RM, EDF, ...)
  - ARINC 653
    - priority-preemptive L1 scheduler
      - easy porting of existing software
- Dynamic scheduling + static TDM
  - many TDM slots of short duration ➔ increased cost
  - interruption at the end of TDM slots ➔ worse deadline guarantees

Proposition

- Fully static scheduling (L0+L1)
  - Time-triggered process scheduling within partition allocated TDM slots
- Conditional scheduling tables
  - Precise start dates
  - Execution condition

<table>
<tr><th></th><th colspan="2">P1</th><th>P2</th></tr>
<tr><td>t_1</td><td rowspan="2">if c_1 then f</td><td rowspan="2">if $\neg c_1$ then g</td><td></td></tr>
<tr><td>t_2</td><td></td></tr>
<tr><td>t_3</td><td colspan="2"></td><td>if c_3 then h</td></tr>
<tr><td>t_4</td><td colspan="2">if c_4 then f</td><td></td></tr>
</table>

easy and predictable implementations

simple model for complex systems

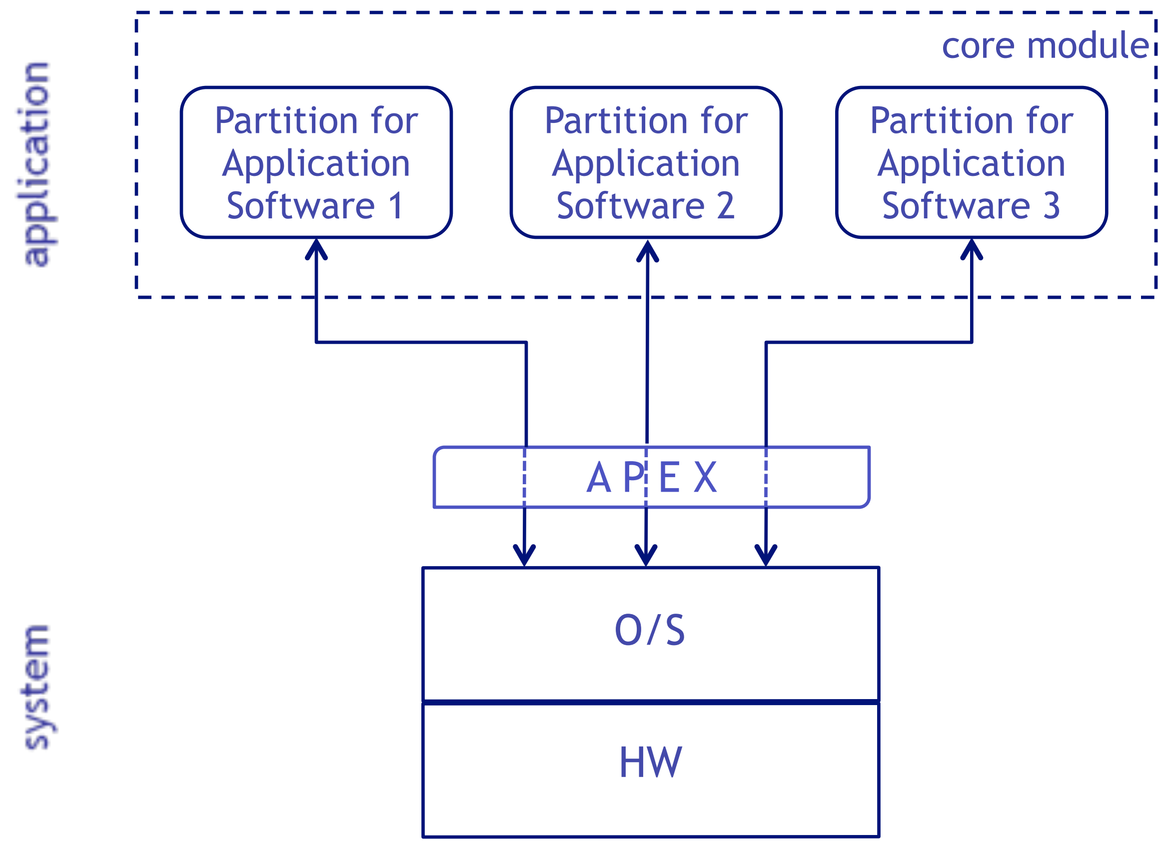
better use of resources

automatic generation from data-flow formalisms, e.g. SCADE or Simulink

Outline

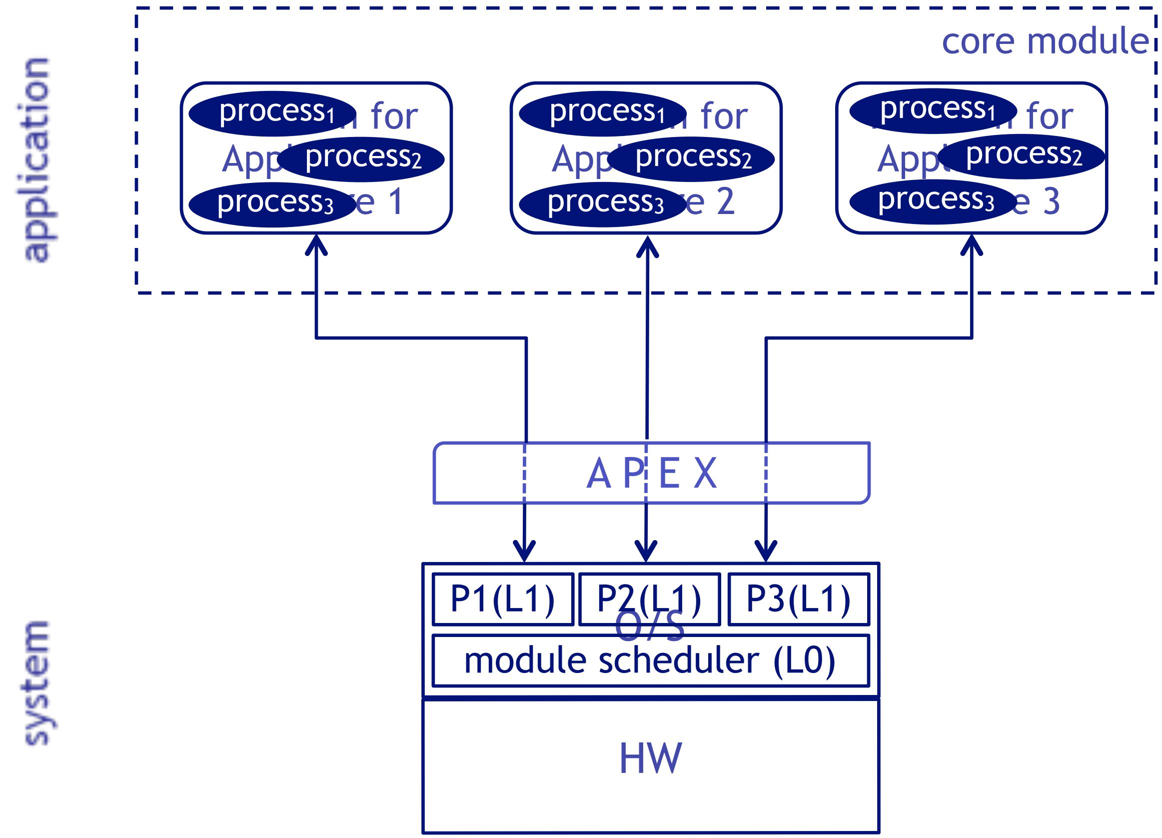
- Motivation
  - Avionics embedded computing systems
  - Integrated Modular Avionics (IMA)
- ARINC 653 overview
  - Focus on temporal aspects
- Time-triggered implementation model
  - Reservation/Scheduling tables
- Proposition: time-triggered IMA
  - Time-triggered IMA implementation

ARINC 653



application
system
core module
Partition for Application Software 1
Partition for Application Software 2
Partition for Application Software 3
A P E X
O/S
HW

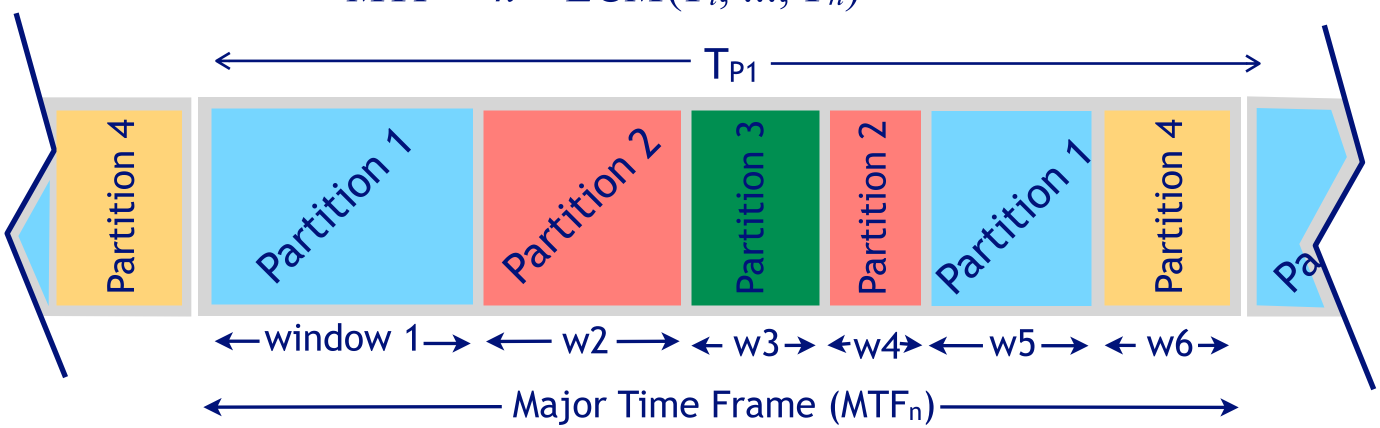
ARINC 653


application
system
core module
process1
for
App
process2
process3
e 1
e 2
e 3
A P E X
P1(L1)
P2(L1)
P3(L1)
O/S
module scheduler (L0)
HW

ARINC 653 - Partitions

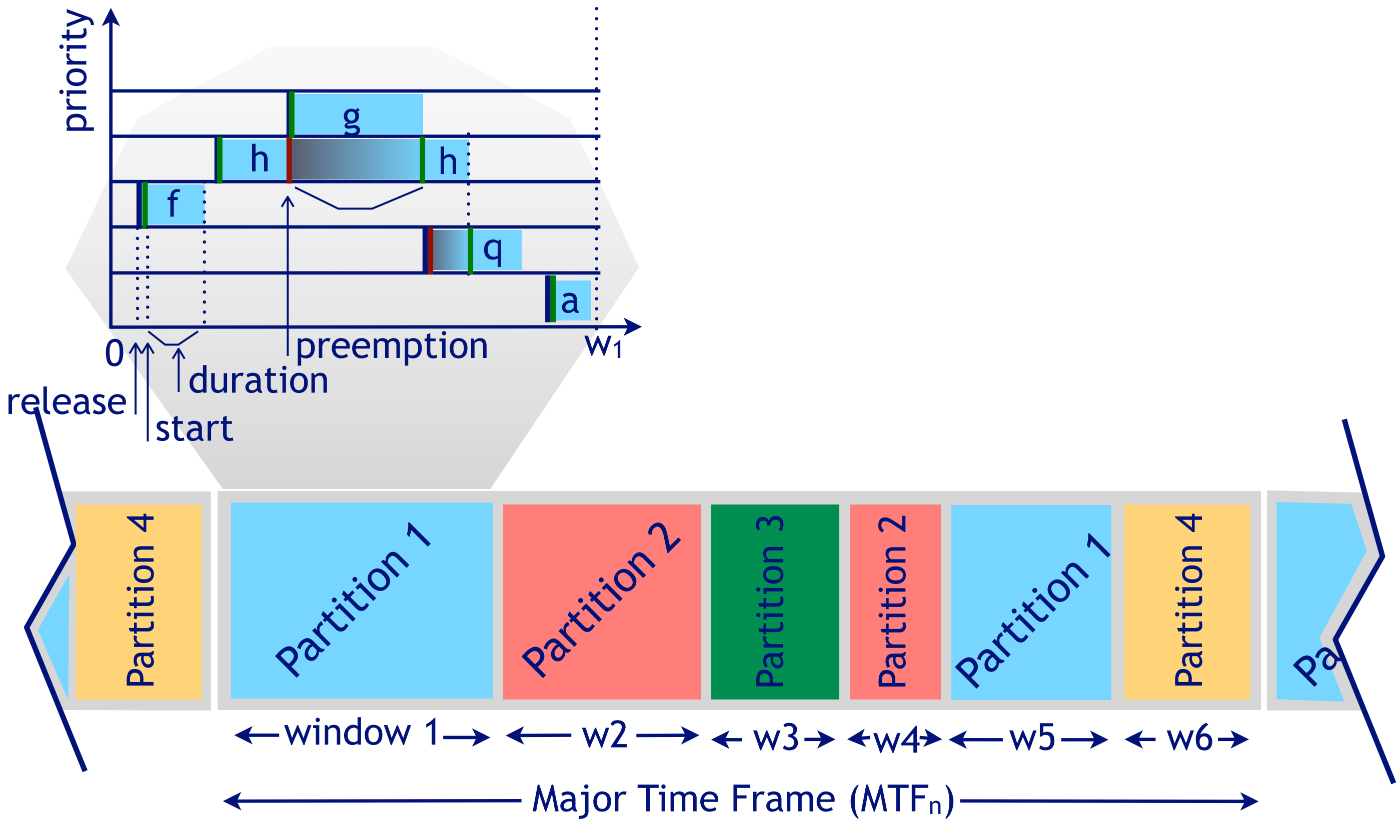
- Static allocation of resources
  - One partition ↔ one application
- Static scheduling ➔ TDM
  - Fixed time windows ➔ exclusive access to resources
  - One partition ➔ several windows

$$MTF = k \times LCM(T_i, \ldots, T_n)$$



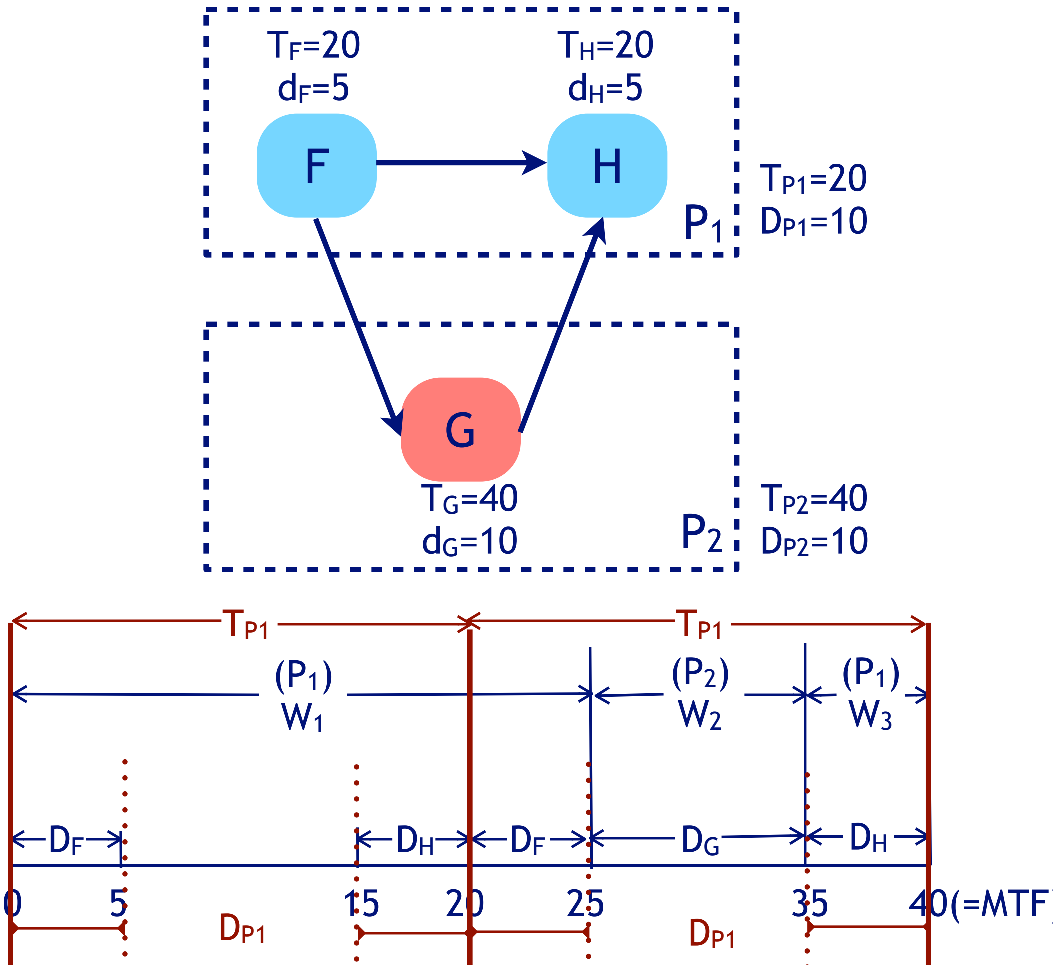
ARINC 653 - Processes

- Application functional behavior
- Priority preemptive scheduling



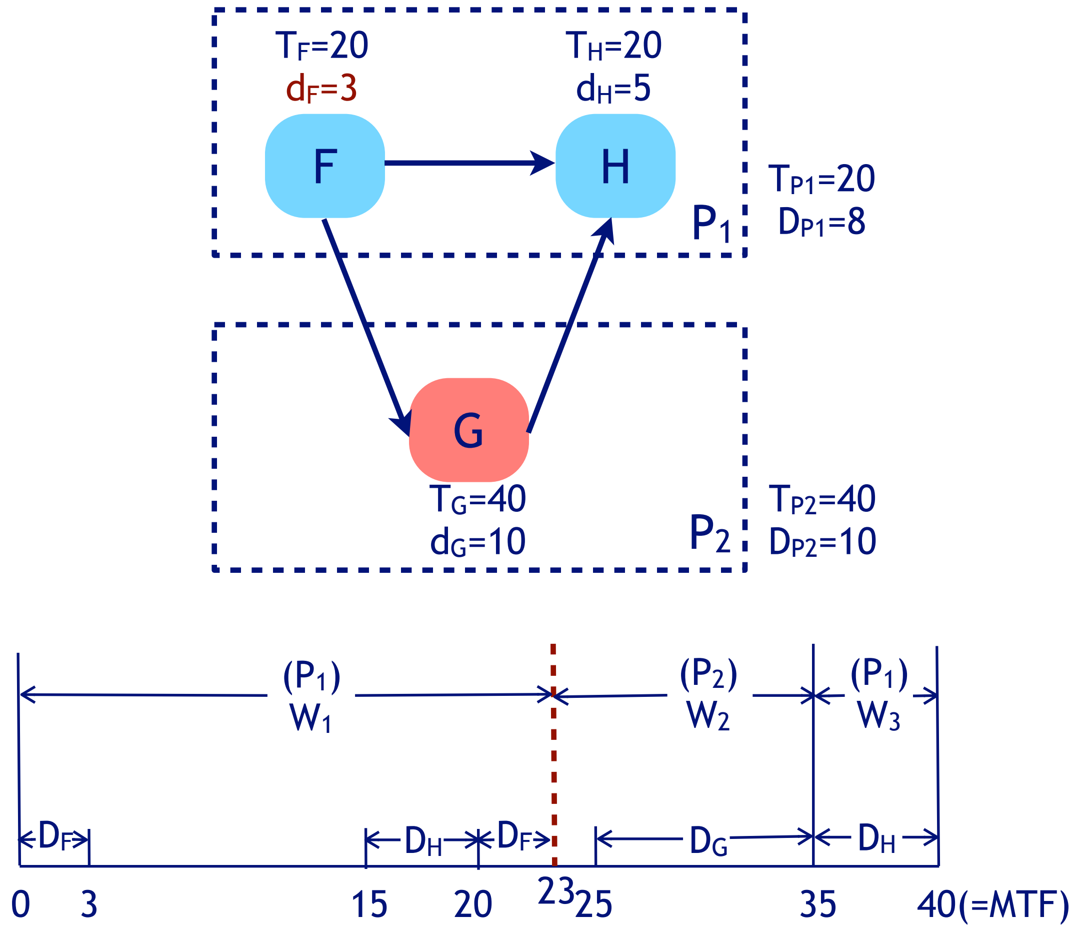
ARINC 653 - MTF configuration

- According to partition and process requirements



ARINC 653 - MTF configuration

- Not always unique



ARINC 653 - Structure of an implementation

- Configuration file (for the O/S)
  - Module configuration
    - window allocation to partitions
    - memory management
    - module scheduling (window start dates + durations)
- Main programs
  - One for each partition
    - processes creation
    - communication ports creation
    - partition scheduling

Outline

- Motivation
  - Avionics embedded computing systems
  - Integrated Modular Avionics (IMA)
- ARINC 653 overview
  - Focus on temporal aspects
- Time-triggered implementation model
  - Reservation/Scheduling tables
- Proposition: time-triggered IMA
  - Time-triggered IMA implementation

INRIA

Time-triggered implementation model

non-partitioned

resources

time	P_1	P_2	P_3	Bus
0	F_1@true			
1				send(P_1,inA)@true
2	F_2@inA=true			send(P_3,inB)@true
3			F_3@inB =false	send(P_3,inA)@true
4			N@inA =false	
5				
6		M@true		
7				

- Periodic non-preemptive execution model
- Table size = execution cycle duration
- Operations with disjoint conditions can run concurrently
- Data dependencies respected
- No data race

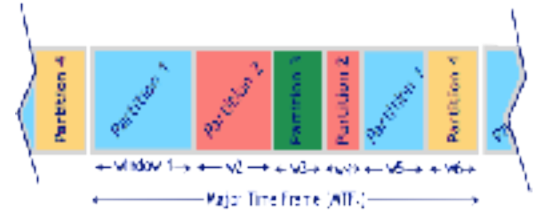
Outline

- Motivation
  - Avionics embedded computing systems
  - Integrated Modular Avionics (IMA)
- ARINC 653 overview
  - Focus on temporal aspects
- Time-triggered implementation model
  - Reservation/Scheduling tables
- Proposition: time-triggered IMA
  - Time-triggered IMA implementation

Time-triggered IMA

	P1		P2
t_1	$f@c_1$	$g@\neg c_1$	
t_2			$h@c_3$
t_3	$f@c_4$		

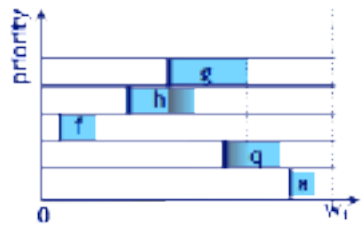
consistent



- Table size = MTF
- Each operation o_i is associated to a partition P_i
- Slot reservation for window changes
- Precomputed preemption
  - Allow for operations spanning over several windows
  - Multiple reservations per operation

Time-triggered implementation

	P1		P2
t_1	$f@c_1$	$g@\neg c_1$	
t_2			$h@c_3$
t_3	$f@c_4$		



+ APEX

- For each partition
  - One aperiodic process/scheduled operation
  - One periodic process/slot change
  - Fixed priorities
    - higher priority given to periodic processes
  - Start dates fixed w.r.t. the partition period

Time-triggered implementation

```
#include "local_definitions.c"
const int OpNum ;   # of operations
const int DNum ;    # of start/end dates
```

processes associated to operations

```
PROCESS_ATTRIBUTE_TYPE* op(int OPi){
  return
    {op_name[OPi],op_wrapper[OPi],
     op_stack[OPi],LO_PRIO,0,
     op_duration[OPi],HARD};
}
PROCESS_ID_TYPE OP_PID[OpNum] ;

int d_i= DNum-1;
void slot_change() { scheduler function
  RETURN_CODE_TYPE ret ;
  d_i = (d_i+1)%DNum ;
  for(int i=0;i<OpNum;i++) {
    if((op_start[d_i][i])())
      START(OP_PID[i],&ret);
    else if ((op_resume[d_i][i])())
      RESUME(OP_PID[i],&ret);
    else if ((op_suspend[d_i][i])())
      SUSPEND(OP_PID[i],&ret);
} }
```

1

processes associated to slot changes and the start dates

```
PROCESS_ATTRIBUTE_TYPE* dates(int Di){
  return
    {date_name[Di],slot_change,
     date_stack,HI_PRIO,part_period,
     date_duration,HARD};
}
const SYSTEM_TIME_TYPE slot_offset[DNum];
```

inter-partition ports creation

```
void init_inter_partition_ports() ;

int main() {
  RETURN_CODE_TYPE ret ;
  PROCESS_ID_TYPE d_pid ;
  init_inter_partition_ports() ;
  for(int i=0;i<OpNum;i++)
    CREATE_PROCESS(op(i),OP_PID+i,&ret);
  for(int i=0;i<DNum;i++) {
    CREATE_PROCESS(dates,&d_pid,&ret);
    DELAYED_START(d_pid,
                  slot_offset[i],&ret);
  }
```

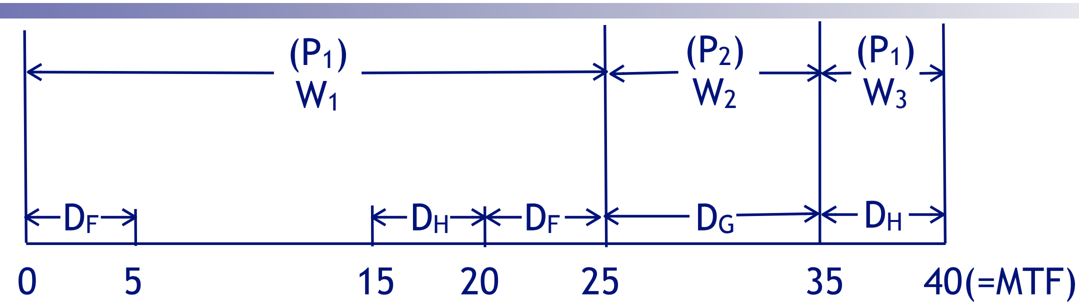
initializations

partition execution

```
  SET_PARTITION_MODE(NORMAL,&ret);
  return 0 ; }
```

2

Time-triggered implementation



- No preemption → no aperiodic process

```
#include "local_definitions.c"
const int opNum = 2;
PROCESS_ATTRIBUTE_TYPE op[opNum]= {
  {"f", f, 1000, LO_PRIO, 0.020, 0.005, HARD},
  {"h", h, 1000, LO_PRIO, 0.020, 0.005, HARD}};
SYSTEM_TIME_TYPE slot_offset[opNum] = {0.005, 0.020} ;
PROCESS_ID_TYPE OP_PID[opNum];

int main () {
  RETURN_CODE_TYPE ret;
  init_inter_partition_ports();
  for(int i=0;i<opNum;i++){
    CREATE_PROCESS(op[i],OP_PID+i,&ret);
    DELAYED_START(OP_PID[i],slot_offset[i],&ret);
  }

  SET_PARTITION_MODE(NORMAL, &ret);
  return (0);
}
```

Conclusion

ARINC 653 time partitioning constraints

+

conditional scheduling tables

⇓

fully RT static schedule

⇓

Automatic synthesis of ARINC-based implementations from data-flow specifications

- Future work
  - L1 scheduler implementation
  - Evaluation