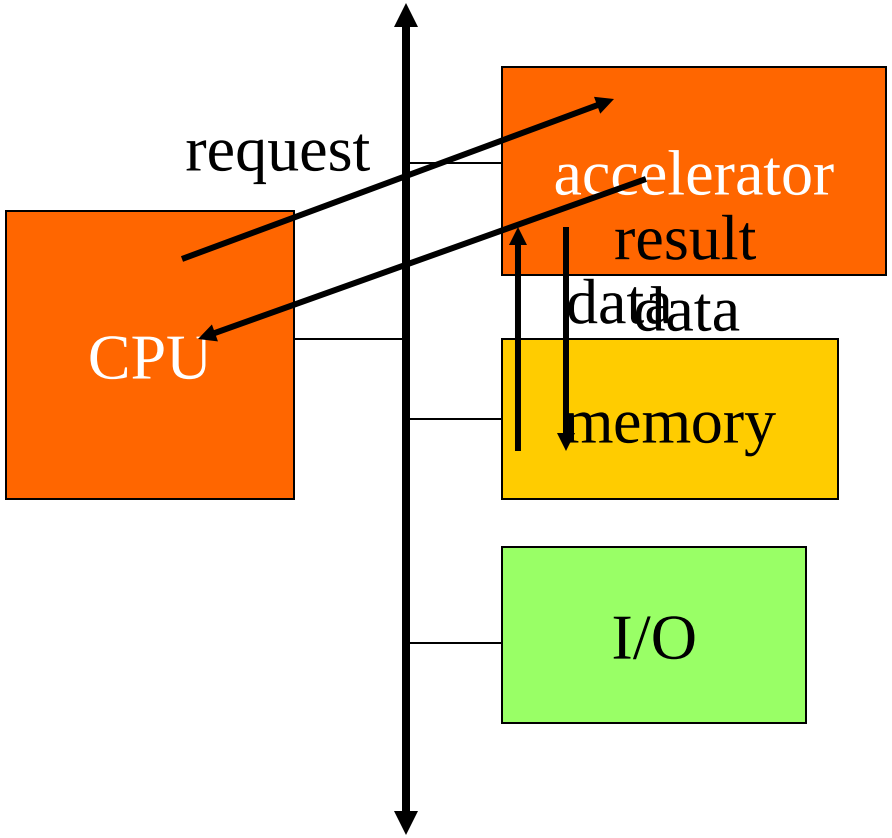
Accelerated systems

- Use additional computational unit dedicated to some functions?
  - Hardwired logic.
  - Extra CPU.
- Hardware/software co-design: joint design of hardware and software architectures.

Accelerated system architecture



Accelerator vs. co-processor

- A co-processor executes instructions with op-code.
  - Instructions are dispatched by the CPU.
- An accelerator appears as a device on the bus.
  - Its programming model interface is functionally equivalent to an I/O device although it does not perform input or output
  - is controlled by it registers.
  - Doe not execute instructions

System design tasks

- Design a heterogeneous multiprocessor architecture.
  - Processing element (PE): CPU, coprocessor, accelerator, etc.
- Program the system.

Accelerated system design

- First, determine that the system really needs to be accelerated.
  - How much faster is the accelerator on the core function?
  - How much data transfer overhead?
- Design the accelerator itself.
- Design CPU interface to accelerator.

Accelerator implementations

- Application-specific integrated circuit.
- Field-programmable gate array (FPGA).
- Standard component.
  - Example: graphics processor.

Accelerated system platforms

- Several off-the-shelf boards are available for acceleration in PCs:
  - FPGA-based core;
  - PC bus interface.

Accelerator/CPU interface

- Accelerator registers provide control registers for CPU.
- Data registers (buffers) can be used for small data objects.
- Accelerator may include special-purpose read/write logic.
  - Especially valuable for large data transfers.
  - DMA to transfer a large volume of data without intervention of CPU

Accelerator/CPU interface

- Design CPU-side interface
  - Application software need to talk to the accelerator (data, instruction)
  - Synchronization between CPU and accelerator
    - The accelerator should know when it has the required data
    - The CPU should know when it has received the designed reuslts.

System integration/debugging

- Try to debug the CPU/accelerator interface separately from the accelerator core.
- Build scaffolding to test the accelerator.
- Hardware/software co-simulation can be useful.

Caching problems

- Main memory provides the primary data transfer mechanism to the accelerator.
- Programs must ensure that caching does not invalidate main memory data.
  - CPU reads location S.
    - Cache can be updated
  - Accelerator writes location S.
    - Main memory is updated
  - CPU reads location S.
    - Cache and main memory have different data

Shared memory

- When CPU and accelerator operate concurrently and communicated through shared memory
- As with cache, main memory writes to shared memory may cause invalidation:
  - CPU reads S.
  - Accelerator writes S.
  - CPU reads S.

Synchronization

- CPU write data into a memory buffer
- Start the accelerator
- Wait for the accelerator to finish
- Read the shared memory

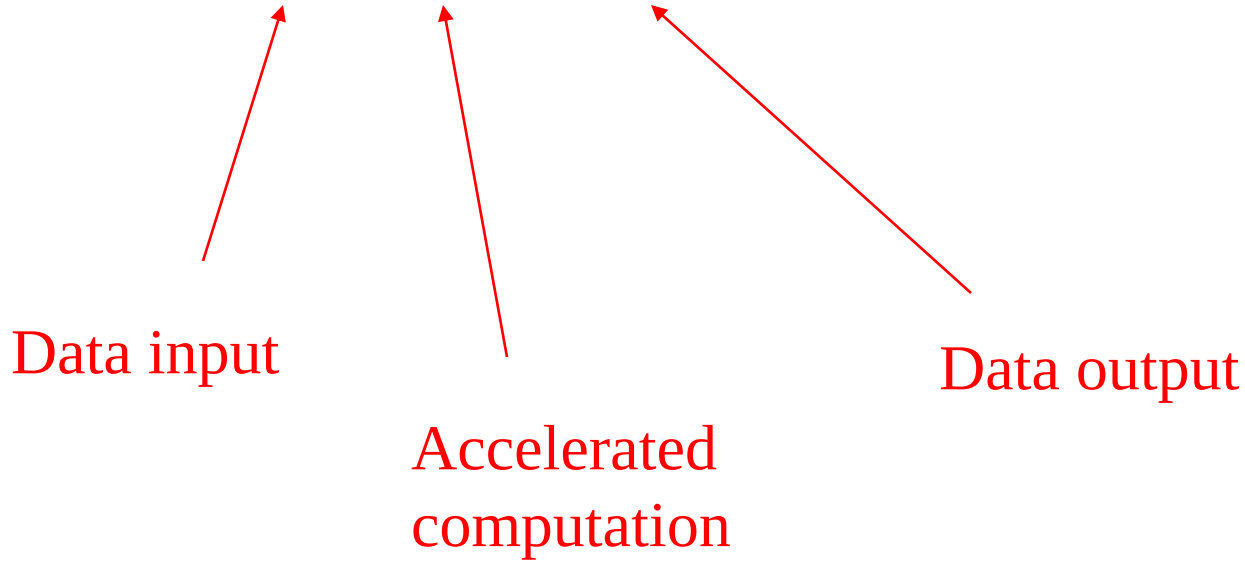
- Need to use the accelerator's status register as a semaphore
- Atomic test-and-set operation

Accelerator speedup

- Critical parameter is speedup: how much faster is the system with the accelerator?
- Must take into account:
  - Accelerator execution time.
  - Data transfer time.
  - Synchronization with the master CPU.

Accelerator execution time

- Total accelerator execution time:
  - $t_{accel} = t_{in} + t_x + t_{out}$



  - $t_{accel} = \max \{t_{in}, t_x, t_{out}\}$ if pipelined

Accelerator speedup

- Assume loop is executed n times.
- If the software loop is replaced with the accelerator, compare accelerated system to non-accelerated system:
  - $S = n(t_{CPU} - t_{accel})$
    $= n[t_{CPU} - (t_{in} + t_x + t_{out})]$

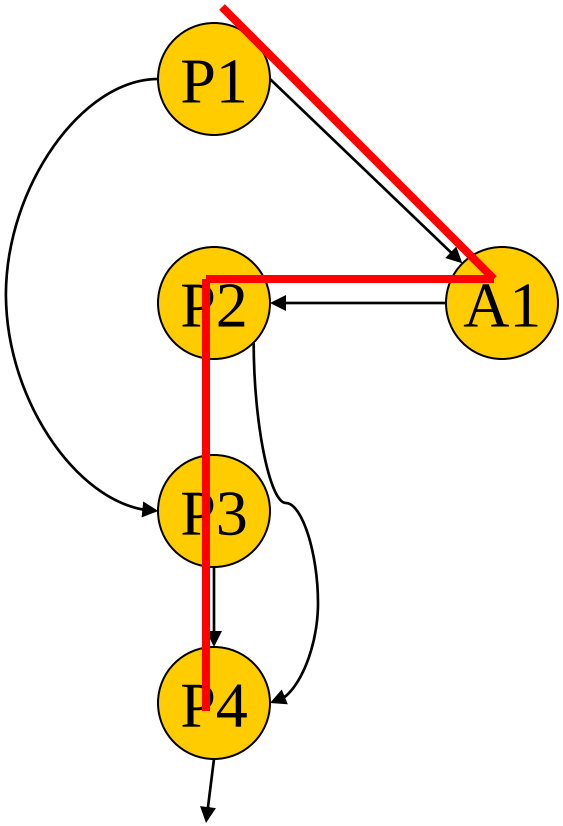
Execution time on CPU

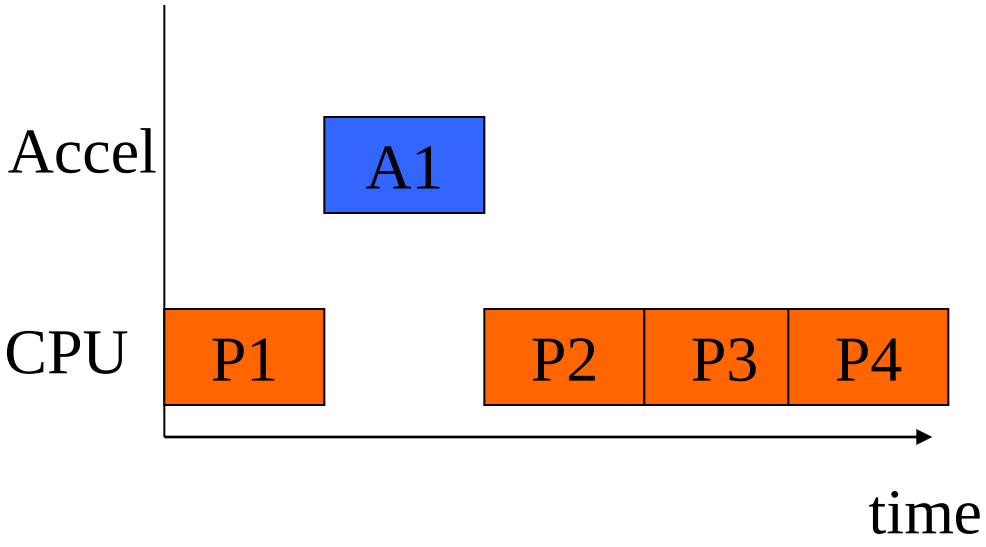
Single- vs. multi-threaded

- One critical factor is available parallelism:
  - single-threaded/blocking: CPU waits for accelerator;
  - multithreaded/non-blocking: CPU continues to execute along with accelerator.
- To multithread, CPU must have useful work to do.
  - Synchronization: software must also support multithreading.

Total execution time

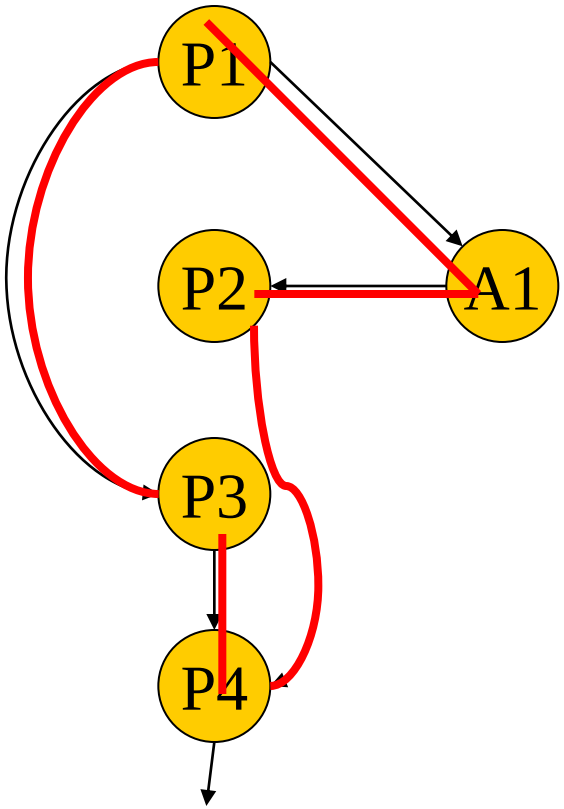
- Single-threaded:

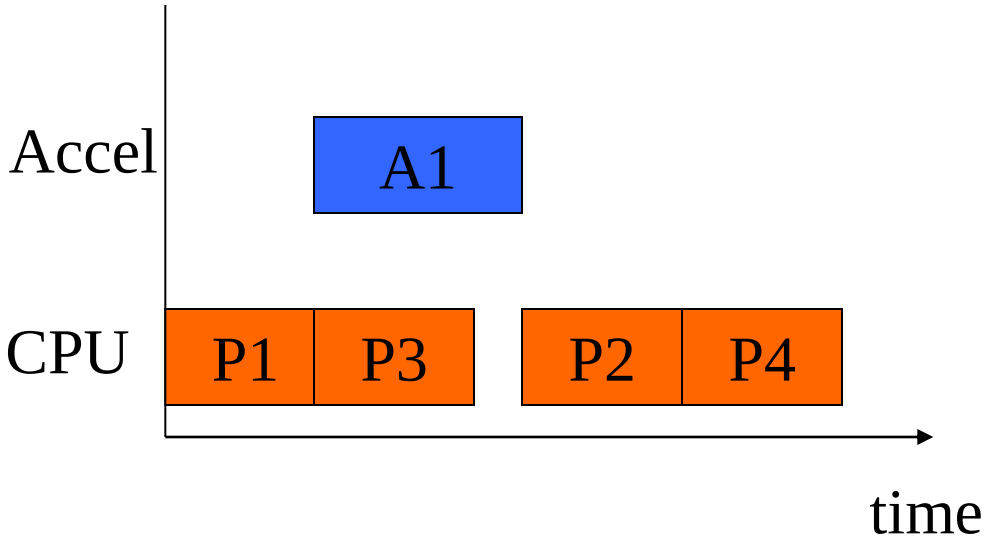




Total execution time

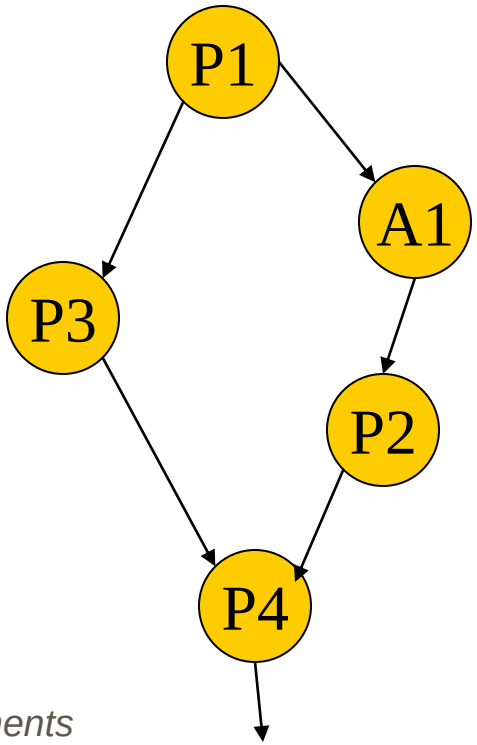
- Multi-threaded:





Execution time analysis

- Single-threaded:
  - Count execution time of all component processes.
- Multi-threaded:
  - Find longest path through execution.



Sources of parallelism

- Overlap I/O and accelerator computation.
  - Perform operations in batches, read in second batch of data while computing on first batch.
- Find other work to do on the CPU.
  - May reschedule operations to move work after accelerator initiation.

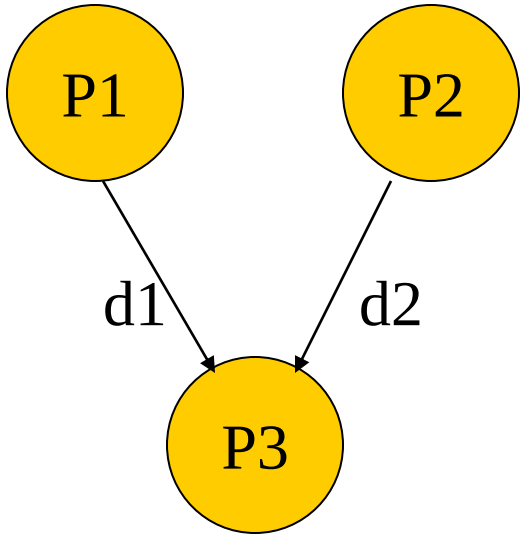
Data input/output times

- Bus transactions include:
  - flushing register/cache values to main memory if necessary;
  - time required for CPU to set up transaction;
  - overhead of data transfers by bus packets, handshaking, etc.

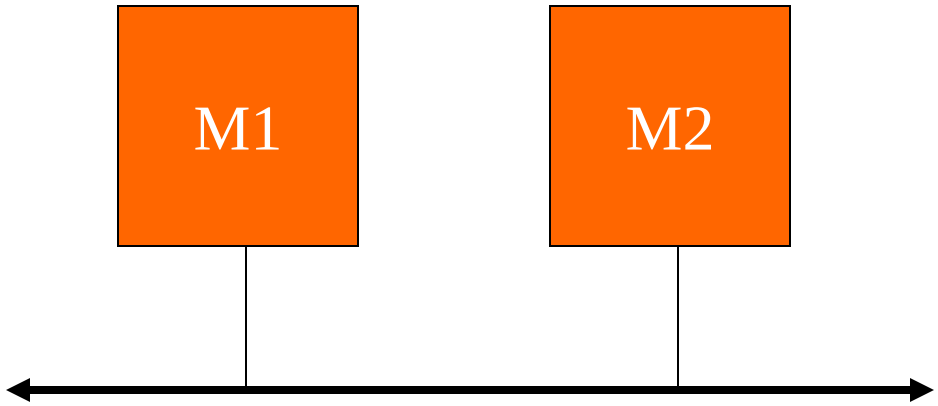
Scheduling and allocation

- Must:
  - schedule operations in time;
  - allocate computations to processing elements.
- Scheduling and allocation interact, but separating them helps.
  - (Alternatively) allocate, then schedule.

Example: scheduling and allocation



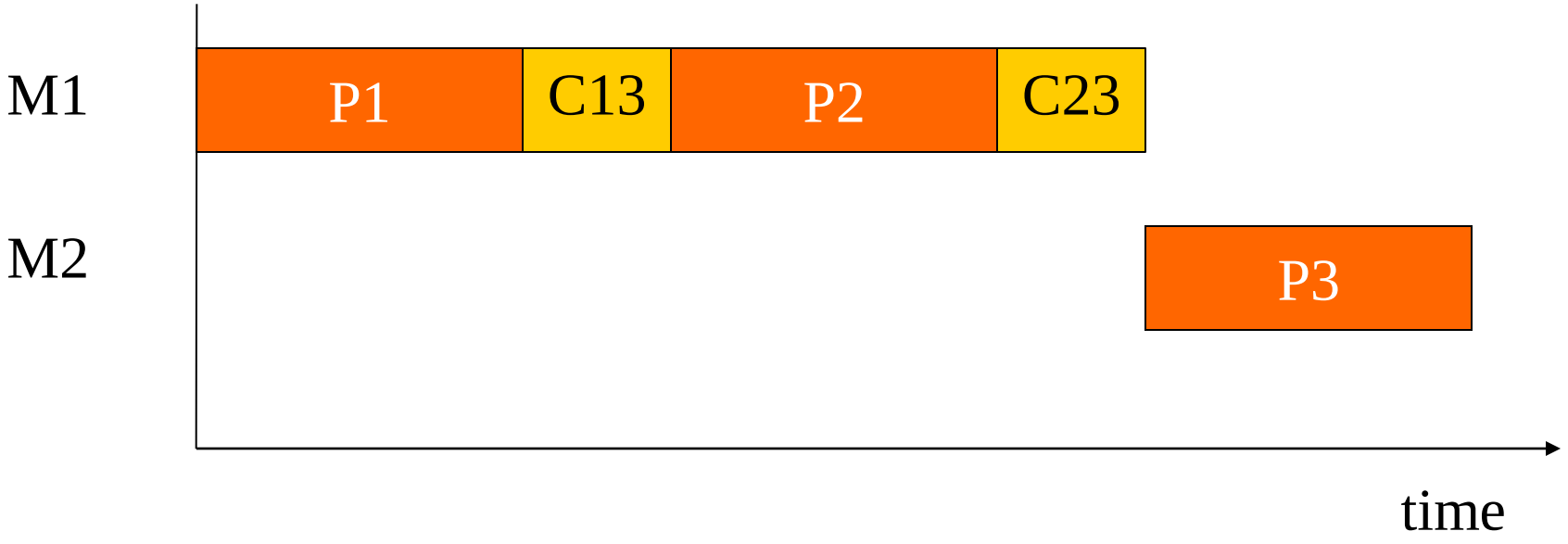
Task graph



Hardware platform

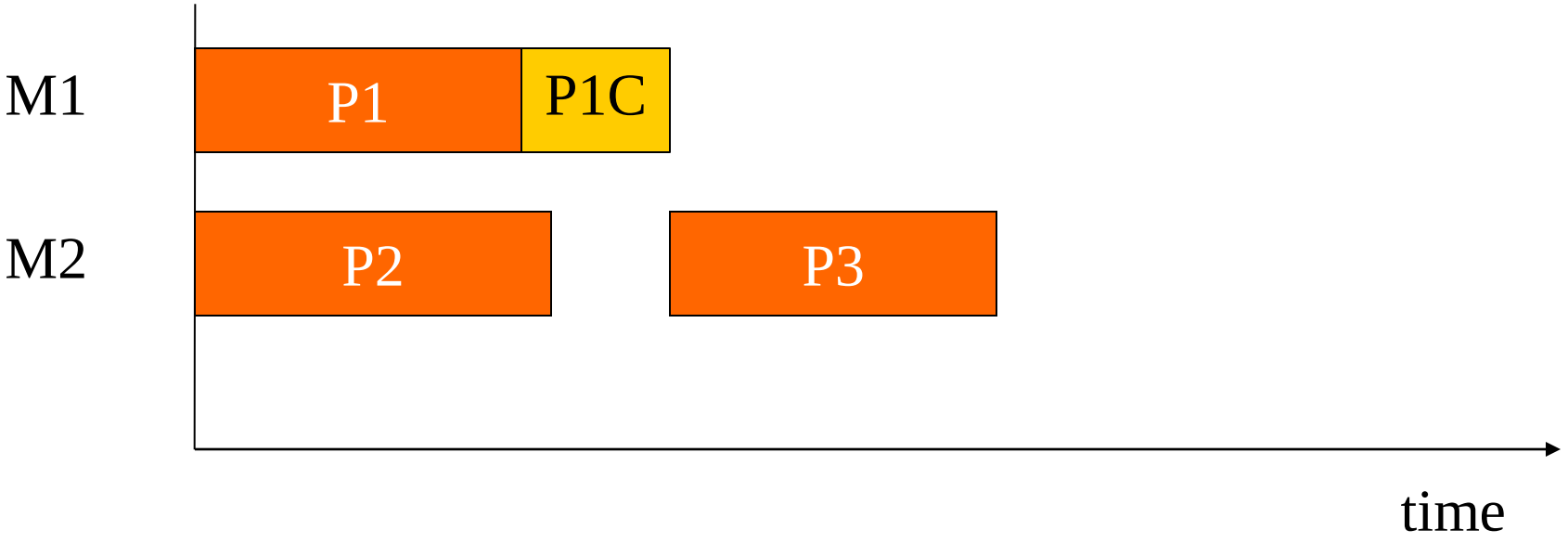
First design

- Allocate P1, P2 -> M1; P3 -> M2.



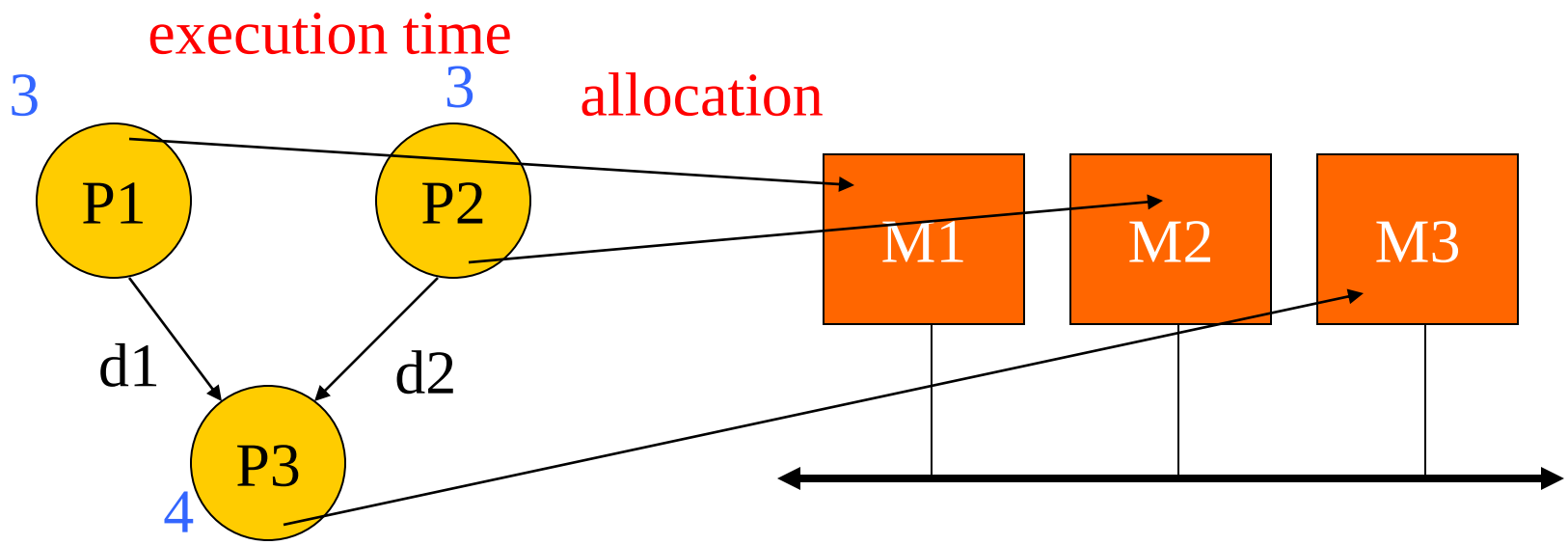
Second design

- Allocate P1 -> M1; P2, P3 -> M2:



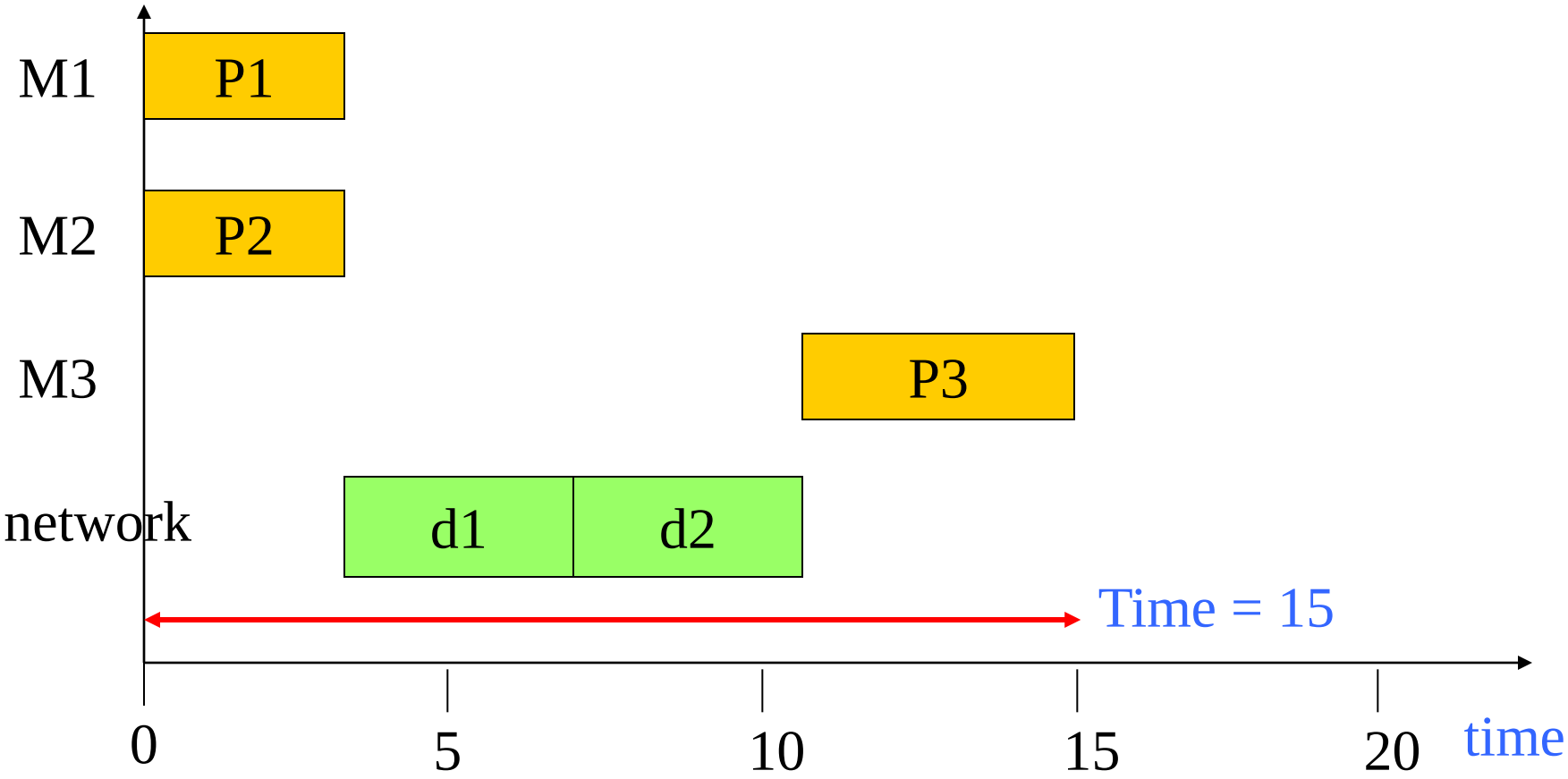
Example: adjusting messages to reduce delay

- Task graph:
- Network:



Transmission time = Td1=Td2=4

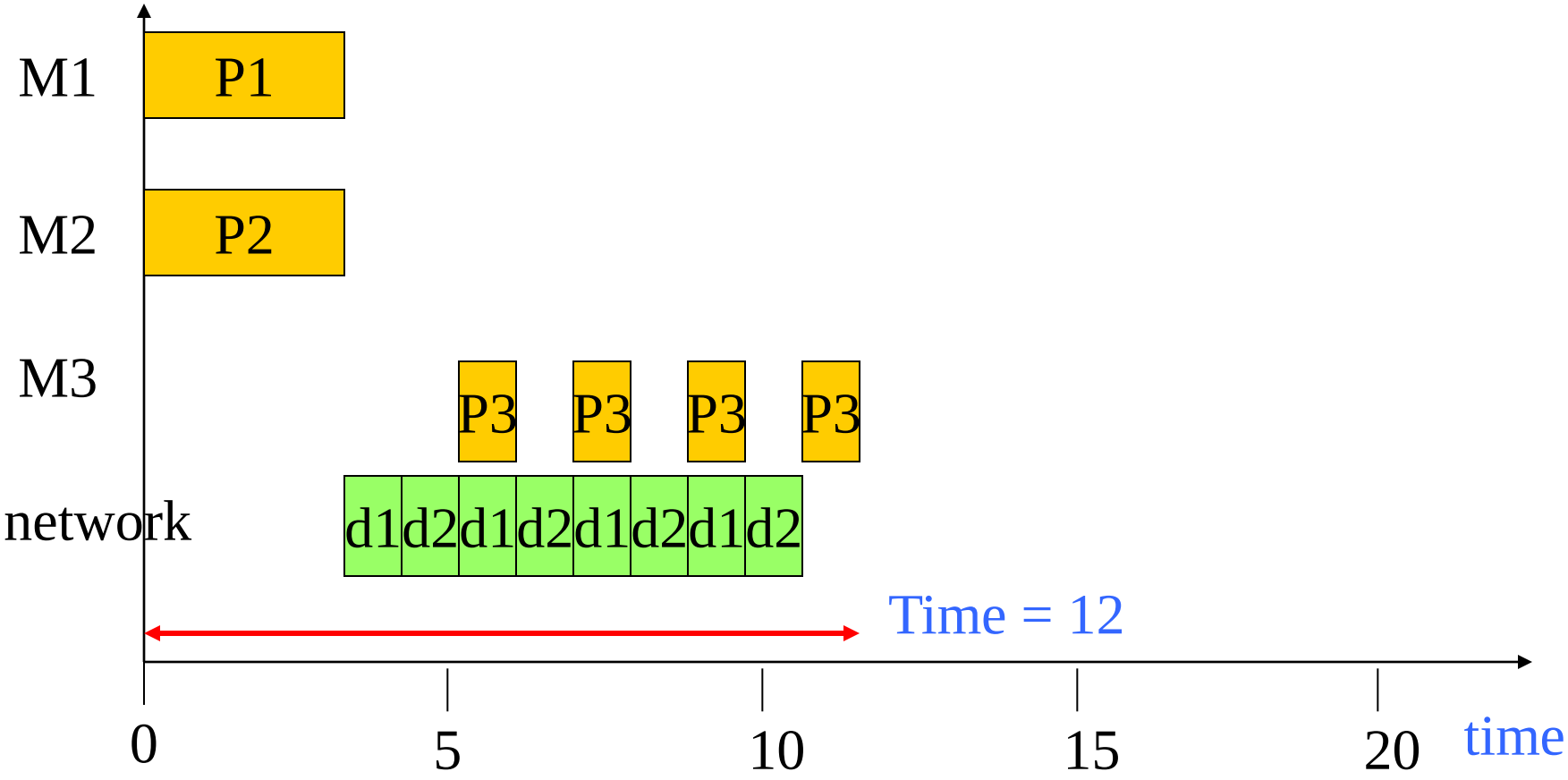
Initial schedule



New design

- Modify P3:
  - reads one packet of d1, one packet of d2
  - computes partial result
  - continues to next packet

New schedule

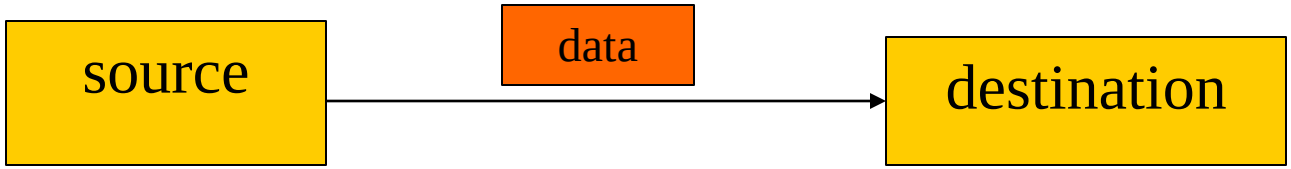


Buffering and performance

- Moving data in microprocessor incurs significant and sometimes unpredictable costs
- Buffering may sequentialize operations.
  - Next process must wait for data to enter buffer before it can continue.
- Buffer policy (queue, RAM) affects available parallelism.

Copying an array

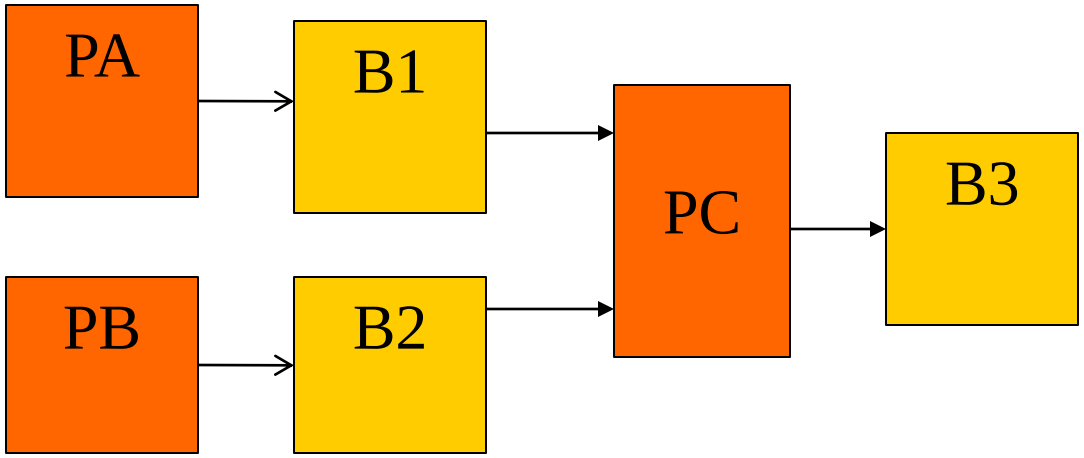
- Read a data from the source memory
- Transfer the data through the bus
- Write the data into the destination memory
- The transfer rate is limited by the slowest element



Buffers and latency

- Three processes separated by buffers:

PA: copying array A
PB: copying array B
PC: computing C[i]= f(A[i],B[i])



Buffers and latency schedules

(2n+1) cycle latency

A[0]
A[1]
…
B[0]
B[1]
…
C[0]
C[1]
…

Must wait for all of A before getting any B

3-cycle latency

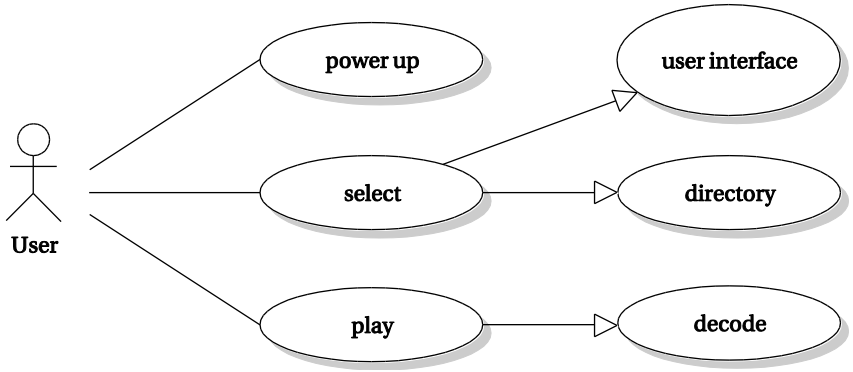
A[0]
B[0]
C[0]
A[1]
B[1]
C[1]
…

Multiprocessors

- Consumer electronics systems.
- Cell phones.
- CDs and DVDs.
- Audio players.
- Digital still cameras.

Consumer electronics use cases

- Multimedia: stored in compressed form, uncompressed on viewing.
- Data storage and management: keep track of your multimedia, etc.
- Communication: download, upload, chat.

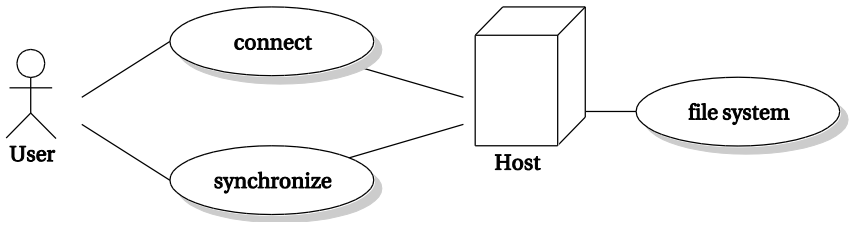


Non-functional requirements for CE

- Often battery-operated, strict power budget.,
- Very inexpensive.
- User interface must be capable but inexpensive.

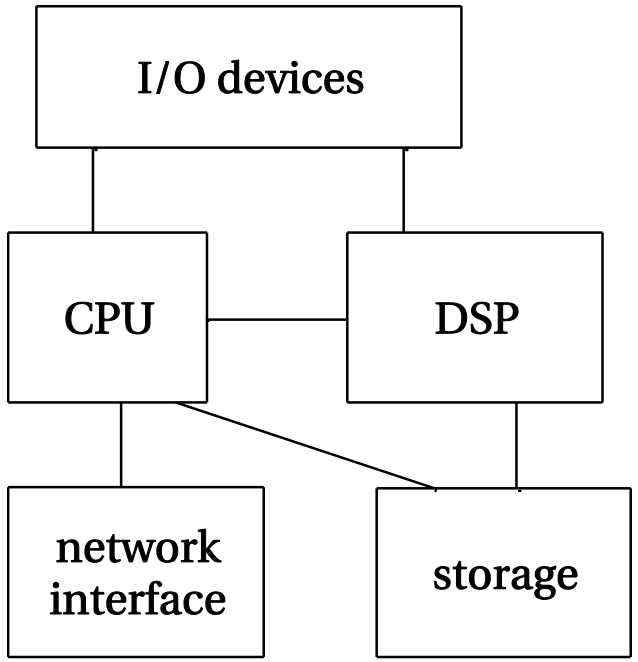
CE devices and hosts

- Many devices talk to host system.
  - PC host does things that are hard to do on the device.
- Increasingly, CE devices communicate directly over the internet, avoiding the host for access.



Platforms and OS

- Many CE devices use a DSP for signal processing and a RISC CPU for other tasks.
- OS runs on the CPU
  - maintain processes for concurrency and file systems
- I/O devices include buttons, screen, USB.

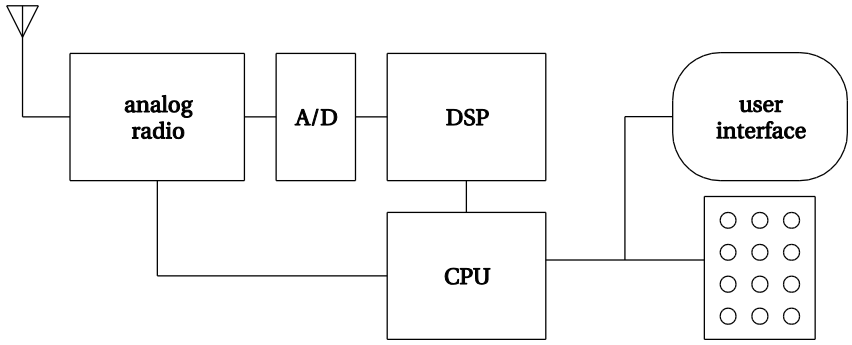


Flash memory

- Flash is widely used for mass storage.
- Flash wears out on writing (up to 1 million cycles).
  - Directory is most often written, wears out first.
- Flash file system has layer that moves contents to levelize wear.
  - Hides wear leveling from API.

Cell phone platforms

- Today's cell phones use analog front end, digital baseband processing.
  - Future cell phones will perform IF processing with DSP.
- Baseband processing in DSP:
  - Voice compression.
  - Network protocol.
- Other processing:
  - Multimedia functions.
  - User interface.
  - File system.
  - Applications (contacts, etc.)



Mobile Phone Trends

TABLE I
MOBILE PHONE TRENDS IN 5-YEAR INTERVALS.

year	1995	2000	2005	2010	2015
cellular generation	2G	2.5-3G	3.5G	pre-4G	4G
cellular standards	GSM	GPRS UMTS	HSPA	HSPA LTE	LTE LTE-A
downlink bitrate [Mb/s]	0.01	0.1	1	10	100
display pixels [×1000]	4	16	64	256	1024
battery energy [Wh]	1	2	3	4	5
CMOS [ITRS, nm]	350	180	90	50	25
PC CPU clock[MHz]	100	1000	3000	6000	8500
PC CPU power [W]	5	20	100	200	200
PC CPU MHz/W	20	50	30	30	42
phone CPU clock[MHz]	20	100	200	500	1000
phone CPU power [W]	0.05	0.05	0.1	0.2	0.3
phone CPU MHz/W	400	2000	2000	2500	3000
workload [GOPS]	0.1	1	10	100	1000
software [MB]	0.1	1	10	100	1000
#programmable cores	1	2	4	8	16

Power and Battery Capacity

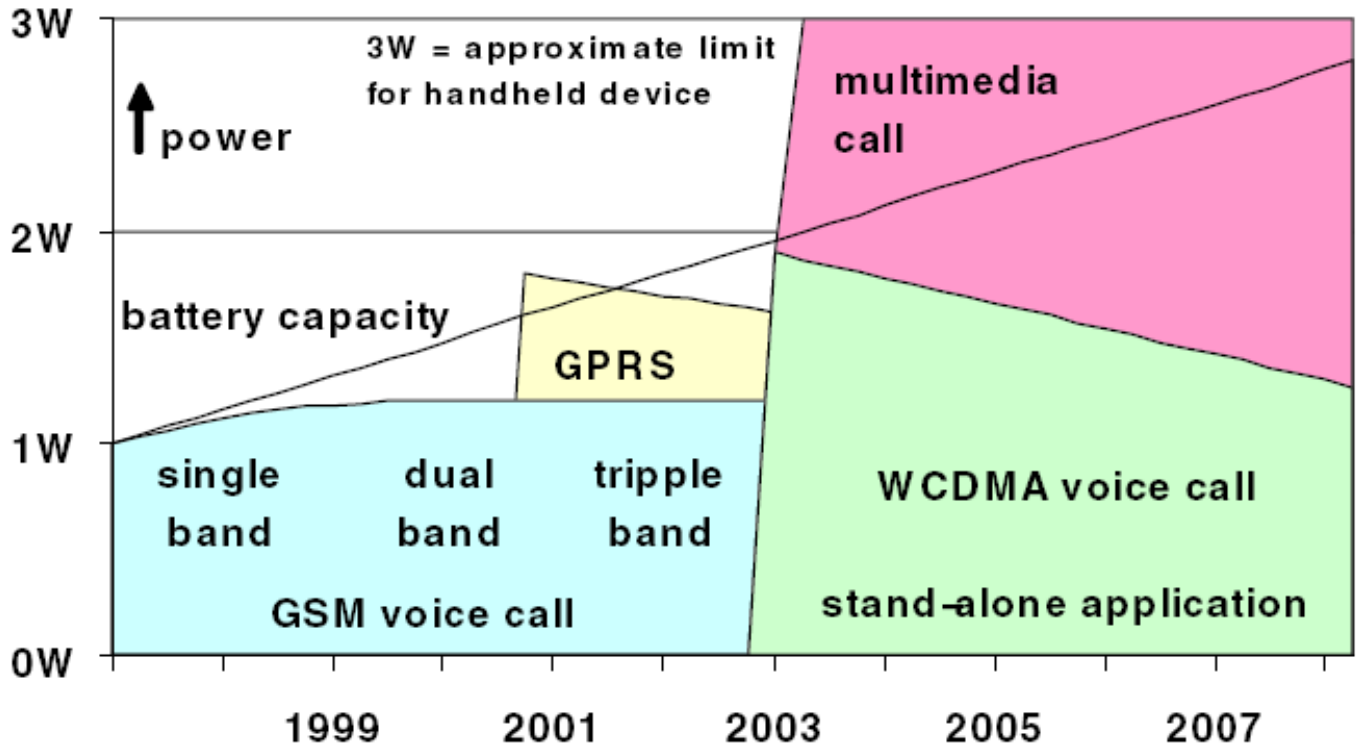
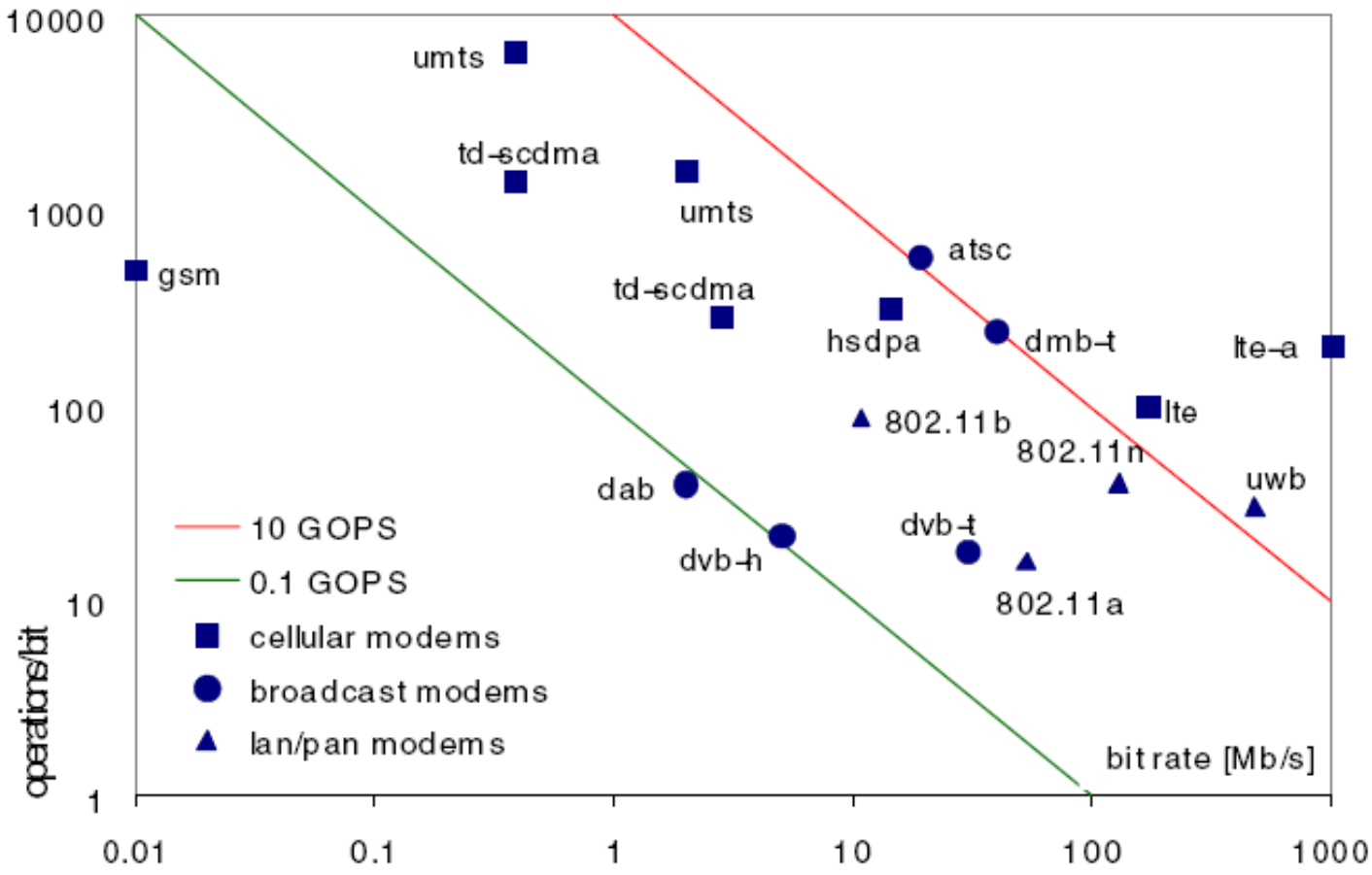
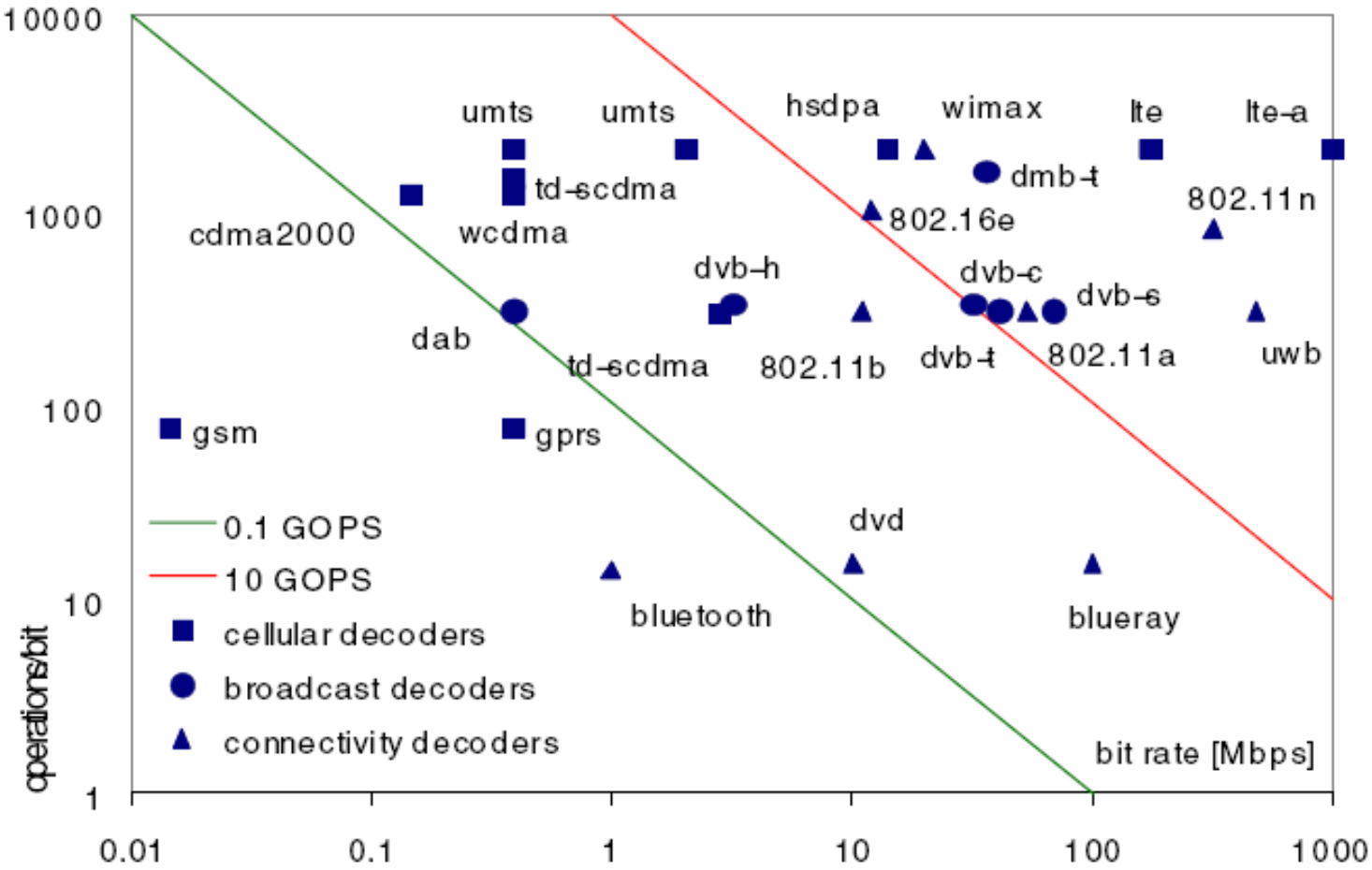


Fig. 1. Battery capacity and power consumption at maximum output power level in cellular transmitters, adapted from [3].

Radio Demodulation Workload



Radio Decoding Workload



3.5G Workload

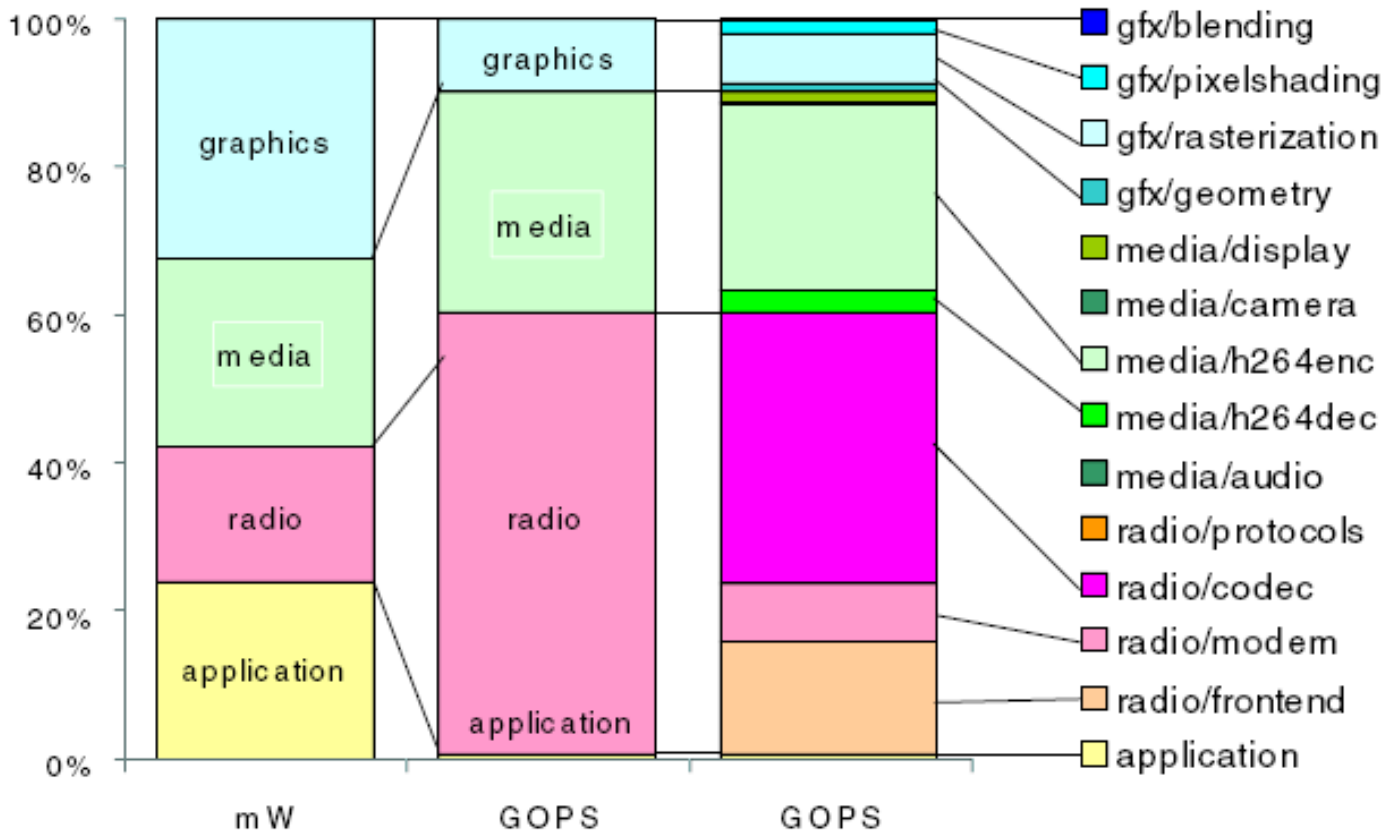


Fig. 3. 3.5G workload (power consumption) as fractions of 100GOPS (1W).

Workload vs Energy/operation

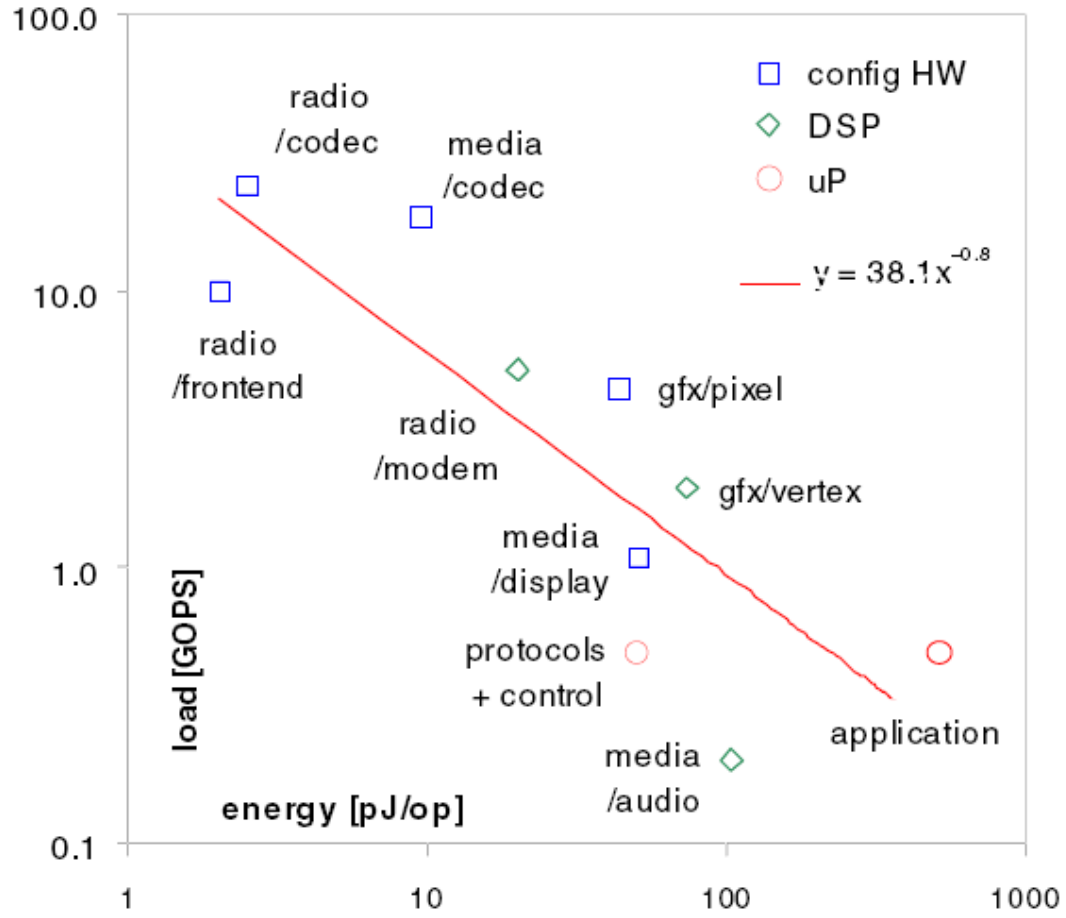


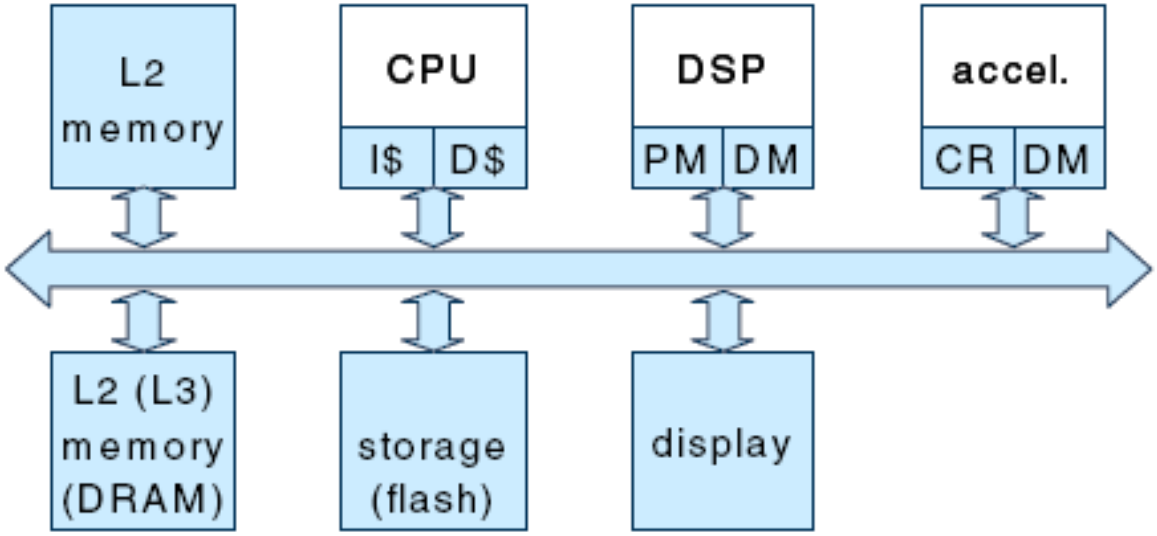
Fig. 4. Workloads [GOPS] versus energy/operation [pJ].

Value of Programming

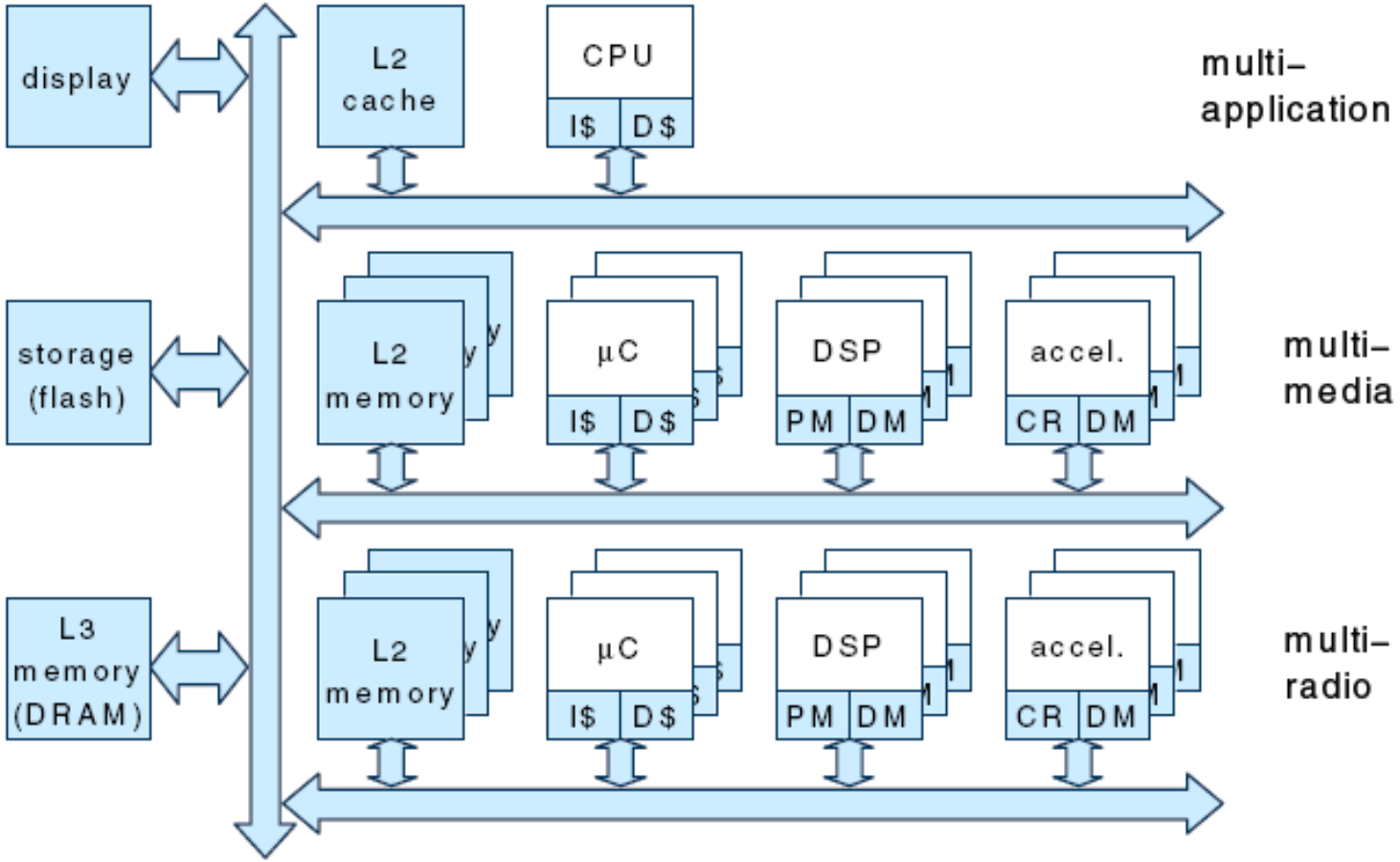
TABLE II
VALUE-AFFORDABILITY OF PROGRAMMABLE SOLUTIONS (EXAMPLES).

	radio	video	3D graphics
very high	protocol stacks		geometry proces.
high	channel estimation		pixel shading
medium	demodulation	motion estimation	
low	turbo decoder (i)fft	entropy (de)coding deblocking	
very low	filters	filters scaling	rasterization pixel blending

2/2.5G Dual-core Architecture



3/3.5G Multi-core Architecture



Cell-phone Chips

TABLE III
PUBLISHED INDUSTRIAL CELL-PHONE CHIPS ("..." DENOTES A SHARED RADIO-MEDIA CORE).

year	ref	source	cmos	total	application		radio		media	
			nm	# cores	core(s)	MHz	core(s)	MHz	core(s)	MHz
1992	[1]	Philips	1000	1	n.a.		KISS-16-V2	...	...	20
2000	[2]	Infineon	250	2	CPU	78	Oak	...	...	78
2001	[9]	Samsung	180	2	ARM9		Teaklite		n.a.	
2003	[6]	Toshiba	130	3	n.a.		n.a.		3x RISC	125
2004	[3]	Nokia	130	2	CPU	50	DSP	...	...	160
2004	[10]	Qualcomm	130	3	ARM9	180	DSP	95	DSP	95
2004	[11]	Renesas	130	2	CPU	216	n.a.		DSP	216
2005	[12]	NEC	130	4	ARM9	200	n.a.		2xARM9 + DSP	200
2006	[13]	ST	130	2	ARM8	156	ST122 DSP	...	...	156
2007	[14]	Infineon	90	2	ARM9	380	TEAKlite	...	...	104
2008	[15]	Renesas et al	65	4	ARM11	500	ARM9	166	ARM11 + SHX2	500
2008	[16]	TI	45	5	ARM11	840	?		C55 + ?	480
2008	[17]	NEC	65	3	ARM11	500	ARM11	250	DSP	500
2009	[18]	Panasonic	45	4	ARM11	486	ARM11	245	2xDSP	216
2009	[19]	Renesas	65	2	CPU	500	n.a.		SHX2	500

Power Management Knobs

TABLE IV

POWER MANAGEMENT KNOBS: f_C DENOTES CLOCK FREQUENCY, V_{DD} AND V_t DENOTE SUPPLY AND THRESHOLD VOLTAGE, AND P_D AND P_S DENOTE DYNAMIC AND STATIC POWER CONSUMPTION.

	knob		throughput	power
stop the clock:	$f_C \downarrow 0$	$\Rightarrow$	$\downarrow 0$	$P_D \downarrow 0$
frequency scaling (FS)	$f_C \downarrow$	$\Rightarrow$	$\downarrow$	$P_D \downarrow$
voltage scaling (VS)	$V_{DD} \downarrow$	$\Rightarrow$	$\downarrow$	$P_D \downarrow$
power down	$V_{DD} \downarrow 0$	$\Rightarrow$	$\downarrow 0$	$P_S \downarrow 0$
forward body bias (FBB)	$V_t \downarrow$	$\Rightarrow$	$\uparrow$	$P_S \uparrow$
reverse body bias (RBB)	$V_t \uparrow$	$\Rightarrow$	$\downarrow$	$P_S \downarrow$