Energy aware memory allocation in real-time systems

CCM-SRAM allocation to reduce consumption

Loïc Thomas

LAAS CNRS

lthomas@laas.fr

July 24, 2023

Current consumption graphic

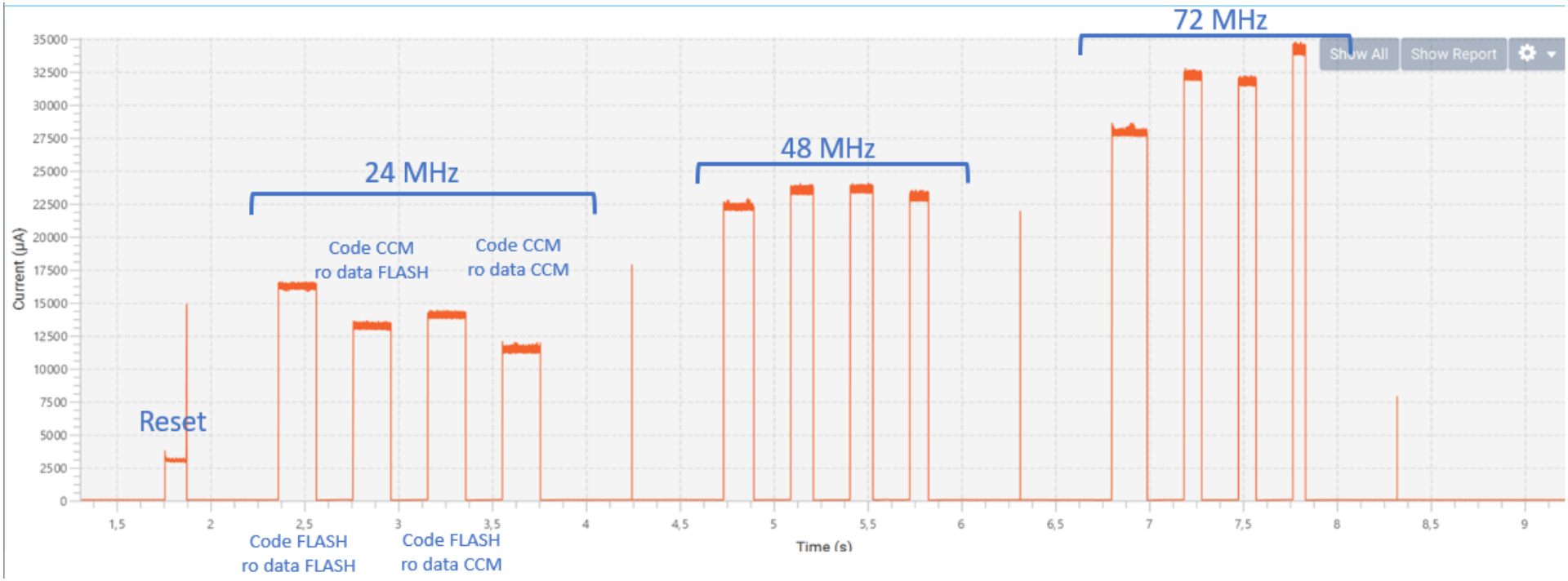
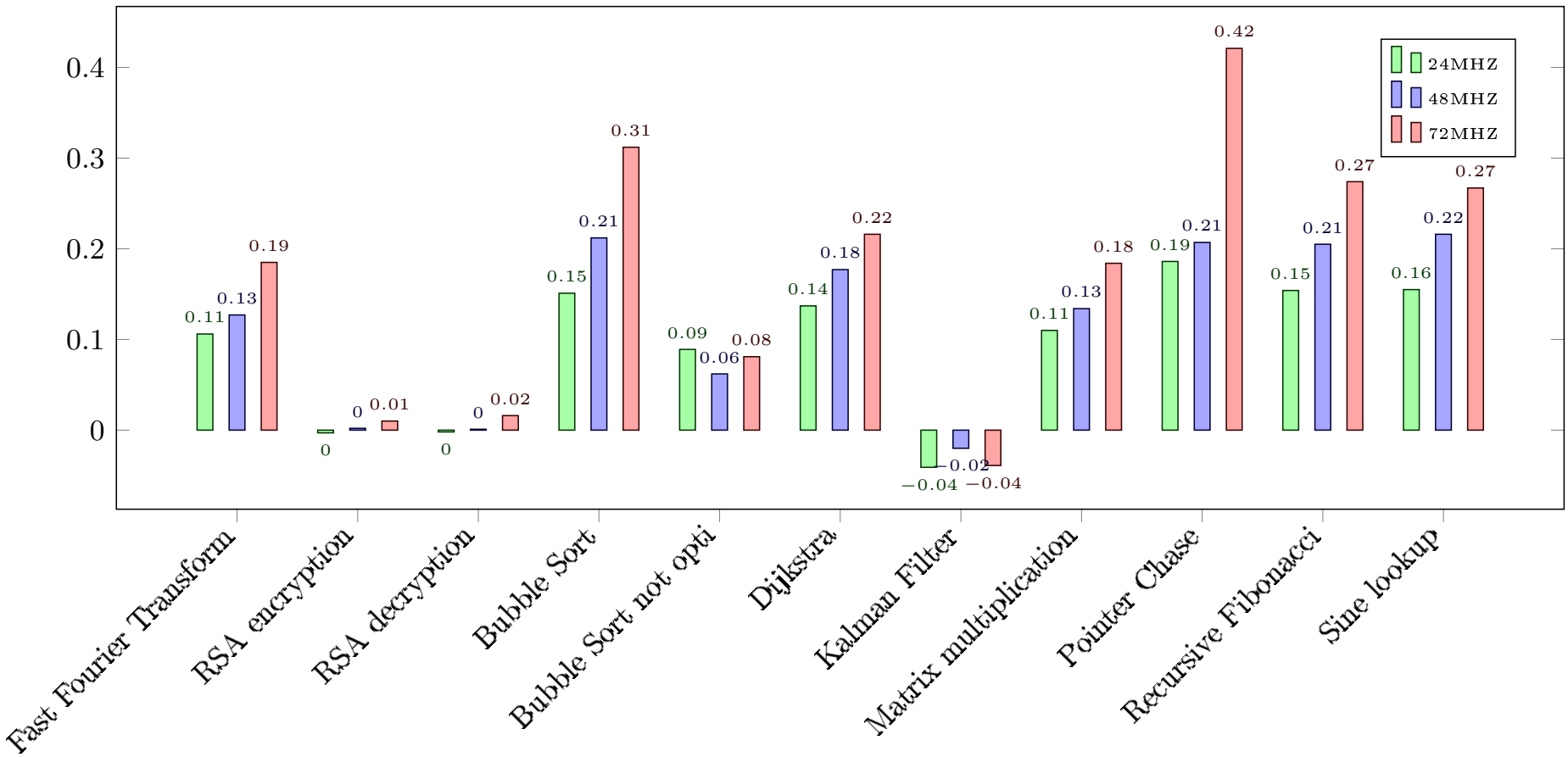


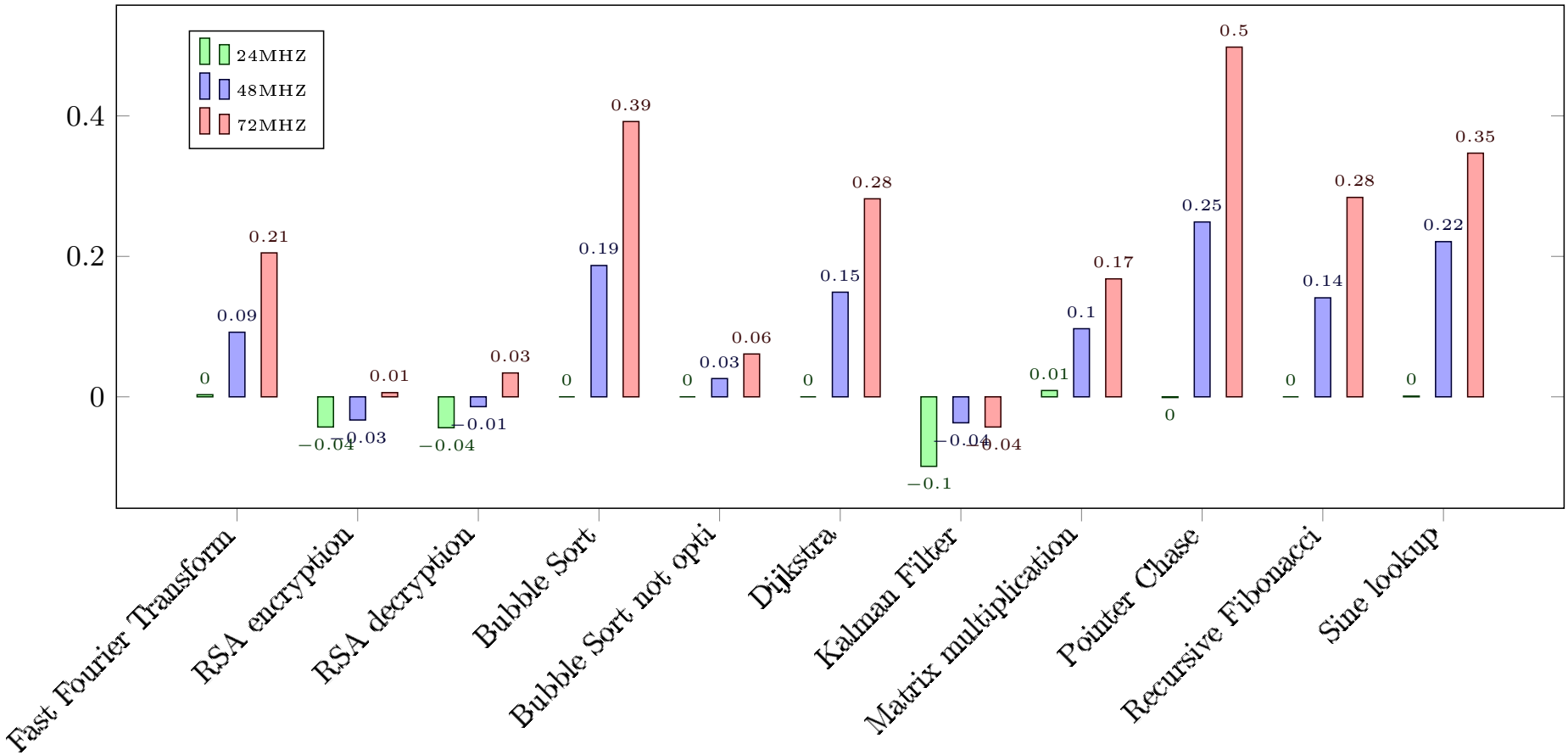
Figure: Intensity consumption graph for differents pointer chase executions

Energy decrease when instruction are moved into CCM



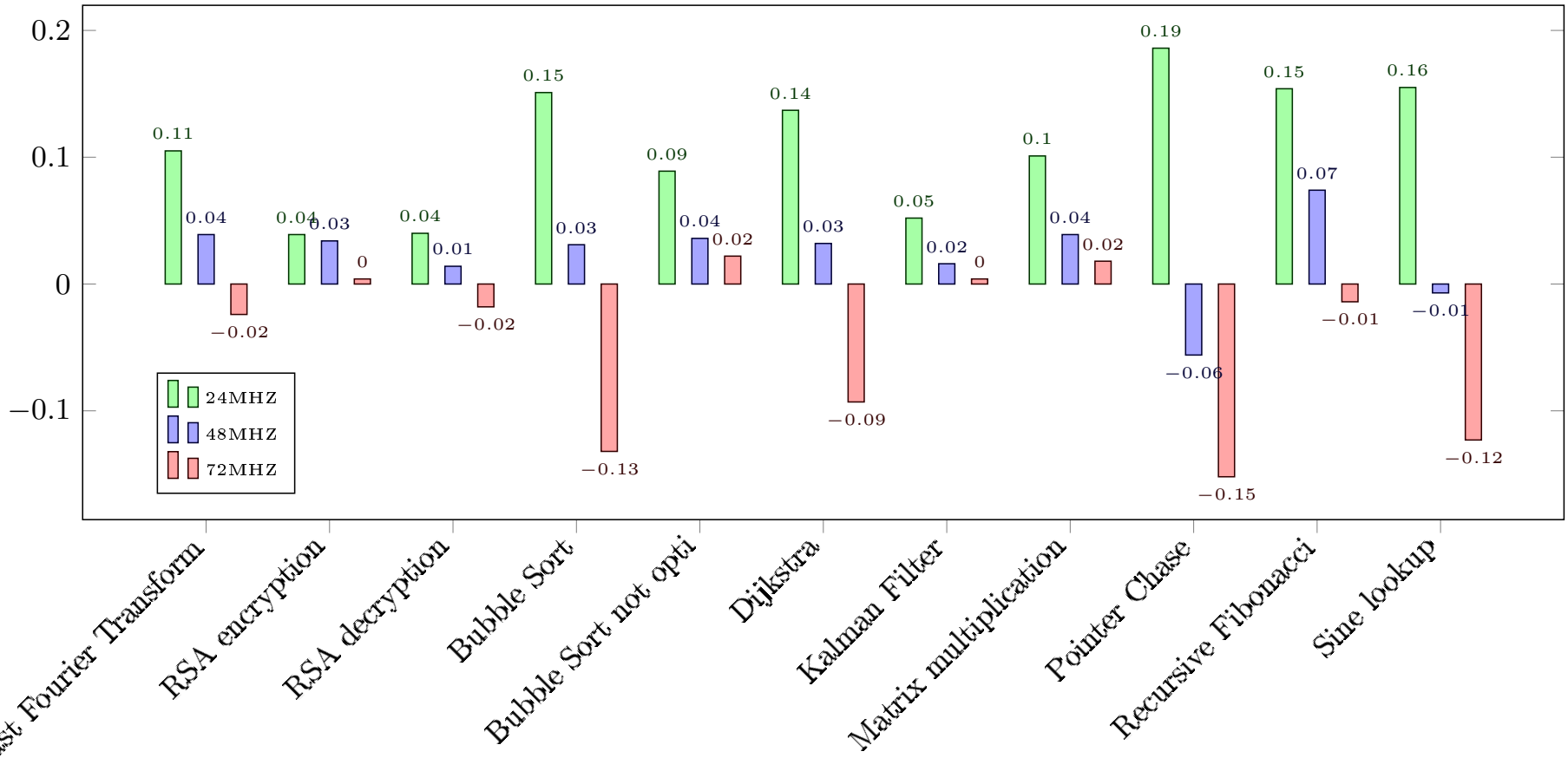
Moving instruction into CCM SRAM can overall improve energy consumption

Runtime decrease when instruction are moved into CCM



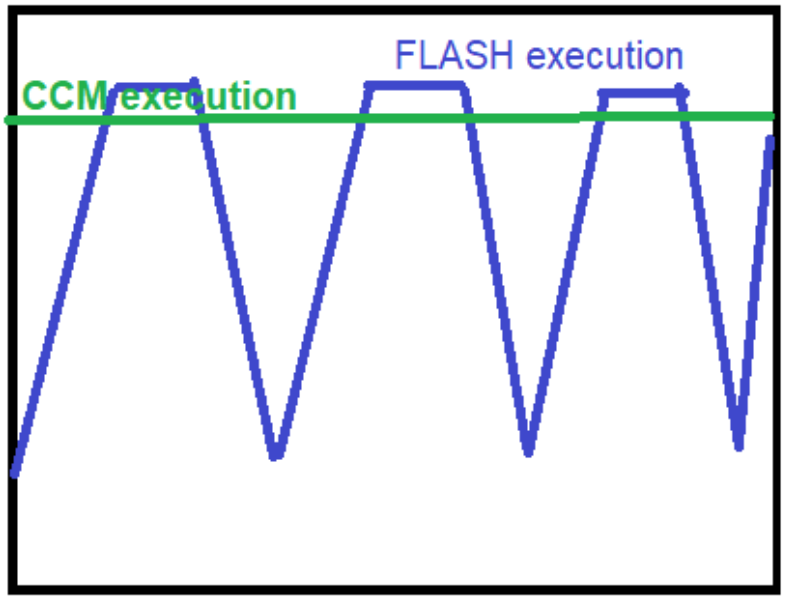
In high frequencies the runtime is reduced when code is moved into CCM SRAM.

Intensiy decrease when instruction are moved into CCM

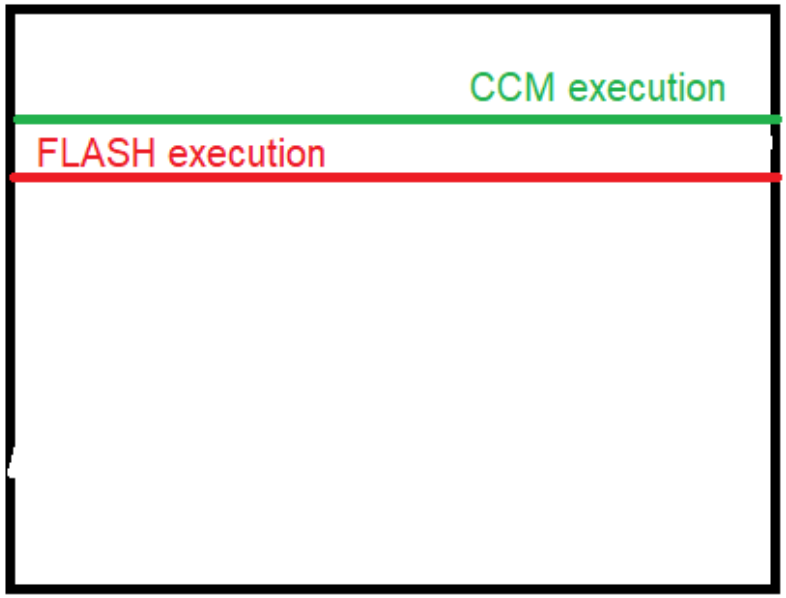


In high frequencies, intensity is lowered when the code is running from flash.

Wait states impact on intensity



What we could observe if we had a sampling frequency of 48 MHz



What we really see

Wait states lower the average intensity of an execution. This effect

Absolute energy when code is running from CCM (mJ)

	24MHz	48MHz	72MHz
FFT	0.262	0.255	0.251
RSA enc	0.668	0.664	0.694
RSA dec	5.239	5.167	5.469
bubble sort	11.294	10.963	10.893
bubble sort no opti	43.018	42.525	41.67
dijkstra	36.319	35.19	34.829
kalman	2.302	2.358	2.665
matrix mul	0.227	0.226	0.222
pointer chase	8.816	9.402	10.026
recursive	5.199	5.036	4.971
sine lookup	14.42	14.62	14.933

Energy remains almost the same even if the frequency increases.

Absolute energy when code is running from FLASH (mJ)

	24MHz	48MHz	72MHz
FFT	0.293	0.292	0.308
RSA enc	0.666	0.665	0.701
RSA dec	5.229	5.171	5.56
bubble sort	13.3	13.908	15.832
bubble sort no opti	47.238	45.318	45.361
dijkstra	42.077	42.742	44.416
kalman	2.211	2.311	2.566
matrix mul	0.255	0.261	0.272
pointer chase	10.826	11.853	17.32
recursive	6.142	6.334	6.848
sine lookup	17.069	18.641	20.366

Due to the wait states there is a waste of energy in high frequencies.

Pists

Comparaison with FLASH

Moving instructions in CCM SRAM reduces runtime and energy consumption. It is almost always better to run code from CCM.

Energy over frequency

- Energy is constant when code runs from CCM at every frequency. There are only advantages to run from CCM at maximum frequency.
- When instructions are in FLASH we can gain energy if the frequency is lower

Offline algorithm

Choose the best tasks to move in CCM and which one we lower the frequency

$$\min. \quad \sum_{i=1}^{n} \sum_{\forall p, \forall d, \forall f} E_i^{p,d,f} \cdot x_i^{p,d,f}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \left(\sum_{\forall d} m_i^P x_i^{\diamond,d} + \sum_{\forall p} m_i^D x_i^{p,\diamond} \right) \leq M^{\diamond}, \ \diamond \in \{F, C\}$$

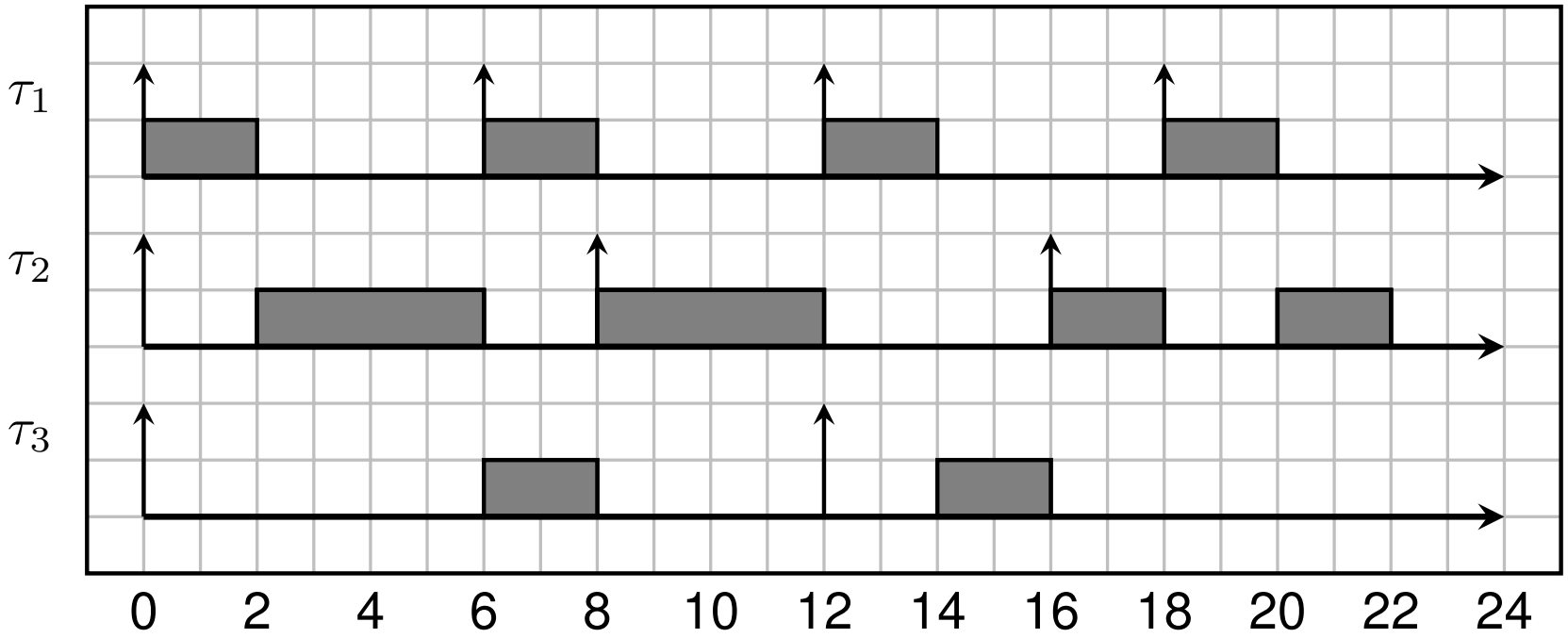
$$\sum_{i=1}^{n} \left(m_i^I + \sum_{\forall p} m_i^D x_i^{p,S} \right) \leq M^S$$

$$\sum_{\forall p, \forall d, \forall f} x_i^{p,d,f} = 1, \quad \forall i$$

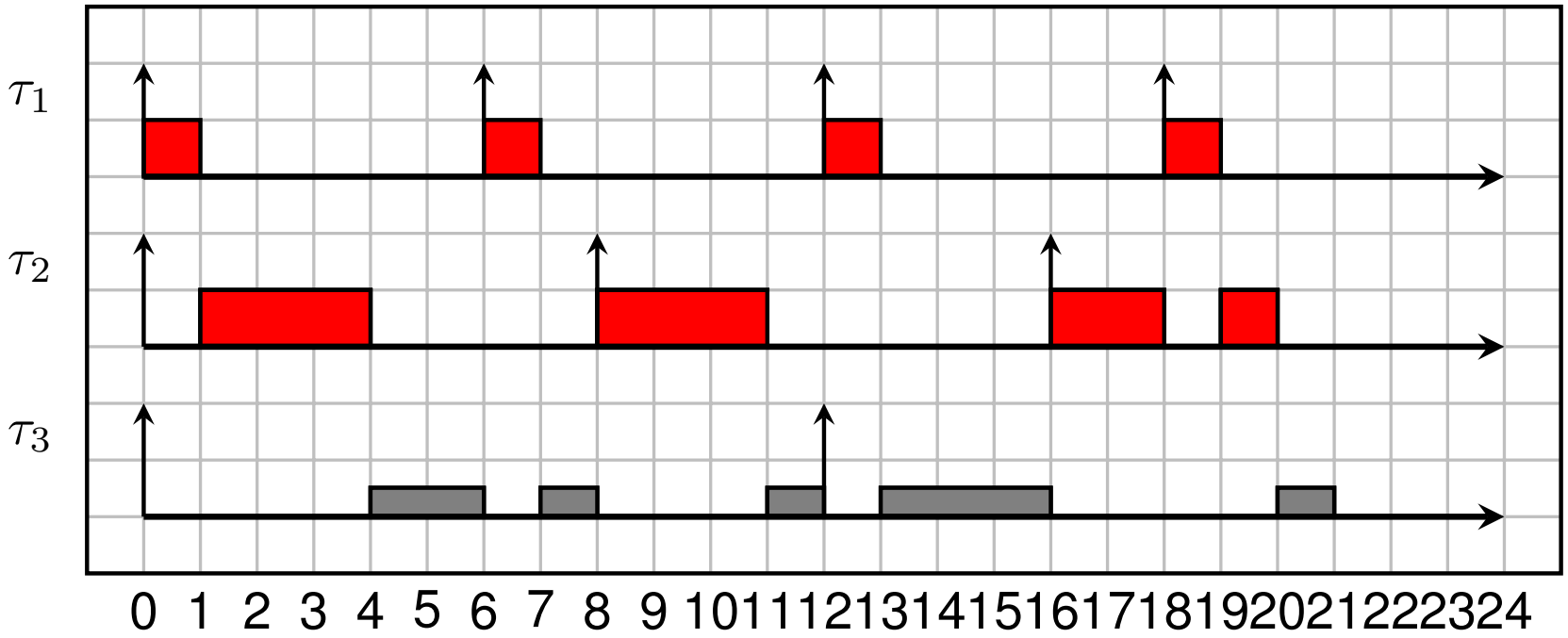
$$\sum_{i=1}^{n} \sum_{\forall p, \forall d, \forall f} U_i^{p,d,f} \cdot x_i^{p,d,f} \leq 1$$

$$x_i^{p,d,f} \in \{0, 1\}, \quad \forall i, p, d, \forall f \in [8, 72]$$

Normal EDF example



Application of the previous algorithm



Online algorithms

Dynamic frequency scaling for FLASH

If we can still complete the task with the worst case time execution before the deadline. Then we lower the frequency.

Dynamic CCM SRAM allocation

- Divide the CCM into slots
- Before executing copy task instructions in the CCM slot
- the worst case execution time is equal to the CCM execution time with the associated memcopy

$\Rightarrow$ Changes in the scheduling algorithm: mutex, blocking, non-preemptive tasks . . .

Different execution modes

- Range 1 mode (normal) : normal execution from 8 to 150 MHz.
- Range 1 boost mode : frequency up to 170 MHz, less wait states but higher voltage .
- Range 2 low power mode : frequency between 8 and 32 MHz, better consumption in these frequencies.

FLASH cache

Allows zero wait states and pre fetch when instructions are in FLASH $\rightarrow$ same results as the CCM.

The End

Questions? Comments?