Sharif University of Technology
Department of Computer Science and Engineering

Lec. 5:
Power, Energy, and Reliability

M. Ansari

Fall 2023

According to Peter Marwedel's Lectures

Safety-Critical Embedded Systems

- Safety requirements cannot come in as an afterthought.
- Safety requirements have to be considered right from the beginning.
- For safety-critical systems, the system as a whole must be more dependable than any of its parts.

Safety-Critical Embedded Systems

- Allowed failure may be in the order of 1 failure per 10^9 hours.
  - 1000 times less than typical failure rates of chips.
- Unfortunately, safety-critical embedded systems are not testable. Instead, safety must be shown by a combination of testing and reasoning.

Kopetz Design Principles

1. Safety requirements have to be considered in the entire design process.

2. Precise specifications of design hypotheses must be made right at the beginning.
   - These include expected failures and their probability.

Kopetz Design Principles

3. Fault containment regions (FCRs) must be considered. Faults in one FCR should not affect other FCRs.

4. Differentiate between original and follow-up errors.
   - Timing, cause and effect

Kopetz Design Principles

5. Well-defined interfaces have to hide the internals of components.

6. It must be ensured that components fail independently.

7. Components should consider themselves to be correct unless two or more other components pretend the contrary to be true.
   - Principle of self-confidence.

Kopetz Design Principles

8. Fault tolerance mechanisms should be decoupled from the regular function.

9. The system must be designed for diagnosis. For example, it has to be possible to identifying existing (but masked) errors.

Kopetz Design Principles

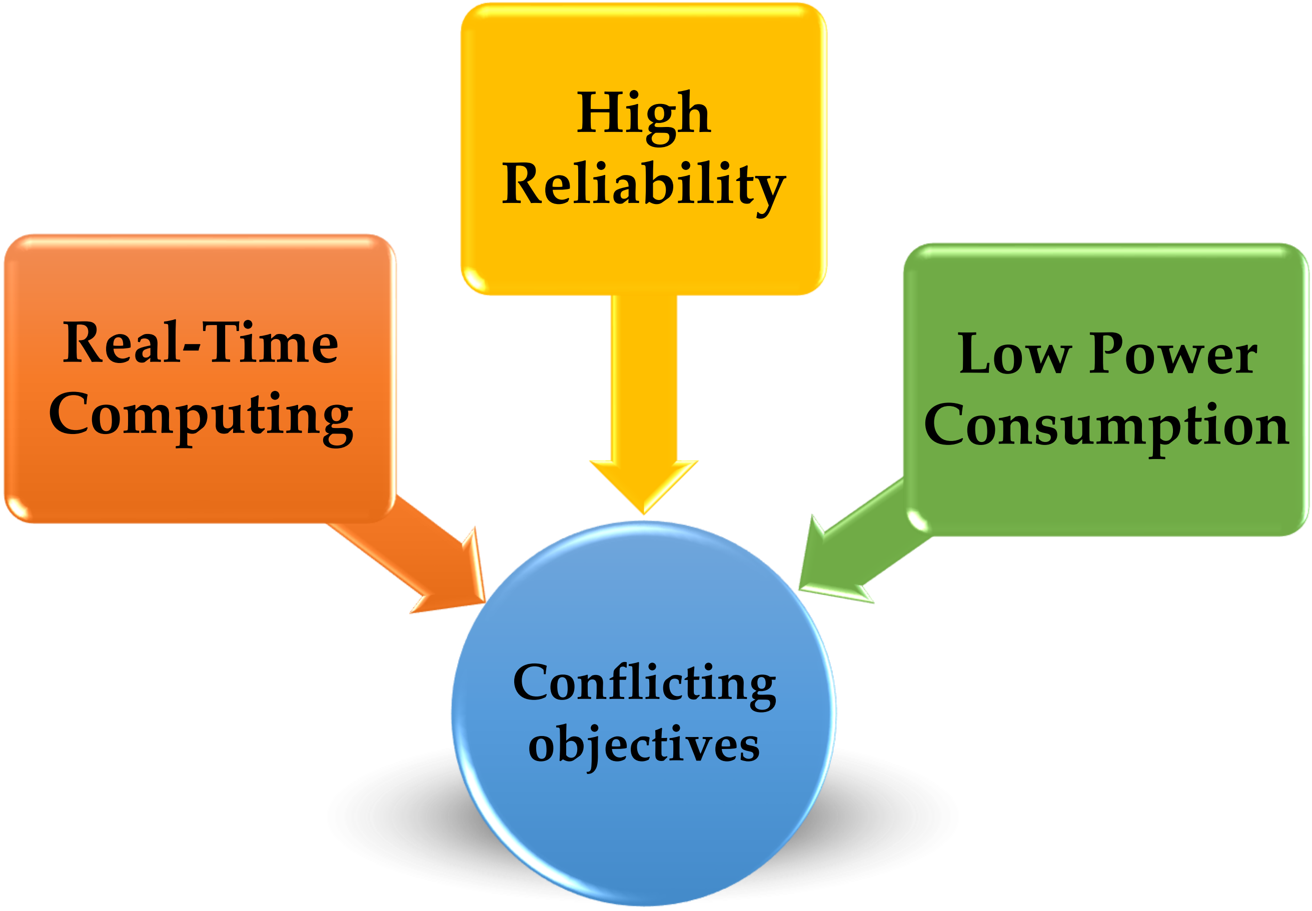
10. The man-machine interface must be intuitive and forgiving.
    - Safety should be maintained despite mistakes made by humans.

11. Every anomaly should be recorded.
    - These anomalies may be unobservable at the regular interface level.
    - This recording should involve internal effects, since otherwise they may be masked by fault-tolerance mechanisms.

Kopetz Design Principles

12. Provide a never-give up strategy.
    - Embedded systems may have to provide uninterrupted service.
    - The generation of pop-up windows or going offline is unacceptable.

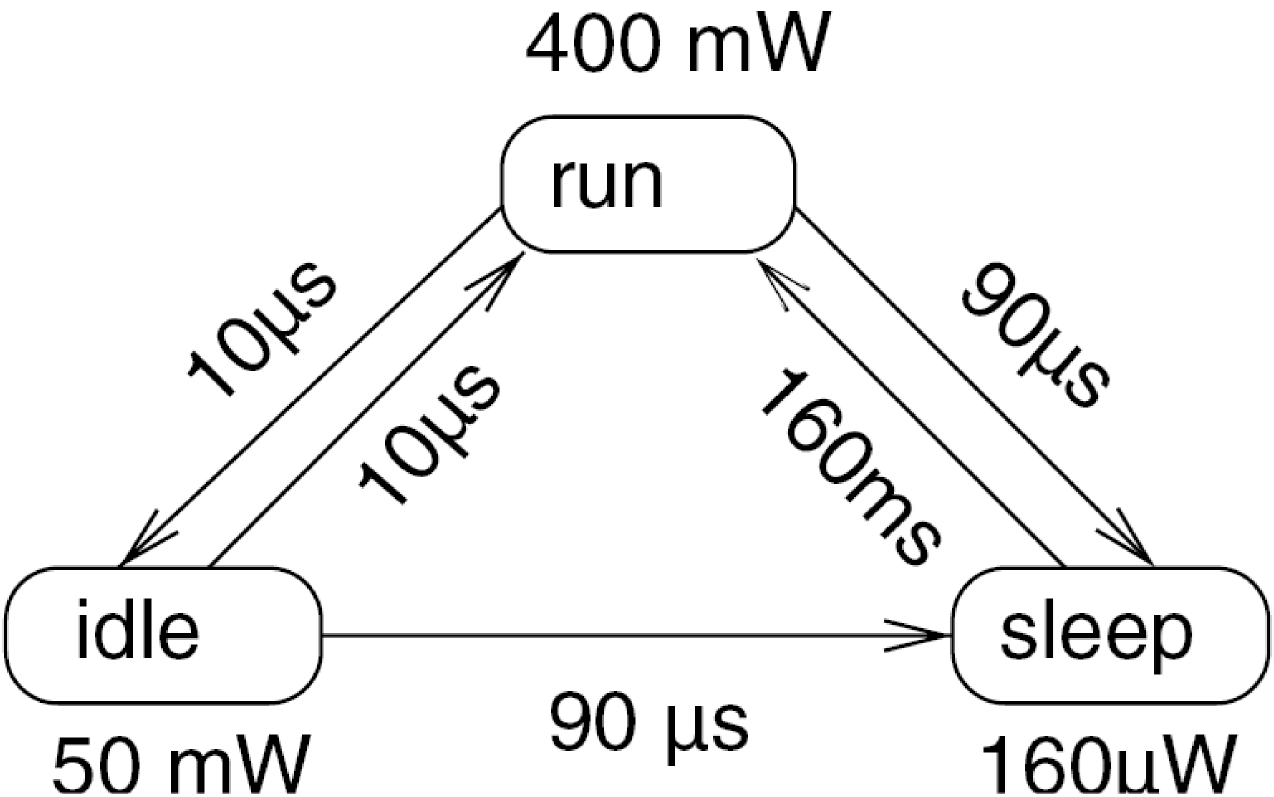
Safety-Critical Embedded System Requirements



Dynamic Power Management

- Main Idea: the shutdown of idle system components.
- An advantage of DPM is its generality, which allows its usage not only for digital circuitry, but also for other system components such as displays, and hard drives.

Example: StrongArm SA 1100



Example: StrongArm SA 1100

- The processor is fully operational in the run state.

- In the idle state, it is just monitoring the interrupt inputs.
  - reactive nature of ES

- In the sleep state, all on-chip activity is shutdown.

DVS-Enabled Processors

- DVS-enabled processors have the ability to dynamically change their supply voltage and operational frequency settings during run-time of the application.

$$P_{SW} = \alpha C_L V_{DD}^2 f \qquad Delay \propto \frac{C_L \cdot V_{dd}}{(V_{dd} - V_{th})^{\alpha}}$$

Example 1: Crusoe Processor

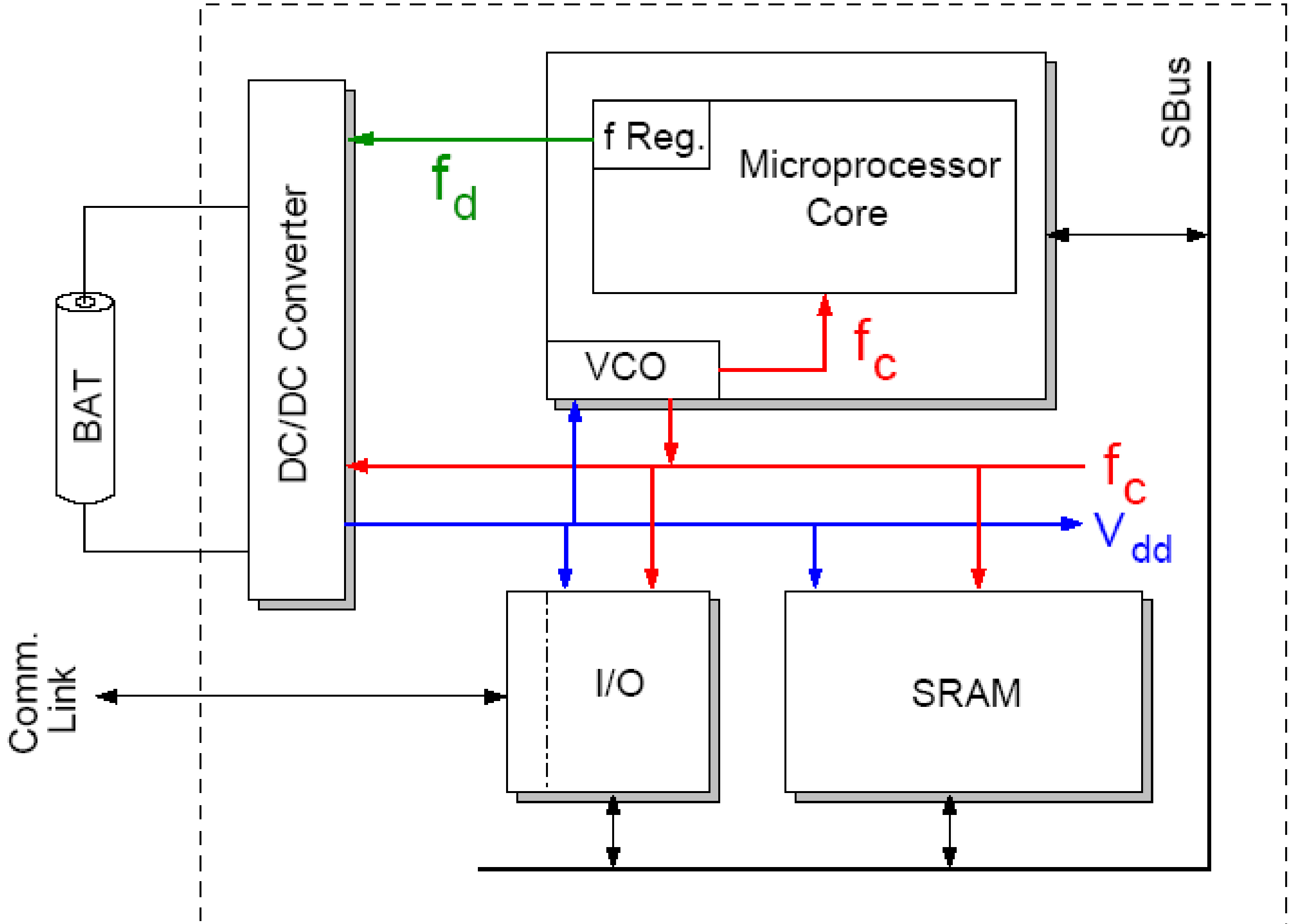
- By Transmeta
- 32 voltage levels between 1.1 and 1.6 volts
- Clock can be varied between 200 MHz and 700 MHz in increments of 33 MHz
- Transitions from one voltage/frequency pair to the next takes about 20 ms.

Example 2: Mobile Pentium III

- Two different speed/voltage pairs are provided
- Intel SpeedStep Technology

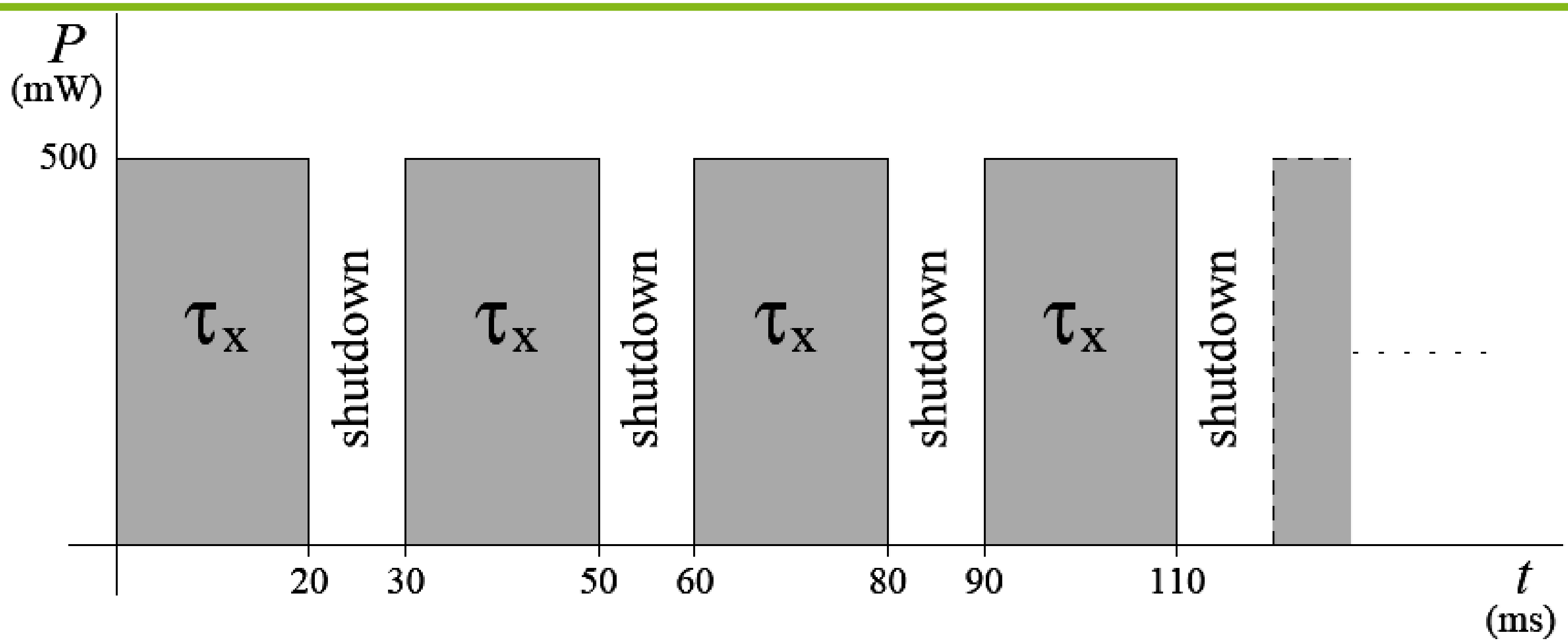
DVS-Enabled Processors

$$f \propto V_{DD}$$



VCO: voltage controlled oscillator
Note that the frequency register is under software control.

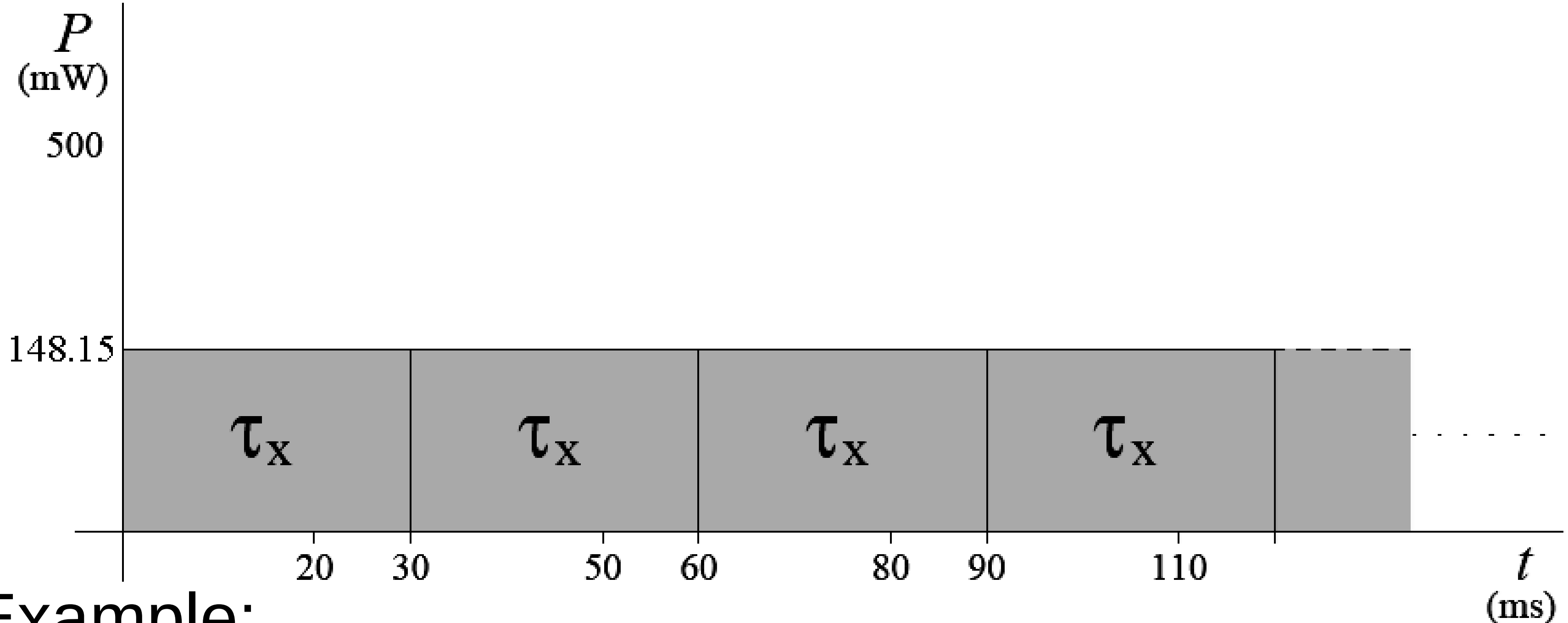
DVS vs. DPM



Example:

- DPM
  - Execution time= 20 ms
  - *F*= 33MHz, *Vdd*= 3.3V
  - Power dissipation = 500 mW
  - P_{AV}=(2/3)*500 mW= 333.333 mW
  - Period= 30 ms

DVS vs. DPM



Example:

- DVS
  - Execution time= 20 ms -> 30 ms
  - *F*= 33MHz -> 22MHz, *Vdd*= 3.3V -> 2.2V
  - Power dissipation = 500 mW -> $(2/3)^3$*500 mW =148.15mW = P_{AV}
  - Period= 30 ms

Single Event Upsets (SEU)

- Bit-flips due to the impact of particles on flip-flops.

$$\lambda_{SEU} \propto \exp(-Q_{CRIT})$$

$$Q_{CRIT} = V_{DD}.C_L$$

- **DVS has a negative impact on SEU rate**

System Level Low Energy Design

- DVS (Dynamic Voltage Scaling):
  - lowering V_{dd} and F
- ABB (Adaptive Body Biasing):
  - lowering *Vbs*
- DPM (Dynamic Power Management):
  - turning off the unused components

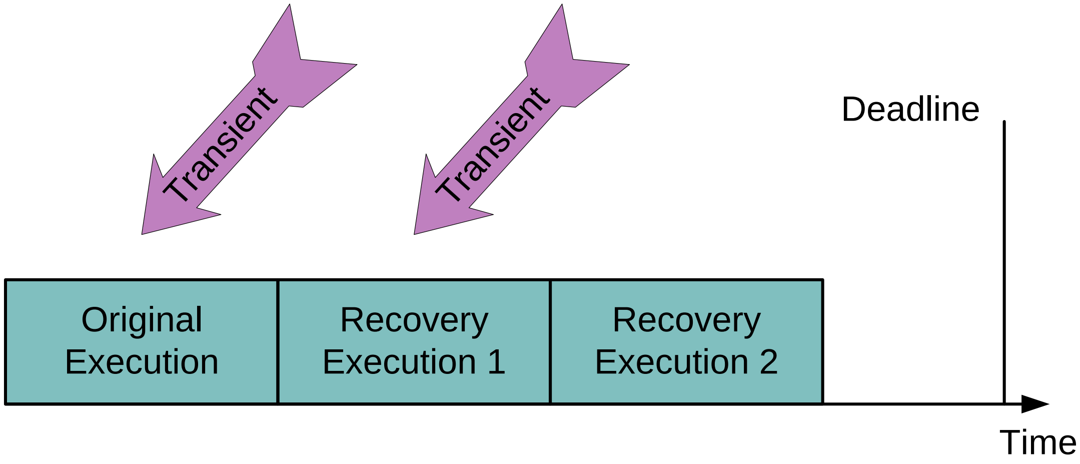
Energy Consumption

$$E_{cyc} = \underbrace{C_{eff} V_{dd}{}^{2}}_{Dynamic \quad Energy} +$$

$$\underbrace{\frac{L_g}{f(V_{dd})} \left(V_{dd} K_3 e^{K_4 V_{dd}} e^{K_5 V_{bs}} + |V_{bs}| I_j \right)}_{Static \quad Energy}$$

Reliability Problems in DVS-Enabled Systems

- Increased fault rate and error rate
  - Reduced Noise Margin
  - Reduced Critical Charge
    - More susceptible to SEUs
- Increased execution time $\rho = 1 - e^{-\lambda \cdot T_{exe}}$
- Reduced slack time in real time systems



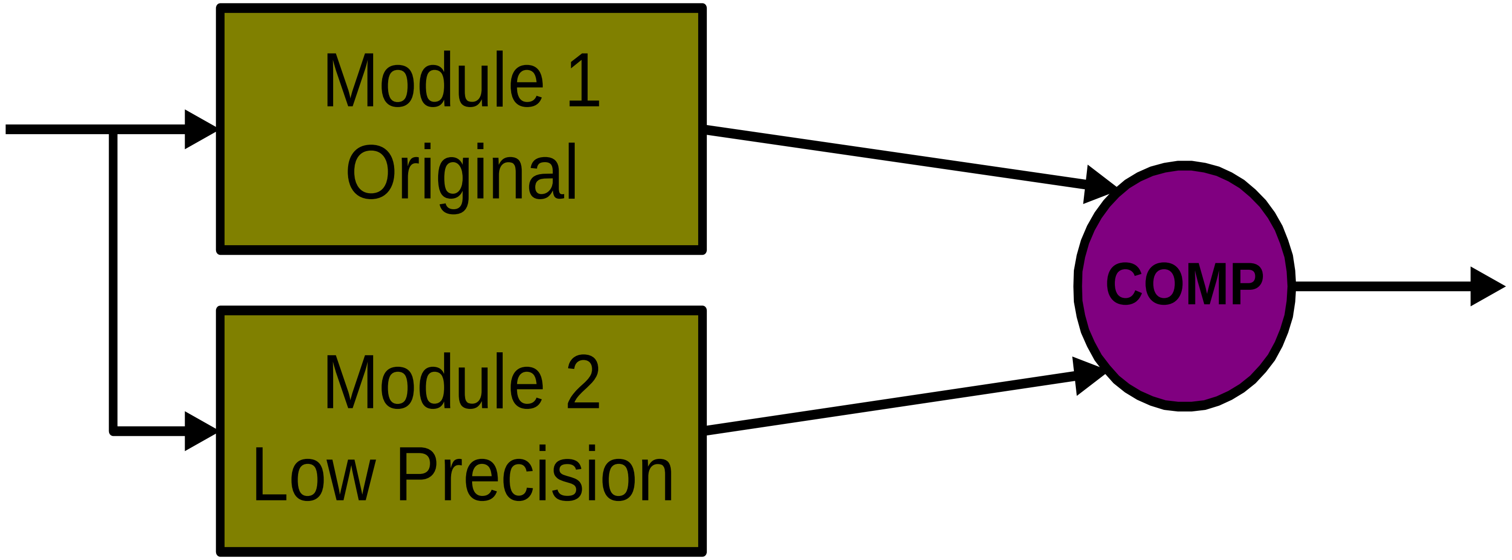
Energy Problem in FT Systems

- Fault Handling requires redundancy.

- Redundancy is simply the addition of
  - Time
  - Hardware
  - Software
  - Information

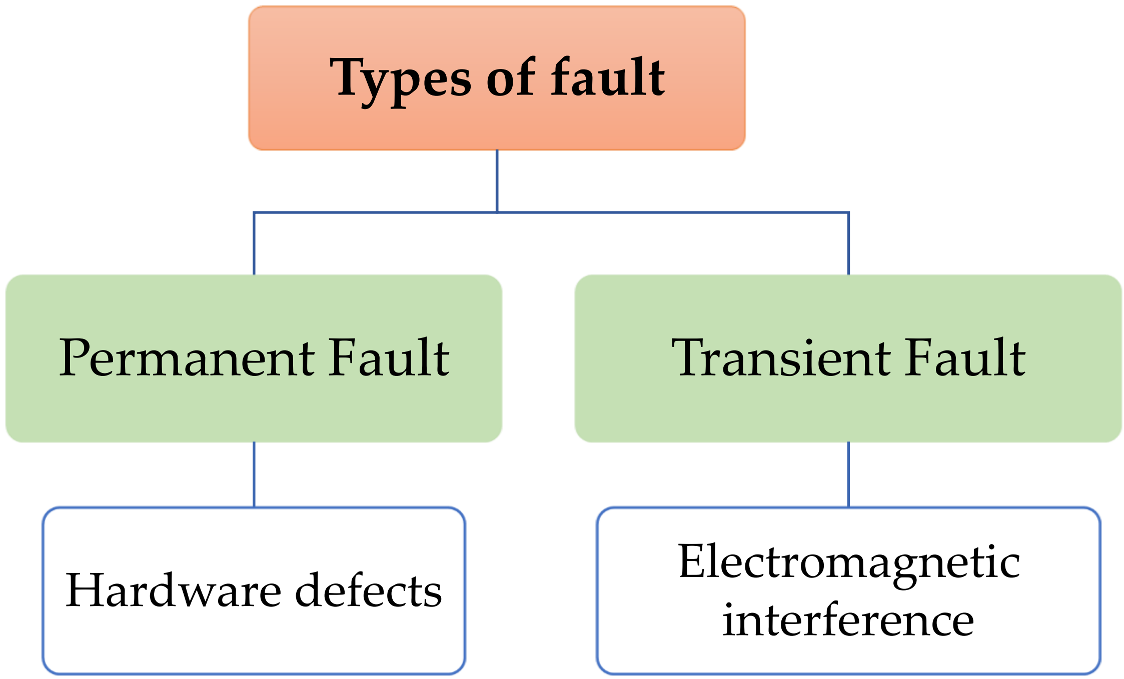
  beyond what is needed for normal system operation.

Example 1: Low Precision Redundant Units

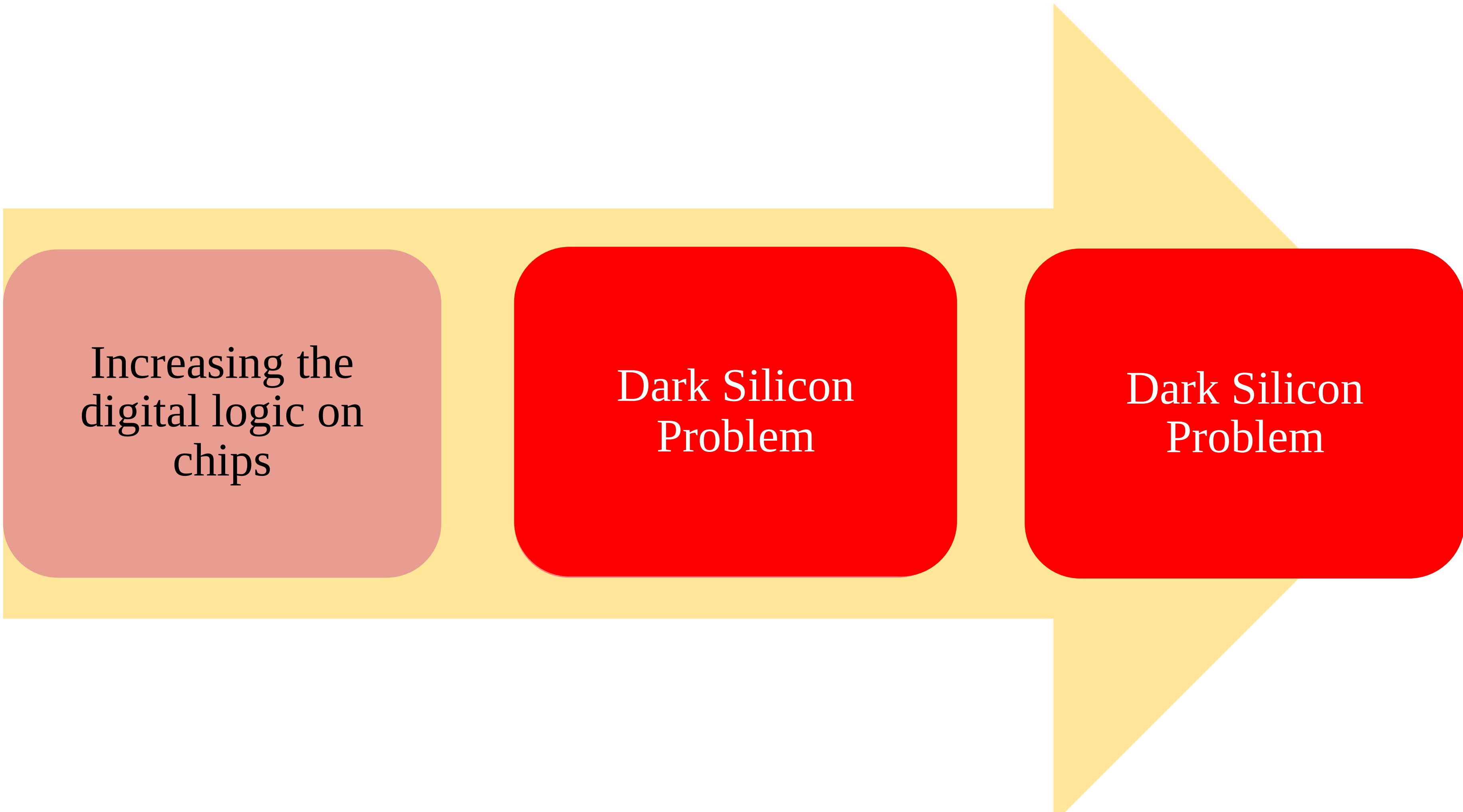


Fault Tolerance

- Fault-tolerance techniques → High reliability
- Example:
  - Common approach to deal with the permanent faults → Hardware redundancy → Increasing the average/peak power



Dark Silicon Problem



Dark Silicon Problem (2)

- Dark Silicon Problem (core-level):
  - A percentage of the total available cores in a many core system cannot be powered-on simultaneously due to thermal constraints.
- Thermal Design Power (TDP)
  - Highest sustainable power
  - Solutions:
    - Chip’s cooling

Challenges

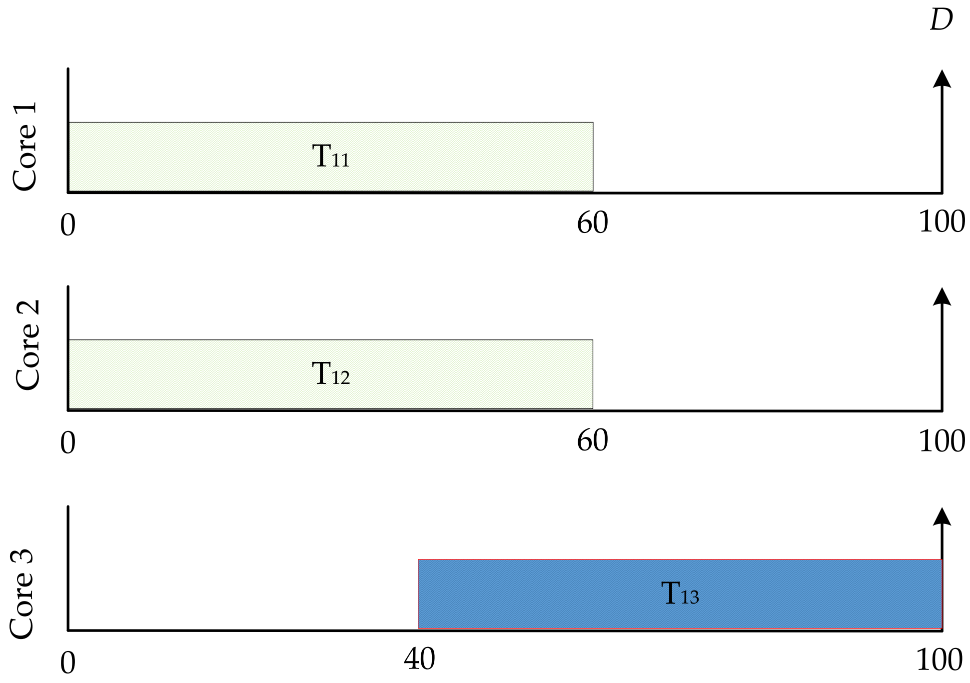
- Power Management
  - Thermal Design Point (TDP) → Reliability degradation and violation of timing constraints
- Reliability Management
  - Fault-tolerance techniques → Extra power consumption
- Real-Time Constraints
  - Power management and fault-tolerance techniques → Violation of timing constraints

Optimistic TMR

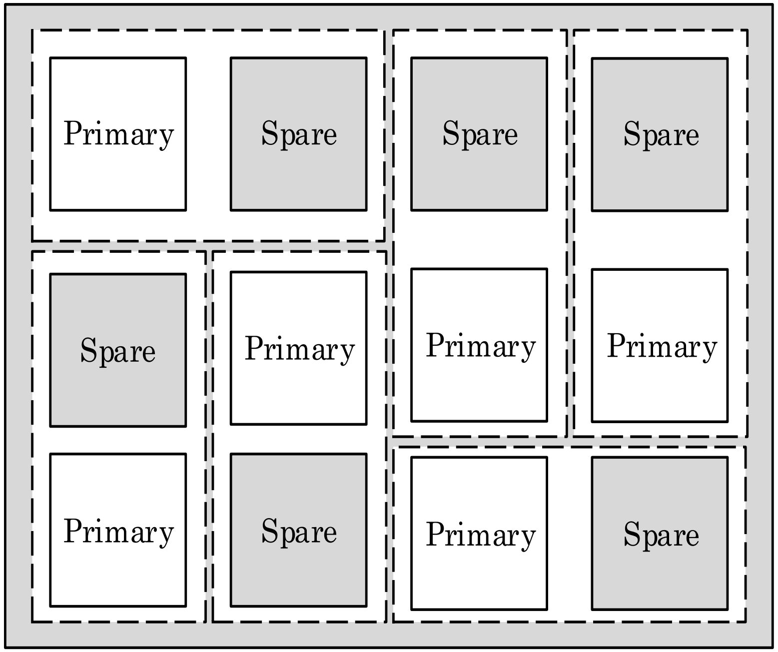
- A TMR system tolerates one fault by running an application on three identical processing units simultaneously and voting on the three outputs.
- An Optimistic TMR (OTMR) scheme has been proposed to reduce the energy consumption in a TMR system.
- The idea is to turn off or slow down one processing unit, provided that it can catch up and finish the computation before the deadline if the other two units do encounter an error.

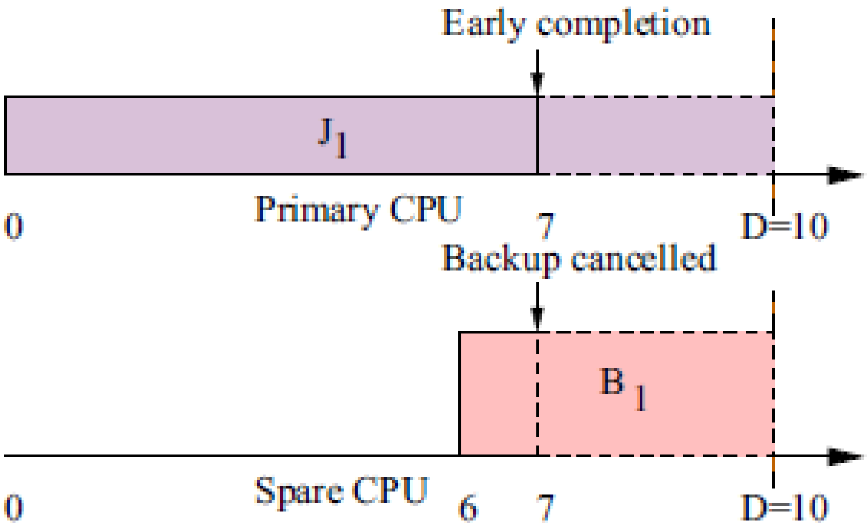
Optimistic TMR (Cont.)

- Schedules two copies of each tasks based on EDF
- Schedules the third copy of each tasks based on EDL



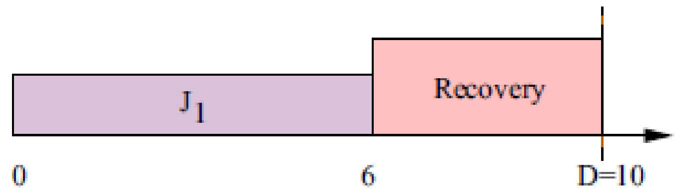
LESS: Low Energy Standby-Sparing



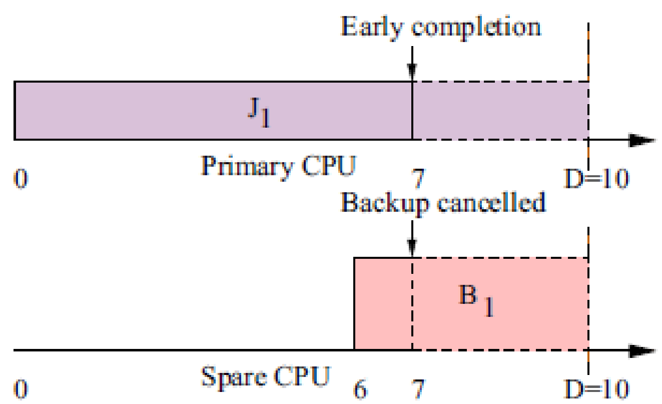


32

RAPM vs LESS



RAPM for a single task



Standby-sparing system for a single task

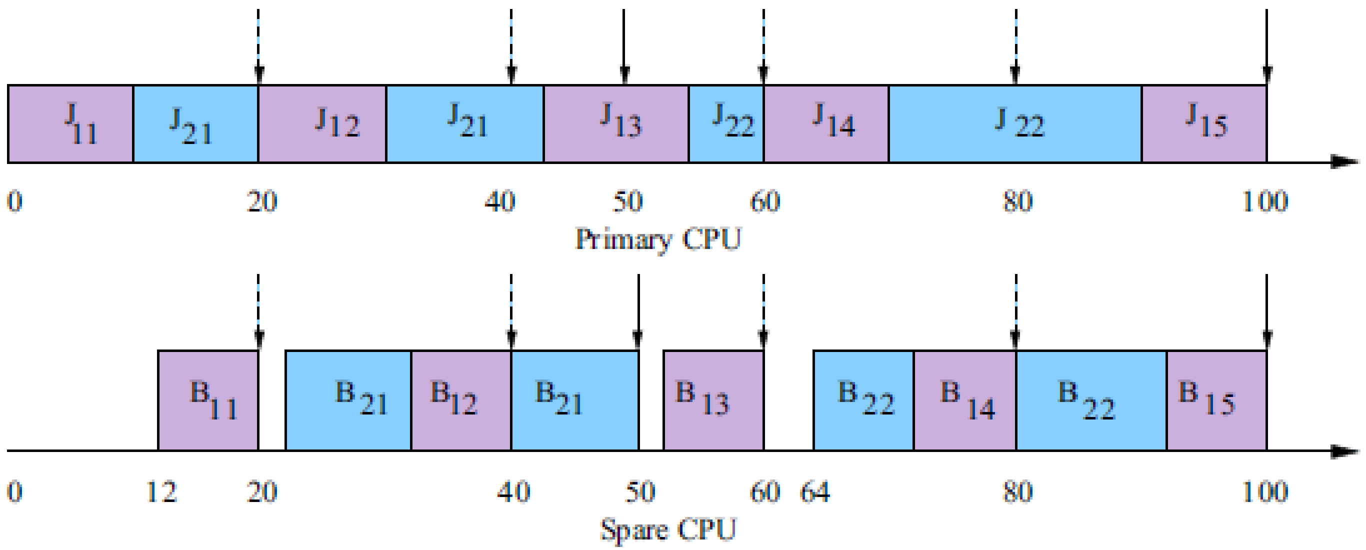
LESS [Ejlali09]

- Co-management of energy and reliability
- Hardware redundancy with DVFS
- Preserving the original reliability
- Primary processor → DVFS technique
- **Some important limitations**
  - Non-preemptive and aperiodic jobs

LESS [Haq11]

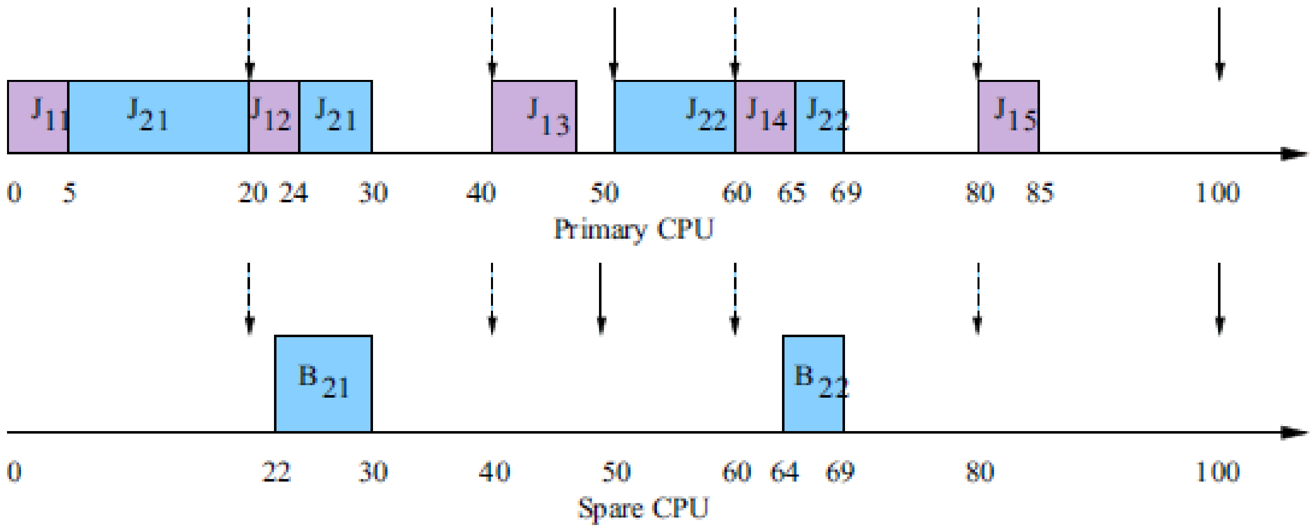
- **“Earliest Deadline First” on Primary Processor**
  - Executing a job with the earliest deadline
- **“Earliest Deadline Late” on Spare Processor**
  - Delaying the backup tasks
  - Obtaining idle intervals as early as possible on the schedule

LESS [Haq11] (Cont.)



Standby-sparing system for periodic tasks

LESS [Haq11] (Cont.)



Taking advantage of early completions (fault-free execution)

LESS [Haq13]

- An energy-management technique
  - Executing preemptive fixed-priority real-time tasks
- **"Rate-Monotonic-Scheduling" on Primary Processor**
  - Executing a job with the least period
- **"DPM and dual-queue mechanism" on Spare Processor**
  - Maximally delaying the backup tasks

Summary

- Safety-Critical Embedded Systems
  - Reliability
  - Power
  - Energy
- DPM and DVFS
- Fault Tolerance
- Dark Silicon Problem
  - Thermal Design Power