Real-Time Embedded DSP Systems

Goals:

- ✓ To know why the recent DSPs are like what we will see
- ❑ To know DSP environment and constraints
- ❑ To know how to choose DSPs
- ❑ To know when to go for HW, when to part the algorithm between HW and SW

DSP applications → Embedded Real time systems

Real-time systems maintain a continuous timely interaction with the environment (RTC)

Hard Real-time / Soft Real-time Systems

Violating the constraints cause failure / performance degradation

DSP Systems are usually Hard Real-time → Feasibility? and Cost?

- ❑ *Understanding the tasks and execution environment characteristics*

→ *bounds and constraints*

- ❑ *Prediction of tasks and execution environment behavior*

→ *corner/worst cases*

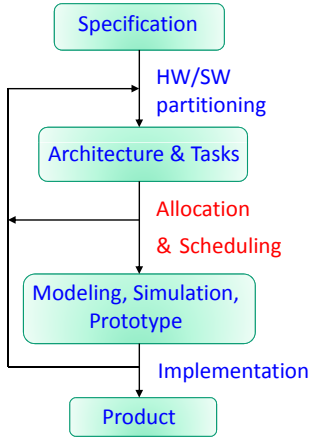
Real-Time Embedded DSP Systems…

Tasks Characteristics:

- *Timeliness parameters, e.g. arrival times, events rates, …*
- *Deadlines*
- *Resource utilization profiles*
- *Relative importance*
- *Worst case execution time*
- *Ready and suspension times*
- *Precedence and exclusion constraints*

Execution Environment Characteristics:

- *System loading*
- *Service latencies*
- *Resources and their interactions*
- *Interrupt priorities and timing*
- *Queuing disciplines*
- *Arbitration mechanisms*



Real-Time Embedded DSP Systems…

Real-time versus Time-shared systems

Time-shared use multi-programming or multi-tasking to maximize throughput

Real-time systems designed for providing predictably fast response to urgent tasks

→ Might have some other non-real-time tasks

Differences→ Real-time Systems need:

- ➢ *High degree of schedulability*

Timing requirements of the system must be satisfied at high degrees of resource usage

- ➢ *Worst-case latency*

Ensuring the system still operates under worst-case response time to events

- ➢ *Stability under transient overload*

When the system is overloaded by events and it is impossible to meet all deadlines, the deadlines of selected critical tasks must still be guaranteed

Real-Time Embedded DSP Systems…

Real-time versus Time-shared systems…

Characteristic	Time-shared systems	Real-time systems
System capacity	High throughput	Schedulability and the ability of system tasks to meet all deadlines
Responsiveness	Fast average response time	Ensured worst-case latency, which is the worst-case response time to events
Overload	Fairness to all	Stability – When the system is overloaded, important tasks must meet deadlines while others may be starved

Real-time Multiple Tasks Categories

1) *Synchronous → Sharply Predictable → tasks can be packed in one*
2) *Asynchronous → Unpredictable*
3) *Isochronous → Loosely Predictable (in a time window)*

S: Camcorder Audio and Video,

A: Calls from phones received by a BTS,

I: Audio and Video in Video over IP systems, when video arrives so will relative audio in a time window

Real-Time Embedded DSP Systems…

Scheduling and Resource Allocation to meet all the deadlines

1) *Offline Algorithms: By the designer*

2) *Online Algorithms: By the OS or other software*

2-1) Static (fixed priorities): e.g. RMA, Rate Monotonic Algorithms → Higher rates first

2-2) Dynamic (changeable priorities): e.g. EDF, Earliest Deadline First

Static priority assignment is suitable for deterministic tasks

→ Truly hard real-time system is not feasible unless for deterministic tasks

In DSP systems using complicated RTOSes are avoided when possible!

Example: DSP/BIOS from TI→ Works for all TI DSPs

Pre-emptive static priority RTOS→ 15 Task Priority, HWI, SWI

On top of this → Use the processor recourses as much as possible

Real-Time Embedded DSP Systems…

"Embedded" Real-time System Requirements

1) Efficiency

Performance ←→ Cost

→ Processor Cycle, Power, Size, Memory

Selection, Optimization

2) Acceptable Timeliness

→ Thru Resource Management

Complexity of resource management ←→ Cost

Expensive → over-capacity processor, complicated OS (online/real-time resource management)

The resource management that is used is usually static and requires analysis of the system prior to executing it in its environment.

Real-Time Embedded DSP Systems…

Selection/Design Guide

1. *Response Time → Optimal Partitioning into HW and SW*

   1) Is the Architecture suitable?

   2) Are the programmable processing resources powerful enough?

   High utilization > 90% makes the system unpredictable,

   more time needed to develop

   3) Are the communication speeds adequate?

   4) Are enough communication ports/IOs available?

   5) Is the right scheduling method available/possible?

2. *Failure Recovery → no reset button/ or hard to access, not possible to check all cases*

   Internal or external Failures : Processor , board, link/connectivity failure, invalid environmental behavior, … ← Simulation can help

Real-Time Embedded DSP Systems…

Selection/Design Guide…

3. *Scheduling Loss*

Liu & Layland (1973)

For 'n' periodic tasks with fixed periods, a feasible schedule that will always meet deadlines exists if the CPU utilization is below a specific bound (depending on the number of tasks).

$$U = \sum_{k=1}^{n} \frac{C_k}{T_k} \le n(2^{1/n} - 1)$$

n	U_{max}
1	100%
2	83%
∞	69%

C_k: Worst case computation time of task k,

T_k: Release Period of task k, n: number of scheduled tasks

The other 30.7% of the CPU can be dedicated to lower-priority non real-time tasks.

Rule of Thumb → 90% needs twice time to develop

→ 95% triple time to develop

Real-Time Embedded DSP Systems…

Selection/Design Guide…

4. *Distributed and Multi-Processor Architecture*

   → *Several nodes: DSP+ μP + FPGA*

   Common practice in today designs

   → *Must consider the following points to decide:*

   4-1) Initialization

   4-2) Processors Interfaces

   4-3) Load distribution and timeliness

   4-4) Managing shared resources

Real-Time Embedded DSP Systems…

Characteristics of Embedded Systems

Application Specific

Monitoring and reacting to the environment

Processing the Information

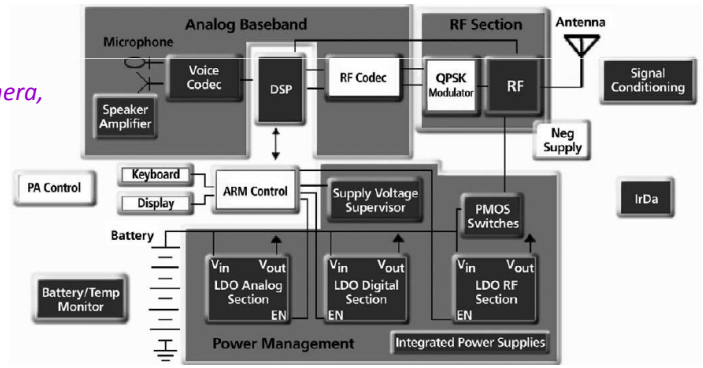
Control the environment

Examples:

Airbag system,

Digital Still Camera,

Cell Phone, …



Real-Time Embedded DSP Systems…

Components of Embedded Systems

Firmware: Programs deep in the HW, not often changed

Software: Programs that can be changed by the user

Memory RAM, ROM, Flash, …

User Interface: Some LEDs to GUI on LCD

Sensors: Sense the real world

Actuators: React to or control the real word

Emulation and Diagnostics:

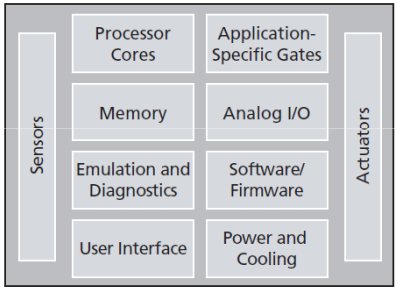
→ *e.g. JTAG (Joint Test Action Group)*

Application specific gates

Analog IO:

→ *A/D and D/A + ASP (filters, amplifiers, …)*

Last but not least the Processor: 8-bit MCU to 64-bit μP can be ASIC or FPGA



Real-Time Embedded DSP Systems…

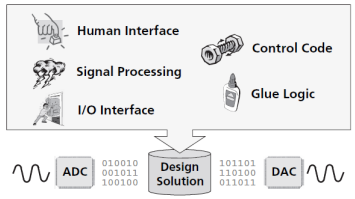
Development Life Cycle

1) Examine the overall system needs

BDTI Selection Criteria:

Price and BOM size, Performance,

Time to Market, Power, Size, Features, …



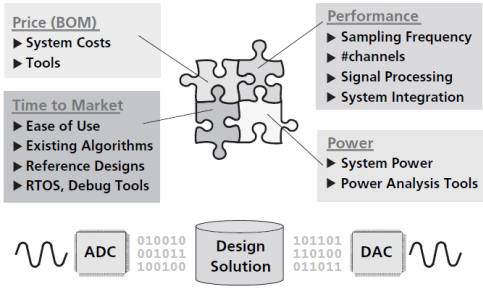
Rules in this phase:

For Fixed Cost,

maximize the performance

For Fixed Performance,

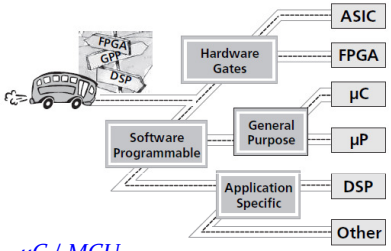
minimize the cost



Real-Time Embedded DSP Systems…

Development Life Cycle…

2) Select the Hardware Components Required



Strengths	▸ Fastest possible computation ▸ Excellent design support tools ▸ Some kind of PLD usually required in design ▸ Ability to synthesize almost any peripheral ▸ Easy to develop with ▸ Flexible features – field reprogrammable
Signal Processing	▸ Excellent for hi-speed & parallel signal processing
Home Run Apps	▸ Glue logic, radar/sensor arrays

FPGA (→ASIC)

µC / MCU

PIC12 · 68HC11/16 · MCS51 · MSP430

Strengths	▸ Good control peripherals ▸ May have ability to use mid-range O/Ss ▸ Very low cost ▸ Integrated FLASH ▸ Can be very low power
Signal Processing	▸ Poor to fair
Home Run Apps	▸ Embedded control, small home appliances

ASIC: Only for extremely low power, extremely high performance high volume, good time to market margin, less flexible

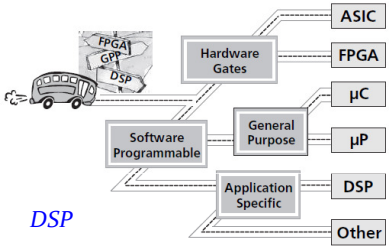
FPGA: More flexible/faster time to market faster than processors, but more development time

Typical application: radar

Real-Time Embedded DSP Systems…

Development Life Cycle…

2) Select the Hardware Components Required…



DSP

BlackFin/Sharc · C2000/C5000/C6000 · DSP56xxx/StarCore

Strengths	▸ Architecture optimized for computing DSP algorithms ▸ Excellent MIP/mW/$$ tradeoff ▸ Efficient compilers – can program entire app in C ▸ Some have a real-time O/S (for task scheduling) ▸ Can be very low power
Signal Processing	▸ Good to excellent
Home Run Apps	▸ Cell phones, telecom infrastructure, digital cameras ▸ DSL/cable/modems, audio/video, multimedia

Strengths	▸ Familiar Design Environment (tools, s/w, emulation) ▸ Robust communication peripherals ▸ Ability to use higher-end O/Ss (control code) ▸ Great for compiling generic (non-tuned) C code
Signal Processing	▸ Fair to good
Home Run Apps	▸ PC, PDA

µP (GPP)

Go for MCU when low cost low chip count, low processing power needed

Go for DSP only if cost, size and power, requirements cannot be met by a GPP or other DSP related features are needed like fast IO, low delay interrupt, specific instructions, … are needed

Real-Time Embedded DSP Systems…

Development Life Cycle…

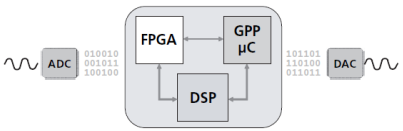
2) Select the Hardware Components Required…

Other rules in this phase:

FPGAs are good for bit manipulation applications

Processors are better for numerical applications

If single DSP will do the job go for it!



DSP better than FPGA if algorithms are so complex and resources are already in DSPs

Have a DSP if special memory accesses needed

Have an FPGA and a DSP if possible to:

- *Use smaller/cheaper DSP/MCU by offloading some computationally intensive tasks to the FPGA*
- *Increase the computational throughput*
- *Make a prototype of a new signal processing algorithm (like an EVM)*
- *To pack the glue logic and have flexibility in it*

Have a GPP along with a DSP if there are so many non-real-time tasks…

Embedded DSP systems development

*Technical trend is towards **programmability***

Real-Time Embedded DSP Systems…

Development Life Cycle…

2) Select the Hardware Components Required…

Programmable or HW or Mixed ?

Most of the time cost is the final saying!

Cost is a multifold issue usually fulfilled by a mixed strategy

Device cost, NRE (non-recurring engineering), Manufacturing cost, Opportunity cost, Low time to market gain, physical advantages (power dissipation, weight, size, …)

3) Understanding DSP basics and architecture

- *What makes a DSP a DSP?!*
- *How fast it can go?*
- *How can I achieve maximum performance? Max # channels for an algorithm*
- *IO options? GPIO, UART, CAN, SPI, USB, McBSP, HPI*
- *IO speed and performance. Can IO keep up with Max # channels?*
- *Memory speed?*

Hi-speed internal memory? Special access, DMA

Sample based and frame based →