Design with Microprocessors

Tajana Simunic Rosing
Department of Computer Science and Engineering
University of California, San Diego.
ES Design

Hardware components

Specification

Concept

Software Components

Hardware

Verification and Validation
Embedded system hardware is frequently used in a loop ("hardware in a loop"): cyber-physical systems
Hardware platform architecture
CPUs

- CPU performance
  - Cycle time.
  - CPU pipeline.
    - Latency & Throughput
  - Memory system.
    - Indeterminacy in execution
    - Cache miss: compulsory, conflict, capacity

- CPU power consumption.

- Compare
  - ARM7, TI C54x, TI 60x DSPs, Xilinx Vertex II, single purpose controllers
Selecting a Microprocessor

- **Issues**
  - Technical: speed, power, size, cost
  - Other: development environment, prior expertise, licensing, etc.

- **Speed: how evaluate a processor’s speed?**
  - Clock speed – but instructions per cycle may differ
  - Instructions per second – but work per instr. may differ
    - MIPS: 1 MIPS = 1757 Dhrystones per second (based on Digital’s VAX 11/780). A.k.a. Dhrystone MIPS. Commonly used today.
    - So, 750 MIPS = 750*1757 = 1,317,750 Dhrystones per second
  - SPEC: set of more realistic benchmarks, but oriented to desktops
    - Suites of benchmarks: automotive, consumer electronics, networking, office automation, telecommunications
RISC vs. CISC

- Complex instruction set computer (CISC):
  - many addressing modes;
  - many operations.

- Reduced instruction set computer (RISC):
  - load/store;
  - pipelinable instructions.
Parallelism in programs

- Parallelism exists in several levels of granularity:
  - Task.
  - Data.
  - Instruction.

- Instruction dependency
  - Data and resource
  - Check at compile &/or run time

```
Ld r1, r2
Add r3, r4
Sub r5, r6
```
Parallelism extraction

- **Static:**
  - Use compiler to analyze program.
  - Simpler CPU control.
  - Can make use of high-level language constructs.
  - Can’t depend on data values.

- **Dynamic:**
  - Use hardware to identify opportunities.
  - More complex CPU.
  - Can make use of data values.
Superscalar

- RISC - 1 inst/cycle
- Superscalar – n inst/cycle
  - \( n \) HW for \( n \)-instr parallel

<table>
<thead>
<tr>
<th>IF</th>
<th>ID</th>
<th>EX</th>
<th>MEM</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>MEM</td>
<td>WB</td>
</tr>
<tr>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>MEM</td>
<td>WB</td>
</tr>
<tr>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>MEM</td>
<td>WB</td>
</tr>
<tr>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>MEM</td>
<td>WB</td>
</tr>
<tr>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>MEM</td>
<td>WB</td>
</tr>
<tr>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>MEM</td>
<td>WB</td>
</tr>
<tr>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>MEM</td>
<td>WB</td>
</tr>
</tbody>
</table>

Execution unit

Register file

\( n \)
Simple VLIW architecture

- Compile time assignment of instructions to FUs
- Large register file feeds multiple function units.

```
Add r1,r2,r3; Sub r4,r5,r6; Ld r7,foo; St r8,baz; NOP
```
Clustered VLIW architecture

- Register file, function units divided into clusters.

![Diagram of Clustered VLIW architecture](image)
Types of CPUs used in ES

- RISC CPUs
  - ARM 7
- CISC CPUs
  - TI C54x
- VLIW
  - TI C6x
- FPGA – Programmable CPUs
  - Virtex II
- Single purpose processors
ARM7 design

- ARM assembly language - RISCy
- ARM programming model
  - Audio players, pagers etc.; 130 MIPS
- ARM memory organization.
- ARM data operations (32 bit)
- ARM flow of control.
# ARM programming model

<table>
<thead>
<tr>
<th>r0</th>
<th>r1</th>
<th>r2</th>
<th>r3</th>
<th>r4</th>
<th>r5</th>
<th>r6</th>
<th>r7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>r8</th>
<th>r9</th>
<th>r10</th>
<th>r11</th>
<th>r12</th>
<th>r13</th>
<th>r14</th>
<th>r15 (PC)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Current Program Status Register (CPSR)**
  - 31: **N**
  - 30: **Z**
  - 29: **C**
  - 28: **V**
  - 0: **0**
ARM status bits

- Every arithmetic, logical, or shifting operation sets CPSR bits:
  - N (negative), Z (zero), C (carry), V (overflow).

- Examples:
  - $-1 + 1 = 0$: NZCV = 0110.
  - $2^{31}-1+1 = -2^{31}$: NZCV = 1001.
ARM pipeline execution

**ARM Pipeline Stages:**
- **Fetch**
- **Decode**
- **Execute**

**Example Instructions:**
- `add r0, r1, #5`
- `sub r2, r3, r6`
- `cmp r2, #3`
ARM data instructions

- ADD, ADC : add (w. carry)
- SUB, SBC : subtract (w. carry)
- MUL, MLA : multiply (and accumulate)
- AND, ORR, EOR
- BIC : bit clear
- LSL, LSR : logical shift left/right
- ASL, ASR : arithmetic shift left/right
- ROR : rotate right
- RRX : rotate right extended with C
ARM flow of control

- All operations can be performed conditionally, testing CPSR:
  - $\text{EQ, NE, CS, CC, MI, PL, VS, VC, HI, LS, GE, LT, GT, LE}$

- Branch operation:
  - $\text{B } #100$
  - Can be performed conditionally.
ARM comparison instructions

- CMP : compare
- CMN : negated compare
- TST : bit-wise AND
- TEQ : bit-wise XOR

These instructions set only the NZCV bits of CPSR.
ARM load/store/move instructions

- **LDR, LDRH, LDRB** : load (half-word, byte)
- **STR, STRH, STRB** : store (half-word, byte)

**Addressing modes:**
- register indirect: `LDR r0, [r1]`
- with second register: `LDR r0, [r1, -r2]`
- with constant: `LDR r0, [r1, #4]`

**MOV, MVN** : move (negated)

```
MOV r0, r1 ; sets r0 to r1
```
Addressing modes

- **Base-plus-offset addressing:**
  \[ \text{LDR } r0, [r1, #16] \]
  - Loads from location \( r1+16 \)

- **Auto-indexing increments base register:**
  \[ \text{LDR } r0, [r1, #16]! \]

- **Post-indexing fetches, then does offset:**
  \[ \text{LDR } r0, [r1], #16 \]
  - Loads \( r0 \) from \( r1 \), then adds 16 to \( r1 \).
ARM subroutine linkage

- Branch and link instruction:
  ```
  BL foo
  ```
  - Copies current PC to r14.

- To return from subroutine:
  ```
  MOV r15, r14
  ```
ARM Summary

- Load/store architecture
- Most instructions are RISCy, operate in single cycle.
  - Some multi-register operations take longer.
- All instructions can be executed conditionally.
What is DSP?

- Digital Signal Processing – the processing or manipulation of signals using digital techniques

Source: Dr D. H. Crawford
What is DSP Used For?

...And much more!

Source: Dr D. H. Crawford
Speech Processing

- Speech coding/compression
- Speech synthesis
- Speech recognition

Source: Dr D. H. Crawford
Some Properties of Speech

The blue-- s---p---o---t i-s---on-the-- k--ey a---g--a----n-----

"k" in "key"

Source: Dr D. H. Crawford
Still Image Coding:

- JPEG (Joint Photographic Experts Group):
  - Discrete Cosine Transform (DCT) based
    - DCT transform of an image brings out a set of coefficients. If a coefficient has a lot of variance over a set of images, then it cannot be removed without affecting the picture quality.
- JPEG2000: Wavelet Transform based

Video Coding:

- MPEG (Moving Pictures Experts Group):
  - DCT-based,
  - Interframe and intraframe prediction,
  - Motion estimation.
- Applications: Digital TV, DVD, etc.

Source: Dr D. H. Crawford
DCT & Inverse DCT

The image is broken into 8x8 groups, each containing 64 pixels. Three of these 8x8 groups are enlarged in this figure, showing the values of the individual pixels, a single byte value between 0 and 255.
DCT – more details

Divide picture into 16 by 16 blocks. (macroblocks)

Each macroblock is 16 pixels by 16 lines. (4 blocks)

Each block is 8 pixels by 8 lines.

8 X 8 Block → DCT → Frequency Coefficients

Source: Xilinx
2D DCT & IDCT Equations

**DCT**

\[
F[u, v] = \frac{1}{N^2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} f[m, n] \cos \left( \frac{(2m + 1)u\pi}{2N} \right) \cos \left( \frac{(2n + 1)v\pi}{2N} \right)
\]

where:
- \(u, v\) = discrete frequency variables (0, 1, 2, ..., N - 1),
- \(f[m, n]\) = N by N image pixels (0, 1, 2, ..., N - 1), and
- \(F[u, v]\) = the DCT result

\[
f[m, n] = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} c[u] c[v] F[u, v] \cos \left( \frac{(2m + 1)u\pi}{2N} \right) \cos \left( \frac{(2n + 1)v\pi}{2N} \right)
\]

where:
- \(m, n\) = image result pixel indices (0, 1, 2, ..., N - 1),
- \(F[u, v]\) = N by N DCT result,
- \(c[\lambda]\) = 1 for \(\lambda = 0\) and \(c[\lambda] = 2\) for \(\lambda = 1, 2, 3, ..., N-1\)
- \(f[m, n]\) = N by N IDCT result

Source: Xilinx
DCT in MPEG2
MPEG time domain processing

- Use motion vectors to specify how a 16x16 macroblock translates between reference frames and current frame, then code difference between reference and actual block.
GOP (Group of Pictures)

- GOP is a set of consecutive frames that can be decoded without any other reference frames.
- Usually 12 or 15 frames.
- Transmitted sequence is not the same as displayed sequence.

![Figure 1: Prediction between MPEG-2 Frames](image)
Types of frames

- I frame (intra-coded)
  - Coded without reference to other frames

- P frame (predictive-coded)
  - Coded with reference to a previous reference frame (either I or P)
  - Size is usually about $\frac{1}{3}$rd of an I frame

- B frame (bi-directional predictive-coded)
  - Coded with reference to both previous and future reference frames (either I or P)
  - Size is usually about $\frac{1}{6}$th of an I frame
MPEG Block Diagram
DSP Devices & Architectures

Selecting a DSP – several choices:
- Fixed-point;
- Floating point;
- Application-specific devices (e.g. FFT processors, speech recognizers, etc.).

Main DSP Manufacturers:
- Texas Instruments (http://www.ti.com)
- Motorola (http://www.motorola.com)
- Analog Devices (http://www.analog.com)

Source: Dr D. H. Crawford
Typical DSP Operations

- Filtering
- Energy of Signal
- Frequency transforms

\[ y(n) = \sum_{i=0}^{L-1} a_i x(n - i) \]

Pseudo C code

```c
for (n=0; n<N; n++)
{
    s=0;
    for (i=0; i<L; i++)
    {
        s += a[i] * x[n-i];
    }
    y[n] = s;
}
```

Source: Dr D. H. Crawford
DSPs and Fixed-Point Design

- Digital signal processing algorithms
  - Often developed in floating point
  - Later mapped into fixed point for digital hardware realization
- Fixed-point digital hardware
  - Lower area
  - Lower power
  - Lower per unit production cost
Fixed-Point Design

- Float-to-fixed point conversion required to target
  - ASIC and fixed-point digital signal processor core
  - FPGA and fixed-point microprocessor core
- All variables have to be annotated manually
  - Avoid overflow
  - Minimize quantization effects
  - Find optimum wordlength
- Manual process supported by simulation
  - Time-consuming
  - Error prone
Fixed-Point Representation

- Fixed point type
  - Wordlength
  - Integer wordlength
- Quantization modes
  - Round
  - Truncation
- Overflow modes
  - Saturation
  - Saturation to zero
  - Wrap-around

SystemC format
www.systemc.org

Copyright Kyungtae Han [2]
Fixed-point arithmetic

Shifting required after multiplications and divisions in order to maintain binary point.
Properties of fixed-point arithmetic

- Automatic scaling is an important advantage for multiplications.

- Example:
  \[ x = 0.5 \times 0.125 + 0.25 \times 0.125 = 0.0625 + 0.03125 = 0.09375 \]
  For \( iwL = 1 \) and \( fwl = 3 \) decimal digits, the less significant digits are automatically chopped off: \( x = 0.093 \)
  Like a floating point system with numbers \( \in (-1..1) \), with no stored exponent (bits used to increase precision).

- Appropriate for DSP/multimedia applications (well-known value ranges)
DSPs

- TI DSPs:
  - Basic features.
  - CISC:
    - C54x family
    - C55x & co-processors
  - VLIW:
    - C60x
C5x family

- Fixed-point DSP.
- Modified Harvard architecture:
  - 1 program memory bus.
  - 3 data memory busses.
- 40-bit ALU.
- Multiple implementations:
  - 1, 2 instructions/cycle.
TI C54x architectural features

- 40-bit ALU + barrel shifter.
- Multiple internal busses: 1 instruction, 3 data, 4 address.
- 17 x 17 multiplier.
- Single-cycle exponent encoder.
- Two address generators with dedicated registers.
TI C54x instruction set features

- Specialized instructions for Viterbi.
- Repeat and block repeat instructions.
- Instructions that read 2, 3 operands simultaneously.
- Conditional store.
- Fast return from interrupt.
C54x CPU

- 40-bit ALU.
- Two 40-bit accumulators.
- Barrel shifter.
- 17 x 17 multiplier/adder.
- Compare/select/store (CSSU) unit.
C54x architectural elements

- **ALU:**
  - 40-bit arithmetic, Boolean operations.
  - Two 16-bit operations when status register 1 C16 bit is set.

- **Accumulators:**
  - Low-order (0-15), high-order (16-31), guard (32-39).

- **Barrel shifter:**
  - Input from accumulator or data memory.
  - Output to ALU.

- **Multiplier:**
  - 17 x 17 multiply with 40-bit accumulate.

- **CSSU unit:**
  - Compares high and low accumulator words.
  - Accelerates Viterbi operations.
C54x registers

- Status registers ST0, ST1:
  - Arithmetic, bit manipulation flags.
  - Data page pointer, auxiliary register pointer.
  - Processor modes.

- Auxiliary registers:
  - Used to generate 16-bit data space addresses.

- Temporary register:
  - Used to hold one multiplicand or dynamic shift count.

- Transition register:
  - Used for Viterbi operations.

- Stack pointer:
  - Top of system stack.

- Circular buffer size register.

- Block-repeat registers.

- Interrupt registers.

- Processor mode status register.
C54x pipeline

- **Program prefetch.** Send PC address on program address bus.
- **Fetch.** Load instruction from program bus to IR.
- **Decode.**
- **Access.** Put operand addresses on busses.
- **Read.** Get operands from busses.
- **Execute.**
C54x power down modes

- Three IDLE instructions:
  - IDLE1 shuts down CPU.
  - IDLE2 shuts down CPU and on-chip peripherals.
  - IDLE3 shuts down chip completely (including PLL).
C54x busses

- PB: program read bus.
- CB, DB: data read busses.
- EB: data write bus.
- PAB, CAB, DAB, EAB: address busses.
- Can generate two data memory addresses per cycle.
  - Stored in auxiliary register address units ARAU0, ARAU1.
# Addressing Modes

<table>
<thead>
<tr>
<th>Addressing mode</th>
<th>Operand field</th>
<th>Register-file contents</th>
<th>Memory contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Immediate</td>
<td>Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Register-direct</td>
<td>Register address</td>
<td>Data</td>
<td></td>
</tr>
<tr>
<td>Register indirect</td>
<td>Register address</td>
<td>Memory address</td>
<td>Data</td>
</tr>
<tr>
<td>Direct</td>
<td>Memory address</td>
<td></td>
<td>Data</td>
</tr>
<tr>
<td>Indirect</td>
<td>Memory address</td>
<td></td>
<td>Memory address</td>
</tr>
</tbody>
</table>
Common addressing modes

- ARn (*): indirect through auxiliary registers.
- DP (@): direct addressing offset from DP register.
- K23 (#): absolute addressing using label.
- Bit addressing (BIT instruction): modify a single bit of a memory location or MMR register.
C54x instructions

- ABDST: absolute distance
- ADD
- ADDC: add w. carry
- ADDM: add immediate to mem
- ADDS: add w/o sign extension
- DADD: double add
- DELAY: memory delay
- DSUB: double subtract
- EXP: accumulator exponent
- LMS: least mean square
- MAC: multiply accumulate
- MACA: multiply by MACA, add to MACB
C54x instructions, cont’d.

- MACP: multiply by program memory, then accumulate
- MAS: multiply by T, then subtract
- MAX, MIN
- MPY: multiply
- NEG: negate
- NORM: normalize

- POLY: evaluate polynomial
- RND: round accumulator
- SAT: saturate accumulator
- SQRUR: square
- SUB: subtract
C54x instructions, cont’d.

- AND
- BIT: test bit
- BITF: test bit shown by immediate
- CMPL: complement accumulator
- OR
- ROL: rotate accumulator left

- SFTA: shift accumulator arithmetically
- XOR
- MVDD: move within data memory
- MVDP: move data to program memory
- READA: read data addressed by ACCA
- WRITA: write data addressed by ACCA

- And many more…. 
CISC CPU: TI’s C55x

- **Pipeline segments:**
  - Fetch.
  - Execute.

![Diagram of pipeline segments](fetch_execute.png)

- **fetch**
- **execute**

- 4
- 7-8
C55x fetch segment

- Prefetch 1:
  - Send address to memory.

- Prefetch 2:
  - Wait for response.

- Fetch:
  - Get instruction from memory and put in IBQ.

- Predecode:
  - Identify where instructions begin and end; identify parallel instructions.
C55x execute segment

- Decode:
  - Decode an instruction pair or single instruction.

- Address:
  - Perform address calculations.

- Access 1/2:
  - Send address to memory; wait.

- Read:
  - Read data from memory. Evaluate condition registers.

- Execute:
  - Read/modify registers. Set conditions.

- W/W+:
  - Write data to MMR-addressed registers, memory; finish.
C55x organization

3 data read busses
3 data read address busses
program address bus

program read bus

Instruction unit
Program flow unit
Address unit
Data unit

32
B, D busses

16
24
24

2 data write busses
2 data write address busses

Single operand
Dual operand

Writes

Instruction fetch
Data read from memory

D, C, D busses

Dual - multiply
Coefficient

Tajana Simunic Rosing
Image/video hardware extensions

- Available in 5509 and 5510.
  - Equivalent C-callable functions for other devices.

- Available extensions:
  - DCT/IDCT.
  - Pixel interpolation
  - Motion estimation.
DCT/IDCT

- 2-D DCT/IDCT is computed from two 1-D DCT/IDCT.
- Put data in different banks to maximize throughput.
C55 motion estimation

- Search strategy:
  - Full vs. non-full.

- Accuracy:
  - Full-pixel vs. half-pixel.

- Number of returned motion vectors:
  - 1 (one 16x16) vs. 4 (four 8x8).

- Algorithms:
  - 3-step algorithm (distance 4,2,1).
  - 4-step algorithm (distance 8,4,2,1).
  - 4-step with half-pixel refinement.
Types of CPUs used in ES

- RISC CPUs
  - ARM 7
- CISC CPUs
  - TI C54x
- VLIW
  - TI C6x
- FPGA – Programmable CPUs
  - Virtex II
VLIW: TI C62/C67

- Up to 8 instructions/cycle.
- 32 32-bit registers.
- Function units:
  - Two multipliers.
  - Six ALUs.
- Data operations:
  - 8/16/32-bit arithmetic.
  - 40-bit operations.
  - Bit manipulation operations.
Partitioned register files

- Many memory ports are required to supply enough operands per cycle.
- Memories with many ports are expensive.

Registers are partitioned into sets, e.g. for TI C60x:
C6x data paths

- General-purpose register files (A and B, 16 words each).
- Eight function units:
  - .L1, .L2, .S1, .S2, .M1, .M2, .D1, .D2
- Two load units (LD1, LD2).
- Two store units (ST1, ST2).
- Two register file cross paths (1X and 2X).
- Two data address paths (DA1 and DA2).
C6x function units

- **.L**
  - 32/40-bit arithmetic.
  - Leftmost 1 counting.
  - Logical ops.

- **.S**
  - 32-bit arithmetic.
  - 32/40-bit shift and 32-bit field.
  - Branches.
  - Constants.

- **.M**
  - 16 x 16 multiply.

- **.D**
  - 32-bit add, subtract, circular address.
  - Load, store with 5/15-bit constant offset.
C6x system

- On-chip RAM.
- 32-bit external memory: SDRAM, SRAM, etc.
- Host port.
- Multiple serial ports.
- Multichannel DMA.
- 32-bit timer.
DSPs

- Great for multimedia
- CISC
  - MMX
  - TI C54x, C55x
- VLIW
  - TI C6x
Programmable Logic Devices (PLD)

- PLDs combine PLA/PAL with memory and other advanced structures
  - Similar to PLA/PAL, hence Field-Programmable Gate Arrays

- Types:
  - Antifuse PLDs
  - EPLD & EEPLD
  - FPGAs with RAMs
  - FPGA with processing
    - Digital Signal Processing
    - General purpose CPU

<table>
<thead>
<tr>
<th>Name</th>
<th>Re-programmable</th>
<th>Volatile</th>
<th>Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fuse</td>
<td>no</td>
<td>no</td>
<td>Bipolar</td>
</tr>
<tr>
<td>EPROM</td>
<td>yes</td>
<td>no</td>
<td>UVCMOS</td>
</tr>
<tr>
<td></td>
<td>out of circuit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EEPROM</td>
<td>yes</td>
<td>no</td>
<td>EECMOS</td>
</tr>
<tr>
<td></td>
<td>in circuit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRAM</td>
<td>yes</td>
<td>yes</td>
<td>CMOS</td>
</tr>
<tr>
<td></td>
<td>in circuit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Antifuse</td>
<td>no</td>
<td>no</td>
<td>CMOS+</td>
</tr>
</tbody>
</table>
Field-Programmable Gate Arrays

- Logic blocks
  - To implement combinational and sequential logic

- Interconnect
  - Wires to connect inputs and outputs to logic blocks

- I/O blocks
  - Special logic blocks at periphery of device for external connections

- Key questions:
  - How to make logic blocks programmable?
  - How to connect the wires?
  - *After the chip has been manufactured*
Antifuse PLDs

Actel’s Axcelerator Family

- **Antifuse:**
  - open when not programmed
  - Low resistance when programmed
Actel’s Axcelerator C-Cell

- **C-Cell**
  - Basic multiplexer logic plus more inputs and support for fast carry calculation
  - Carry connections are “direct” and do not require propagation through the programmable interconnect
Actel’s Accelerator R-Cell

- R-Cell
  - Core is D flip-flop
  - Muxes for altering the clock and selecting an input
  - Feed back path for current value of the flip-flop for simple hold
  - Direct connection from one C-cell output of logic module to an R-cell input; Eliminates need to use the programmable interconnect

- Interconnection Fabric
  - Partitioned wires
  - Special long wires
Altera’s Erasable Programmable Logic Devices (EPLDs)

- Historical Perspective
  - PALs: same technology as programmed once bipolar PROM
  - EPLDs: CMOS erasable programmable ROM (EPROM) erased by UV light

- Altera building block = MACROCELL
Altera EPLDs contain 10s-100s of independently programmed macrocells.

Personalized by EPROM bits:

**Synchronous Mode**
- Flipflop controlled by global clock signal

**Asynchronous Mode**
- Flipflop controlled by locally generated clock signal

+ Seq Logic: could be D, T positive or negative edge triggered
+ product term to implement clear function
AND-OR structures are relatively limited
Cannot share signals/product terms among macrocells

Global Routing: Programmable Interconnect Array

EPM5128:
- 8 Fixed Inputs
- 52 I/O Pins
- 8 LABs
- 16 Macrocells/LAB
- 32 Expanders/LAB

Logic Array Blocks
(similar to macrocells)
Altera’s EEPLD

- Altera’s MAX 7k Block Diagram
EEPLD

- Altera’s MAX 7k Logic Block
SRAM based PLD

- Altera’s Flex 10k Block Diagram
SRAM based PLD

- Altera’s Flex 10k Logic Array Block (LAB)
SRAM based PLD

- Altera’s Flex 10k Logic Element (LE)
FPGA with DSP

- Altera’s Stratix II: Block Diagram
FPGA with DSP

- Altera’s Stratix II:
  - DSP Detail
Xilinx Vertex-II Family

- 88-1000+ pins
- 64-10000+ CLBs
  - Combinational and sequential logic using lookup tables and flip-flops
  - Random-access memory
  - Shift registers for use as buffer storage
- Multipliers regularly placed throughout the CLB array to accelerate digital signal processing applications
- E.g., the XC2V8000: 11,648 CLBs, 1108 IOBs, 90,000+ FFs, 3Mbits RAM (168 x 18Kbit blocks), 168 multipliers
  - Equivalent to eight million two-input gates!
Programmable Interconnect

Configurable Logic Blocks (CLBs)

I/O Blocks (IOBs)
Configurable Logic Block (CLB)

fast connects to neighbours
2 carry paths per CLB (Vertex II Pro)

Virtex II Slice

Look-up tables LUT F and G can be used to compute any Boolean function of ≤ 4 variables.

Example:

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Virtex II Pro
Devices include up to 4 PowerPC processor cores
FPGA with General Purpose CPU & Analog

- Actel’s Fusion Family Diagram
  - FPGA with ARM 7 CPU and Analog Components

![Fusion Family Diagram]

- Optional ARM or 8051 Processor
- User Applications
- Fusion Applets
- Fusion Smart Backbone
  - Analog Smart Peripheral 1
  - Analog Smart Peripheral 2
  - Analog Smart Peripheral n
  - Smart Peripherals in FPGA Fabric (e.g. logic, PLL, FIFO)
Single-purpose processors

- Performs specific computation task
- Custom single-purpose processors
  - Designed for a unique task
- Standard single-purpose processors
  - "Off-the-shelf" -- pre-designed for a common task
  - a.k.a., peripherals
  - serial transmission
  - analog/digital conversions
Timers, counters

- **Timer**: measures time intervals by counting clock pulses
  - To generate timed output events e.g., hold light for 10 s
  - To measure input events e.g., measure a car’s speed

- **Watchdog timer**
  - Reset timer every X time units, else it generates a signal
  - Uses: detect failure, self-reset, timeout on an ATM machine

- **Counter**: counts pulses on a general input signal
  - E.g. count cars passing over by a sensor

---

**Basic timer**

- Clk
- 16-bit up counter
- 16
- Cnt
- Top
- Reset

**Timer/counter**

- Clk
- 2x1 mux
- Cnt_in
- Mode
- Reset
- 16
- Cnt
- Top
- 16-bit up counter
Pulse width modulator

- Generates pulses with specific high/low times
- Duty cycle: % time high
  - Square wave: 50% duty cycle
- Common use: control average voltage to an electric device
  - Simpler than DC-DC converter or digital-analog converter
  - DC motor speed, dimmer lights
- Another use: encode commands, receiver uses timer to decode

25% duty cycle – average pwm_o is 1.25V

50% duty cycle – average pwm_o is 2.5V.

75% duty cycle – average pwm_o is 3.75V.
void WriteChar(char c){
    RS = 1;    /* indicate data being sent */
    DATA_BUS = c;  /* send data to LCD */
    EnableLCD(45);  /* toggle the LCD with appropriate delay */
}

<table>
<thead>
<tr>
<th>CODES</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/D</td>
<td>Sets cursor move direction and/or specifies not to shift display</td>
</tr>
<tr>
<td>I/D = 1 cursor moves left</td>
<td>DL = 1 8-bit</td>
</tr>
<tr>
<td>I/D = 0 cursor moves right</td>
<td>DL = 0 4-bit</td>
</tr>
<tr>
<td>S</td>
<td>Sets interface data length, number of display lines, and character font</td>
</tr>
<tr>
<td>S = 1 with display shift</td>
<td>N = 1 2 rows</td>
</tr>
<tr>
<td>S/C</td>
<td>Move cursor and shifts display</td>
</tr>
<tr>
<td>S/C = 1 display shift</td>
<td>N = 0 1 row</td>
</tr>
<tr>
<td>S/C = 0 cursor movement</td>
<td>F = 1 5x10 dots</td>
</tr>
<tr>
<td>R/L</td>
<td>Sets ON/OFF of all display(D), cursor ON/OFF (C), and blink position (B)</td>
</tr>
<tr>
<td>R/L = 1 shift to right</td>
<td>F = 0 5x7 dots</td>
</tr>
<tr>
<td>R/L = 0 shift to left</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RS</th>
<th>R/W</th>
<th>DB7</th>
<th>DB6</th>
<th>DB5</th>
<th>DB4</th>
<th>DB3</th>
<th>DB2</th>
<th>DB1</th>
<th>DB0</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Clears all display, return cursor home</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Returns cursor home</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Sets cursor move direction and/or specifies not to shift display</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>I/D</td>
<td>ON/OFF of all display(D), cursor ON/OFF (C), and blink position (B)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>D</td>
<td>C</td>
<td>Move cursor and shifts display</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Sets interface data length, number of display lines, and character font</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>WRITE DATA</td>
<td>Writes Data</td>
</tr>
</tbody>
</table>
Keypad controller

N=4, M=4
Summary

- RISC CPUs
  - ARM 7
- CISC CPUs
  - TI C54x
- VLIW
  - TI C6x
- FPGA – Programmable CPUs
  - Altera, Xilinx, Actel
- Single purpose processors
Sources and References