CSE 237A
Design with Microprocessors

Tajana Simunic Rosing
Department of Computer Science and Engineering
University of California, San Diego.
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, TriMedia, Pentium MMX, 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
# General Purpose Processors

<table>
<thead>
<tr>
<th>Processor</th>
<th>Clock speed</th>
<th>Periph.</th>
<th>Bus Width</th>
<th>MIPS</th>
<th>Power</th>
<th>Trans.</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Xeon 65nm tech.</td>
<td>3.5GHz</td>
<td>L1 16KB data, L2 8MB, L3 16MB</td>
<td>64</td>
<td>~900</td>
<td>150W</td>
<td>~1.3B</td>
<td>~$2000</td>
</tr>
<tr>
<td>IBM PowerPC 750X</td>
<td>550 MHz</td>
<td>2x32 K L1, 256K L2</td>
<td>32/64</td>
<td>~1300</td>
<td>5W</td>
<td>~7M</td>
<td>$900</td>
</tr>
<tr>
<td>MIPS R5000</td>
<td>250 MHz</td>
<td>2x32 K L1, 256K L2</td>
<td>32/64</td>
<td>NA</td>
<td>NA</td>
<td>3.6M</td>
<td>NA</td>
</tr>
<tr>
<td>StrongARM SA-110</td>
<td>233 MHz</td>
<td>None</td>
<td>32</td>
<td>268</td>
<td>1W</td>
<td>2.1M</td>
<td>NA</td>
</tr>
</tbody>
</table>

## Microcontroller

<table>
<thead>
<tr>
<th>Processor</th>
<th>Clock speed</th>
<th>Periph.</th>
<th>Bus Width</th>
<th>MIPS</th>
<th>Power</th>
<th>Trans.</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel 8051</td>
<td>12 MHz</td>
<td>4K ROM, 128 RAM, 32 I/O, UART</td>
<td>8</td>
<td>~1</td>
<td>~0.2 W</td>
<td>~10K</td>
<td>$7</td>
</tr>
<tr>
<td>Motorola 68HC811</td>
<td>3 MHz</td>
<td>4K ROM, 192 RAM, 32 I/O, Timer, WDT, SPI</td>
<td>8</td>
<td>~.5</td>
<td>~0.1 W</td>
<td>~10K</td>
<td>$5</td>
</tr>
</tbody>
</table>

## Digital Signal Processors

<table>
<thead>
<tr>
<th>Processor</th>
<th>Clock speed</th>
<th>Periph.</th>
<th>Bus Width</th>
<th>MIPS</th>
<th>Power</th>
<th>Trans.</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>TI C5416</td>
<td>160 MHz</td>
<td>128K, SRAM, 3 T1 Ports, DMA, 13 ADC, 9 DAC</td>
<td>16/32</td>
<td>~600</td>
<td>0.22 W</td>
<td>NA</td>
<td>$34</td>
</tr>
<tr>
<td>TMS320</td>
<td>80 MHz</td>
<td>32KB on chip</td>
<td>32</td>
<td>80</td>
<td>0.01 W</td>
<td>NA</td>
<td>$8</td>
</tr>
</tbody>
</table>

Sources: Intel, Motorola, MIPS, ARM, TI, and IBM Website/Datasheet; Embedded Systems Programming
RISC vs. CISC

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

- Reduced instruction set computer (RISC):
  - load/store;
  - pipelinable instructions.
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
- \( n^2 \) HW for n-instr parallel execution
Simple VLIW architecture

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

```
E box
Add r1,r2,r3; Sub r4,r5,r6; Ld r7,foo; St r8,baz; NOP
```

Diagram showing the architecture with:
- ALU
- ALU
- Load/store
- Load/store
- FU

Tajana Simunic Rosing
Clustered VLIW architecture

- Register file, function units divided into clusters.

Cluster bus

Execution

Register file
Types of CPUs used in ES

- RISC CPUs
  - ARM 7
- CISC CPUs
  - TI C54x
- VLIW
  - TI C6x
  - TriMedia
- 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

```
r0
r1
r2
r3
r4
r5
r6
r7
r8
r9
r10
r11
r12
r13
r14
r15 (PC)
```

CPSR

N Z C V
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

```
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:
  - EQ, NE, CS, CC, MI, PL, VS, VC, HI, LS, GE, LT, GT, LE

- Branch operation:
  - 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)
  MV 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.
Multimedia CPUs

Many registers, adders etc are wide (32/64 bit), Multimedia data types are narrow
  • e.g. 8 bit per color, 16 bit per audio sample per channel
  2-8 values can be stored per register and added

4 additions per instruction; carry disabled at word boundaries.
**Pentium MMX**

64-bit vectors representing 8 byte encoded, 4 word encoded or 2 double word encoded numbers.  
*wrap around/saturating* options.  
Multimedia registers mm0 - mm7, consistent with floating-point registers (OS unchanged).

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Options</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Padd[b/w/d]</td>
<td><em>wrap around,</em></td>
<td>addition/subtraction of</td>
</tr>
<tr>
<td></td>
<td><em>saturating</em></td>
<td>bytes, words, double words</td>
</tr>
<tr>
<td>PSub[b/w/d]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pcmpeq[b/w/d]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pcmpgt[b/w/d]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pmulhw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pmulhw</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Pentium MMX

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Psra[w/d]</td>
<td>Parallel shift of words, double words or 64 bit quad words</td>
</tr>
<tr>
<td>Psll[w/d/q]</td>
<td>No. of positions in register or instruction</td>
</tr>
<tr>
<td>Psrl[w/d/q]</td>
<td></td>
</tr>
<tr>
<td>Punpckl[bw/wd/dq]</td>
<td>Parallel unpack</td>
</tr>
<tr>
<td>Punpckh[bw/wd/dq]</td>
<td>Parallel unpack</td>
</tr>
<tr>
<td>Packss[wb/dw]</td>
<td>saturating</td>
</tr>
<tr>
<td>Pand, Pandn</td>
<td>Parallel pack</td>
</tr>
<tr>
<td>Por, Pxor</td>
<td>Logical operations on 64 bit words</td>
</tr>
<tr>
<td>Mov[d/q]</td>
<td>Move instruction</td>
</tr>
</tbody>
</table>
DSPs

- TI C5x DSP:
  - Basic features.
  - C54x architecture and programming.
  - C55x architecture and programming.
  - C55x co-processor.
  - 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>Data</td>
<td></td>
</tr>
<tr>
<td>Indirect</td>
<td>Memory address</td>
<td>Memory address</td>
<td>Data</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
- SQUR: square
- SUB: subtract
- NORM: normalize
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….
C55x pipeline

- Two segments:
  - Fetch.
  - Execute.

Diagram:
- Fetch segment: 4
- Execute segment: 7-8
- Total: 4 to 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
- 16-bit busses
- 24-bit busses
- 32-bit busses
- 2 data write busses
- 2 data write address busses

Instruction fetch

Data read from memory

Single operand read

Dual-multiply coefficient

Writes to memory

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.

Diagram:
- Block
- Column DCT
- Interim
- Row DCT
- DCT
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
  - TriMedia
- 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:

![Diagram of partitioned register files]

Data path A

- register file A
- L1, S1, M1, D1

Data path B

- register file B
- D2, M2, S2, L2

Address bus

Data bus
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.
VLIW: Trimedia TM-1

![Diagram of VLIW: Trimedia TM-1](image_url)
TM-1 characteristics

- Characteristics
  - Floating point support
  - Sub-word parallelism support
  - If Conversion
  - Additional custom operations

<table>
<thead>
<tr>
<th>Functional Unit</th>
<th>#Functional Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>5</td>
</tr>
<tr>
<td>Integer ALU</td>
<td>5</td>
</tr>
<tr>
<td>Load/Store</td>
<td>2</td>
</tr>
<tr>
<td>DSP ALU</td>
<td>2</td>
</tr>
<tr>
<td>DSPMUL</td>
<td>2</td>
</tr>
<tr>
<td>Shifter</td>
<td>2</td>
</tr>
<tr>
<td>Branch</td>
<td>3</td>
</tr>
<tr>
<td>INT/Float MUL</td>
<td>2</td>
</tr>
<tr>
<td>Float ALU</td>
<td>2</td>
</tr>
<tr>
<td>Float Compare</td>
<td>1</td>
</tr>
<tr>
<td>Float sqrt/div</td>
<td>1</td>
</tr>
<tr>
<td>#Register</td>
<td>128</td>
</tr>
<tr>
<td>Instruction cache</td>
<td>32KB, 8 way</td>
</tr>
<tr>
<td>Data cache</td>
<td>16KB, 8 way</td>
</tr>
<tr>
<td>#Operation slots/instruction</td>
<td>5</td>
</tr>
</tbody>
</table>
Philips TriMedia-Processor

For multimedia-applications, up to 5 instructions/cycle.
DSPs

- Great for multimedia
- CISC
  - MMX
  - TI C54x, C55x
- VLIW
  - TI C6x
  - TriMedia
VIRTEX II FPGAs
Configurable Logic Block (CLB)

- COUT
- CIN

fast connects to neighbourghs
2 carry paths per CLB (Vertex II Pro)
Look-up tables LUT F and G can be used to compute any Boolean function of \( \leq 4 \) variables.

Example:

\[
\begin{array}{cccc|c}
 a & b & c & d & \text{G} \\
 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 1 & 1 \\
 0 & 0 & 1 & 0 & 1 \\
 0 & 0 & 1 & 1 & 0 \\
 0 & 1 & 0 & 0 & 1 \\
 0 & 1 & 0 & 1 & 0 \\
 0 & 1 & 1 & 0 & 0 \\
 0 & 1 & 1 & 1 & 1 \\
 1 & 0 & 0 & 0 & 1 \\
 1 & 0 & 0 & 1 & 0 \\
 1 & 0 & 1 & 0 & 0 \\
 1 & 0 & 1 & 1 & 1 \\
 1 & 1 & 0 & 0 & 0 \\
 1 & 1 & 0 & 1 & 1 \\
 1 & 1 & 1 & 0 & 1 \\
 1 & 1 & 1 & 1 & 0 \\
\end{array}
\]
Virtex II Pro Devices include up to 4 PowerPC processor cores


Tajana Simunic Rosing
## Number of resources in Virtex II

*Table 16: Virtex-II Pro Logic Resources Available in All CLBs*

<table>
<thead>
<tr>
<th>Device</th>
<th>CLB Array: Row x Column</th>
<th>Number of Slices</th>
<th>Number of LUTs</th>
<th>Max Distributed SelectRAM+ or Shift Register (bits)</th>
<th>Number of Flip-Flops</th>
<th>Number of Carry Chains&lt;sup&gt;(1)&lt;/sup&gt;</th>
<th>Number of SOP Chains&lt;sup&gt;(1)&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>XC2VP2</td>
<td>16 x 22</td>
<td>1,408</td>
<td>2,816</td>
<td>45,056</td>
<td>2,816</td>
<td>44</td>
<td>32</td>
</tr>
<tr>
<td>XC2VP4</td>
<td>40 x 22</td>
<td>3,008</td>
<td>6,016</td>
<td>96,256</td>
<td>6,016</td>
<td>44</td>
<td>80</td>
</tr>
<tr>
<td>XC2VP7</td>
<td>40 x 34</td>
<td>4,928</td>
<td>9,856</td>
<td>157,696</td>
<td>9,856</td>
<td>68</td>
<td>80</td>
</tr>
<tr>
<td>XC2VP20</td>
<td>56 x 46</td>
<td>9,280</td>
<td>18,560</td>
<td>296,960</td>
<td>18,560</td>
<td>92</td>
<td>112</td>
</tr>
<tr>
<td>XC2VP30</td>
<td>80 x 46</td>
<td>13,696</td>
<td>27,392</td>
<td>438,272</td>
<td>27,392</td>
<td>92</td>
<td>160</td>
</tr>
<tr>
<td>XC2VP40</td>
<td>88 x 58</td>
<td>19,392</td>
<td>38,784</td>
<td>620,544</td>
<td>38,784</td>
<td>116</td>
<td>176</td>
</tr>
<tr>
<td>XC2VP50</td>
<td>88 x 70</td>
<td>23,616</td>
<td>47,232</td>
<td>755,712</td>
<td>47,232</td>
<td>140</td>
<td>176</td>
</tr>
<tr>
<td>XC2VP70</td>
<td>104 x 82</td>
<td>33,088</td>
<td>66,176</td>
<td>1,058,816</td>
<td>66,176</td>
<td>164</td>
<td>208</td>
</tr>
<tr>
<td>XC2VP100</td>
<td>120 x 94</td>
<td>44,096</td>
<td>88,192</td>
<td>1,411,072</td>
<td>88,192</td>
<td>188</td>
<td>240</td>
</tr>
<tr>
<td>XC2VP125</td>
<td>136 x 106</td>
<td>55,616</td>
<td>111,232</td>
<td>1,779,712</td>
<td>111,232</td>
<td>212</td>
<td>272</td>
</tr>
</tbody>
</table>

**Notes:**
1. The carry-chains and SOP chains can be split or cascaded.

---

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 diagram](image1)

![Timer/counter diagram](image2)
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

![Diagram of pulse width modulator waveforms]

- 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 */
}
Keypad controller

N=4, M=4
Summary

- RISC CPUs
  - ARM 7
- CISC CPUs
  - TI C54x
- VLIW
  - TI C6x
  - TriMedia
- FPGA – Programmable CPUs
  - Virtex II
- Single purpose processors
Sources and References