Some DSP Chip History

First Commercial DSP’s

- 1982 – NEC μPD7720
- 1982 – Texas Instruments TMS 32010

These chips initially cost around $600. DSP’s now cost from $2 to $200.

Texas Instruments (TI) DSP Families

- C2000 series
  32-bit fixed-point DSP’s
  real-time control applications including digital motor control and power conversion

- C5000 Ultra Low Power series
  16-bit fixed-point DSP’s
  Portable devices in audio, voice, communications, medical, security and industrial applications. Used cell phones.

- C6000 High Performance DSP’s
  32-bit fixed and floating-point DSP’s
  Audio, video, and imaging applications
• C6000 High Performance Multicore DSP’s
  Combine both fixed and floating-point capabilities
  Medical imaging, test and automation, video infrastructure, and high-end imaging.

• DaVinci Digital Video Processors
  Optimized for digital video systems. Digital audio, video, imaging, and vision applications. Includes a general purpose processor, video accelerators, an optional DSP, and related peripherals.

• Digital Signal Processor and Arm Microprocessor Platforms
  Include DSP’s from the TI families and ARM MPU processors in one chip.

• Custom high performance DSP’s
  Designed for special customers like manufacturers of 3G and 4G base stations. Include peripherals like FFT units and Turbo Code decoders.
Some Other DSP Manufacturers

LSI (Lucent, Agere), Freescale Semiconductor (Motorola), Analog Devices, Zilog

Fixed vs. Floating-Point DSP’s

• Fixed-point DSP’s are cheaper and use less power but care must be taken with scaling to avoid over and underflow.

• Floating-point DSP’s are easier to program. Numbers are automatically scaled. They are more complicated and expensive.

Advantages of DSP’s over Analog Circuits

• Can implement complex linear or nonlinear algorithms.

• Can modify easily by changing software.

• Reduced parts count makes fabrication easier.

• High reliability
DSP Applications

- **Telecommunications**: telephone line modems, FAX, cellular telephones, wireless networks, speaker phones, answering machines

- **Voice/Speech**: speech digitization and compression, voice mail, speaker verification, and speech synthesis

- **Automotive**: engine control, antilock brakes, active suspension, airbag control, and system diagnosis

- **Control Systems**: head positioning servo systems in disk drives, laser printer control, robot control, engine and motor control, and numerical control of automatic machine tools

- **Military**: radar and sonar signal processing, navigation systems, missile guidance, HF radio frequency modems, secure spread spectrum radios, and secure voice

- **Medical**: hearing aids, MRI imaging, ultrasound imaging, and patient monitoring

- **Instrumentation**: spectrum analysis, transient analysis, signal generators

- **Image Processing**: HDTV, image enhancement, image compression and transmission, 3-D rotation, and animation
TMS320C6713 DSP Starter Kit (DSK) Block Diagram

The C6713 DSK is a low-cost standalone development platform that enables users to evaluate and develop applications for the TI C67xx DSP family. The DSK also serves as a hardware reference design for the TMS320C6713 DSP. Schematics, logic equations and application notes are available to ease hardware development and reduce time to market.

The DSK comes with a full compliment of on-board devices that suit a wide variety of application environments. Key features include:

- A Texas Instruments TMS320C6713 DSP operating at 225 MHz.
- An AIC23 stereo codec
- 16 Mbytes of synchronous DRAM
- 512 Kbytes of non-volatile Flash memory (256 Kbytes usable in default configuration)
- 4 user accessible LEDs and DIP switches
- Software board configuration through registers implemented in CPLD
- Configurable boot options
- Standard expansion connectors for daughter card use
- JTAG emulation through on-board JTAG emulator with USB host interface or external emulator
- Single voltage power supply (+5V)

Figure 1-1, Block Diagram C6713 DSK
TMS320C6713 DSK Features

- A TMS320C6713 DSP operating at 225 MHz.
- An AIC23 stereo codec with Line In, Line Out, MIC, and headphone stereo jacks
- 16 Mbytes of synchronous DRAM
- 512 Kbytes of non-volatile Flash memory (256 Kbytes usable in default configuration)
- 4 user accessible LEDs and DIP switches
- Software board configuration through registers implemented in CPLD
- Configurable boot options
- Expansion connectors for daughter cards
- JTAG emulation through on-board JTAG emulator with USB host interface or external emulator
TMS320C6713, TMS320C6713B Floating-Point Digital Signal Processor,
Main 'C6713 Features

- VelociTI Very Long Instruction Word (VLIW) CPU Core
  Fetches eight 32-bit instructions at once
  - Eight Independent functional units
    * Four ALUs (fixed and floating-point)
    * Two ALUs (fixed-point)
    * Two multipliers (fixed and floating-point)
    32 × 32 bit integer multiply with 32 or 64-bit result
  - Load-store architecture with 32 32-bit general purpose registers

- Instruction Set Features
  - Hardware support for IEEE single and double precision floating-point operations
  - 8, 16, and 32-bit addressable
  - 8-bit overflow protection and saturation
  - Bit-field extract, set, clear; bit-counting; normalization
’C6713 Features (cont. 1)

• L1/L2 Memory Architecture
  – 4K-Byte L1P Program Cache (Direct-Mapped)
  – 4K-Byte L1D Data Cache (2-Way)
  – 256K-Byte L2 Memory Total; 64K-Byte L2 Unified Cache/Mapped RAM and 192K-Byte Additional L2 Mapped RAM

• Device Configuration
  – Boot Mode: HPI, 8-, 16-, 32-Bit ROM Boot
  – Little Endian and Big Endian

• 32-bit External Memory Interface (EMIF)
  – Glueless interface to SDRAM, Flash, SBSRAM, SRAM, and EPROM
  – 512M-byte Total Addressable External Memory Space
'C6713 Features (cont. 2)

- Enhanced Direct-Memory-Access (EDMA) Controller (16 Independent Channels)
- 16-Bit Host-Port Interface (HPI)
- Two Inter-Integrated Circuit Bus (I²C Bus) Multi-Master and Slave Interfaces
- Two Multichannel Audio Serial Ports (McASPs)
- Two Multichannel Buffered Serial Ports (McBSPs)
- Two 32-Bit General Purpose Timers
- Dedicated GPIO Module with 16 pins
- Flexible Phase-Locked-Loop (PLL) Based Clock Generator Module
- IEEE-1149.1 JTAG Boundary Scan
## Instructions Common to C62x and C67x

<table>
<thead>
<tr>
<th>.L unit</th>
<th>.M Unit</th>
<th>.S Unit</th>
<th>.D Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS</td>
<td>MPY</td>
<td>ADD</td>
<td>SET</td>
</tr>
<tr>
<td>ADD</td>
<td>MPYU</td>
<td>ADDK</td>
<td>SHL</td>
</tr>
<tr>
<td>ADDU</td>
<td>MPYUS</td>
<td>ADD2</td>
<td>SHR</td>
</tr>
<tr>
<td>AND</td>
<td>MPYSU</td>
<td>AND</td>
<td>SHRU</td>
</tr>
<tr>
<td>CMPEQ</td>
<td>MPYH</td>
<td>B disp</td>
<td>SS HL</td>
</tr>
<tr>
<td>CMPGT</td>
<td>MPYHU</td>
<td>B IRP¹</td>
<td>SUB</td>
</tr>
<tr>
<td>CMPGTU</td>
<td>MPYHUS</td>
<td>B NRP¹</td>
<td>SUBU</td>
</tr>
<tr>
<td>CMPLT</td>
<td>MPYHSU</td>
<td>B reg</td>
<td>SUB2</td>
</tr>
<tr>
<td>CMPLTU</td>
<td>MPYHL</td>
<td>CLR</td>
<td>XOR</td>
</tr>
<tr>
<td>LMBD</td>
<td>MPHLU</td>
<td>EXT</td>
<td>ZERO</td>
</tr>
<tr>
<td>MV</td>
<td>MPYHULS</td>
<td>EX TU</td>
<td></td>
</tr>
<tr>
<td>NEG</td>
<td>MPYHSLU</td>
<td>MV</td>
<td></td>
</tr>
<tr>
<td>NORM</td>
<td>MPYLH</td>
<td>MVC¹</td>
<td></td>
</tr>
<tr>
<td>NOT</td>
<td>MPYLHU</td>
<td>MVK</td>
<td></td>
</tr>
<tr>
<td>OR</td>
<td>MPYLHU</td>
<td>MVK</td>
<td></td>
</tr>
<tr>
<td>SADD</td>
<td>MPYLSHU</td>
<td>MVKLH</td>
<td></td>
</tr>
<tr>
<td>SAT</td>
<td>SMPY</td>
<td>NEG</td>
<td></td>
</tr>
<tr>
<td>SSUB</td>
<td>SMPYHL</td>
<td>NOT</td>
<td></td>
</tr>
<tr>
<td>SUB</td>
<td>SMPYLH</td>
<td>OR</td>
<td></td>
</tr>
<tr>
<td>SUBU</td>
<td>SMPYH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XOR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZERO</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

See *TMS320C6000 CPU and Instruction Set, Reference Guide*, SPRU189F for complete descriptions of instructions.
## Extra Instructions for the C67x

<table>
<thead>
<tr>
<th>.L unit</th>
<th>.M Unit</th>
<th>.S Unit</th>
<th>.D Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDDP</td>
<td>MPYDP</td>
<td>ABSDP</td>
<td>ADDAD</td>
</tr>
<tr>
<td>ADDSP</td>
<td>MPYI</td>
<td>ABSSP</td>
<td>LDDW</td>
</tr>
<tr>
<td>DPRINT</td>
<td>MPYID</td>
<td>CMPEQDP</td>
<td></td>
</tr>
<tr>
<td>DPSP</td>
<td>MPYSP</td>
<td>CMPEQSP</td>
<td></td>
</tr>
<tr>
<td>DPTRUNC</td>
<td></td>
<td>CMPGTDP</td>
<td></td>
</tr>
<tr>
<td>INTDP</td>
<td></td>
<td>CMPGTSP</td>
<td></td>
</tr>
<tr>
<td>INTDPU</td>
<td></td>
<td>CMPLTDP</td>
<td></td>
</tr>
<tr>
<td>INTSP</td>
<td></td>
<td>CMPLTSP</td>
<td></td>
</tr>
<tr>
<td>INTSPU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPINT</td>
<td></td>
<td>RCPDP</td>
<td></td>
</tr>
<tr>
<td>SPTRUNC</td>
<td></td>
<td>RCPSP</td>
<td></td>
</tr>
<tr>
<td>SUBDP</td>
<td></td>
<td>RSQRDP</td>
<td></td>
</tr>
<tr>
<td>SUBSP</td>
<td></td>
<td>RSQRSP</td>
<td></td>
</tr>
</tbody>
</table>

See *TMS320C6000 CPU and Instruction Set, Reference Guide*, SPRU189F for complete descriptions of instructions.
Addressing Modes

- Linear Addressing – with all registers
- Circular Addressing – with registers A4–A7 and B4–B7

Forms for Indirect Addresses

- Register Indirect
  - No Modification: \(*R\)
  - Preincrement of: \(*++R\)
  - Predecrement of: \(*--R\)
  - Postincrement of: \(*R++\)
  - Postdecrement of: \(*R--\)

- Register Relative
  - No Modification: \(*±R[ucst5]\)
  - Preincrement of: \(*++R[ucst5][ucst5]\)
  - Predecrement of: \(*--R[ucst5]\)
  - Postincrement of: \(*R++[ucst5]\)
  - Postdecrement of: \(*R--[ucst5]\)
Forms for Indirect Addresses (cont.)

- Register Relative with 15-bit Constant Offset
  No Modification  \(+B14/B15[ucst15]\)

- Base + Index
  No Modification  \(\pm R[\text{offsetR}]\)
  Preincrement of  \(\text{++}R[\text{offsetR}]\)
  Predecrement of  \(\text{--}R[\text{offsetR}]\)
  Postincrement of \(R++[\text{offsetR}]\)
  Postdecrement of \(R--[\text{offsetR}]\)

Notes:

\(ucst5 = 5\)-bit unsigned integer constant
\(ucst15 = 15\)-bit unsigned integer constant
\(R = \) base register
\(\text{offsetR} = \) index register

Example: LDW .D1 \(\text{++}A4[9]\), A1
Load a 32-bit word using functional unit D1 into register A1 from the memory byte address:

\(\text{contents of } (A4) + 4 \times 9\)
## TMS320C6713DSK Memory Map

<table>
<thead>
<tr>
<th>Address</th>
<th>C67x Family Memory Type</th>
<th>C6713DSK</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
<td>Internal Memory</td>
<td>Internal Memory</td>
</tr>
<tr>
<td>0x00030000</td>
<td>Reserved Space or Peripheral Regs</td>
<td>Reserved or Peripheral</td>
</tr>
<tr>
<td>0x80000000</td>
<td>EMIF CE0</td>
<td>SDRAM</td>
</tr>
<tr>
<td>0x90000000</td>
<td>EMIF CE1</td>
<td>Flash</td>
</tr>
<tr>
<td>0x90080000</td>
<td></td>
<td>CPLD</td>
</tr>
<tr>
<td>0xA0000000</td>
<td>EMIF CE2</td>
<td>Daughter Card</td>
</tr>
<tr>
<td>0xB0000000</td>
<td>EMIF CE3</td>
<td></td>
</tr>
</tbody>
</table>
Parallel Operations

- The instruction word for each functional unit is 32 bits long.

- Instructions are fetched 8 at a time consisting of $8 \times 32 = 256$ bits. The group is called a fetch packet. Fetch packets must start at an address that is a multiple of 8 32-bit words.

- Up to 8 instructions can be executed in parallel. Each must use a different functional unit. Each group of parallel instructions is called an execute packet.

- The $p$-bit (bit 0) determines if an instruction executes in parallel with another. The instructions are scanned from the lowest address to the highest. If the $p$-bit of instruction $i$ is 1, then instruction $i + 1$ is executed in parallel with instruction $i$. If it is 0, instruction $i + 1$ is executed one cycle after instruction $i$. 
### TMS320C6x Pipeline Phases

<table>
<thead>
<tr>
<th>Stage</th>
<th>Phase</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Fetch</td>
<td>Program Address Generation</td>
<td>PG</td>
</tr>
<tr>
<td></td>
<td>Program Address Sent</td>
<td>PS</td>
</tr>
<tr>
<td></td>
<td>Program Wait</td>
<td>PW</td>
</tr>
<tr>
<td></td>
<td>Program Data Receive</td>
<td>PR</td>
</tr>
<tr>
<td>Program Decode</td>
<td>Dispatch</td>
<td>DP</td>
</tr>
<tr>
<td></td>
<td>Decode</td>
<td>DC</td>
</tr>
<tr>
<td>Execute</td>
<td>Execute 1</td>
<td>E1</td>
</tr>
<tr>
<td></td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td></td>
<td>Execute 10</td>
<td>E10</td>
</tr>
</tbody>
</table>

See *TMS320C6000 CPU and Instruction Set Reference Guide*, SPRU189F, Table 7-1, pp. 7-7 to 7-9, for details of pipeline phases.
Pipeline Operation Assuming One Execute Packet per Fetch Packet

<table>
<thead>
<tr>
<th>Clock Cycle</th>
<th>Fetch Packet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$n$</td>
</tr>
<tr>
<td>1</td>
<td>PG</td>
</tr>
<tr>
<td>2</td>
<td>PS</td>
</tr>
<tr>
<td>3</td>
<td>PW</td>
</tr>
<tr>
<td>4</td>
<td>PR</td>
</tr>
<tr>
<td>5</td>
<td>DP</td>
</tr>
<tr>
<td>6</td>
<td>DC</td>
</tr>
<tr>
<td>7</td>
<td>E1</td>
</tr>
<tr>
<td>8</td>
<td>E2</td>
</tr>
<tr>
<td>9</td>
<td>E3</td>
</tr>
<tr>
<td>10</td>
<td>E4</td>
</tr>
<tr>
<td>11</td>
<td>E5</td>
</tr>
<tr>
<td>12</td>
<td>E6</td>
</tr>
<tr>
<td>13</td>
<td>E7</td>
</tr>
<tr>
<td>14</td>
<td>E8</td>
</tr>
<tr>
<td>15</td>
<td>E9</td>
</tr>
<tr>
<td>16</td>
<td>E10</td>
</tr>
<tr>
<td>17</td>
<td>E10</td>
</tr>
</tbody>
</table>

Need for NOP’s

- Different instruction types require from 1 to 10 execution phases. Therefore, NOP instructions must be added to make sure results of one instruction are needed by another.

- NOP’s can be added manually in hand coded assembly (hard), in linear assembly by the assembler (easier), or by the C compiler (easiest).
TI Software Tools

Code Composer Studio Version 5.4

• Built on Eclipse platform
• Create and edit source code
• Compile (cl6x.exe), assemble (asm6x.exe), and link (lnk6x.exe) programs using project 
  “.pjt” files. (Actually, cl6x.exe is a shell program that can compile, assemble and link.)
• Build libraries with ar6x.exe
• Include a real-time operating system, DSP/BIOS, in the DSP code with real-time data transfer (RTDX) between the PC and DSP
• Load programs into DSP, run programs, single step, break points, read memory and registers, profile running programs, etc.
Building Programs

TMS320C6000 Optimizing Compiler User’s Guide (SPRU187I, April 2001, Figure 1-1, p. 1-2)
Other Software

- Microsoft Visual C++
- MATLAB
- Freeware Digital Filter Design Programs
  - WINDOW.EXE
  - REMEZ.EXE
  - IIR.EXE
  - RASCOS.EXE
  - SQRTRACO.EXE
- Standard MS Windows Programs like MS Word and Excel
- SSH Terminal Program (PUTTY) and SSH File Transfer Program (WINSCP)
First Lab Session

No lab report is required for Chapter 1. The software utility you will use to generate and edit source code, build executable DSP programs, and load these programs into the ’C6713 DSK is called Code Composer Studio. Do the following introductory tasks for your first lab session to learn about the hardware and software tools.

1. Check out the hardware.
The DSK has been installed inside the PC case to keep it secure and allow you access to the lab outside of regular class hours. The important DSK connectors have been brought out to the side of the PC case. The DSK is connected to a USB port on the motherboard and the power supply has been brought out to an external plug.

Find the stereo connectors for the A/D and D/A converters on the case. Notice that the connectors are labeled MIC IN, LINE IN, LINE OUT, and HEADPHONE. The MIC IN input is for low voltage signals. For ENEE 428 you should use only the LINE IN and LINE OUT connectors.
2. Learn how to use Code Composer Studio.

Start Code Composer Studio (CCS) by double clicking on its icon. If the license screen appears select Free.

1. The “Workspace Launcher” window will appear.
   In the first lab session before shared ECE labs server files are set up, use the default workspace C:\Users \<your login name>\workspace_v5_5
   Once your shared file is set up, use it as your default workspace.
   Warning: The lab PC’s are re-imaged when rebooted and any files you store on the C drive will disappear.

2. The “Welcome to Code Composer Studio v5” window should appear. If it does not, click on “Help” then “Welcome to CCS.” Click on the right arrow in the “Getting Started with Code Composer Studio” box and watch the introductory video.
First Lab Session (cont. 2)

3. Click on “Help” and then “Help Contents.”

(a) Expand “Workbench User Guide” and then “Getting Started.”

i. Work through the “Basic Tutorial” following each “Contents” item in order.

ii. Skip the “Team CVS tutorial” which you probably will not use. You can return to this topic later if desired. It explains how groups can manage projects using a version control system.

Skip the rest of the “Workbench User Guide”.

(b) Expand “Code Composer Help.”

i. Expand “Getting Started Quickly.”

• Work through “Creating a New CCS Project.”
  Note that our current version of CCS differs somewhat from the help presentation.
  Experiment with the options.

• Go through the “Graphs Overview.”

• Skip the other topics in “Getting Started Quickly.”
First Lab Session (cont. 3)

ii. Expand “Views and Editors.”
   • Carefully work through the “Breakpoints View” topic. You will use breakpoints in some experiments for code debugging and measuring the execution load of parts of your programs.
   • Quickly browse through the remaining topics under “Views and Editors.”

(c) Do this Classical “Hello World” First C Project
   i. Start Code Composer Studio, Click on “Project” and then “New CCS Project”
   ii. Enter “HelloWorld” for the “Project Name:”
   iii. Enter “C6000” for “Family:”
   iv. In the right “Variant” box select “DSK6713”
   v. Open the “Connection:” box menu and select “Spectrum Digital DSK-EVM eZdsp onboard USB Emulator.”
   vi. Under “Basic Examples” select “Hello World”
   vii. The C code should appear in the right panel. Replace the line “return(0);” with the line “for(;;);,” which is an infinite loop.
First Lab Session (cont. 4)

viii. Build the project by clicking on the green bug icon and run the program. You should see “Hello World!” in the bottom of the CCS window.

(d) Explore the other “Help” topics if you finish all the above items before the end of your first lab session. You can also browse through the TI manuals for the TMS320C6713 DSP CPU and peripherals.

(e) Experiment with capturing screen images and channel data from the Agilent oscilloscope to the PC. See the explanation of how to do this on the class web site after Chapter 1.

You should stay the entire time for each three hour lab period. If you finish a current experiment, go on to the next one.
Hardware and Software References

TI documents for the hardware and software tools are all available online at www.ti.com. Use the TI search engine to find the particular part or document. Enter a document number like “SPRU189” shown in the first item below in the TI search box. You will probably find more up-to-date versions of the documents than the ones listed below. In particular, the following documents will be very useful and are also available locally in the folder C:\c6713dsk\docs\C6713 DSP User Guides:


Hardware and Software References for the C6713 DSK

For detailed information about the c6713 DSK hardware, see the file on the local C drive:

C:\c6713dsk\docs\6713_dsk_techref.pdf

For additional information about the c6713 DSK hardware and documentation on how to use the Board Support Library and the functions it includes, double click on the file on the local C drive:

C:\c6713dsk\docs\help\c6713dsk.hlp