INTRODUCTION TO THE TMS320C6x VLIW DSP

Prof. Brian L. Evans

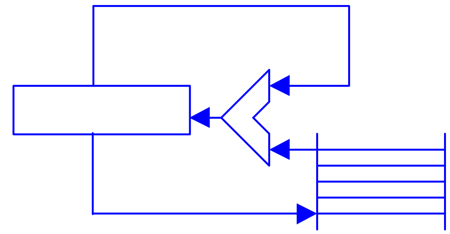
in collaboration with

Niranjan Damera-Venkata and
Magesh Valliappan

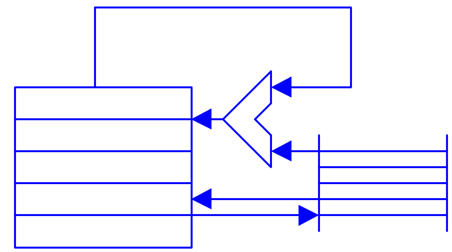
Embedded Signal Processing Laboratory
The University of Texas at Austin
Austin, TX 78712-1084

http://signal.ece.utexas.edu/

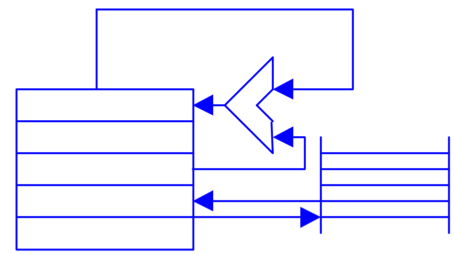
Accumulator architecture



Memory-register architecture



Load-store architecture

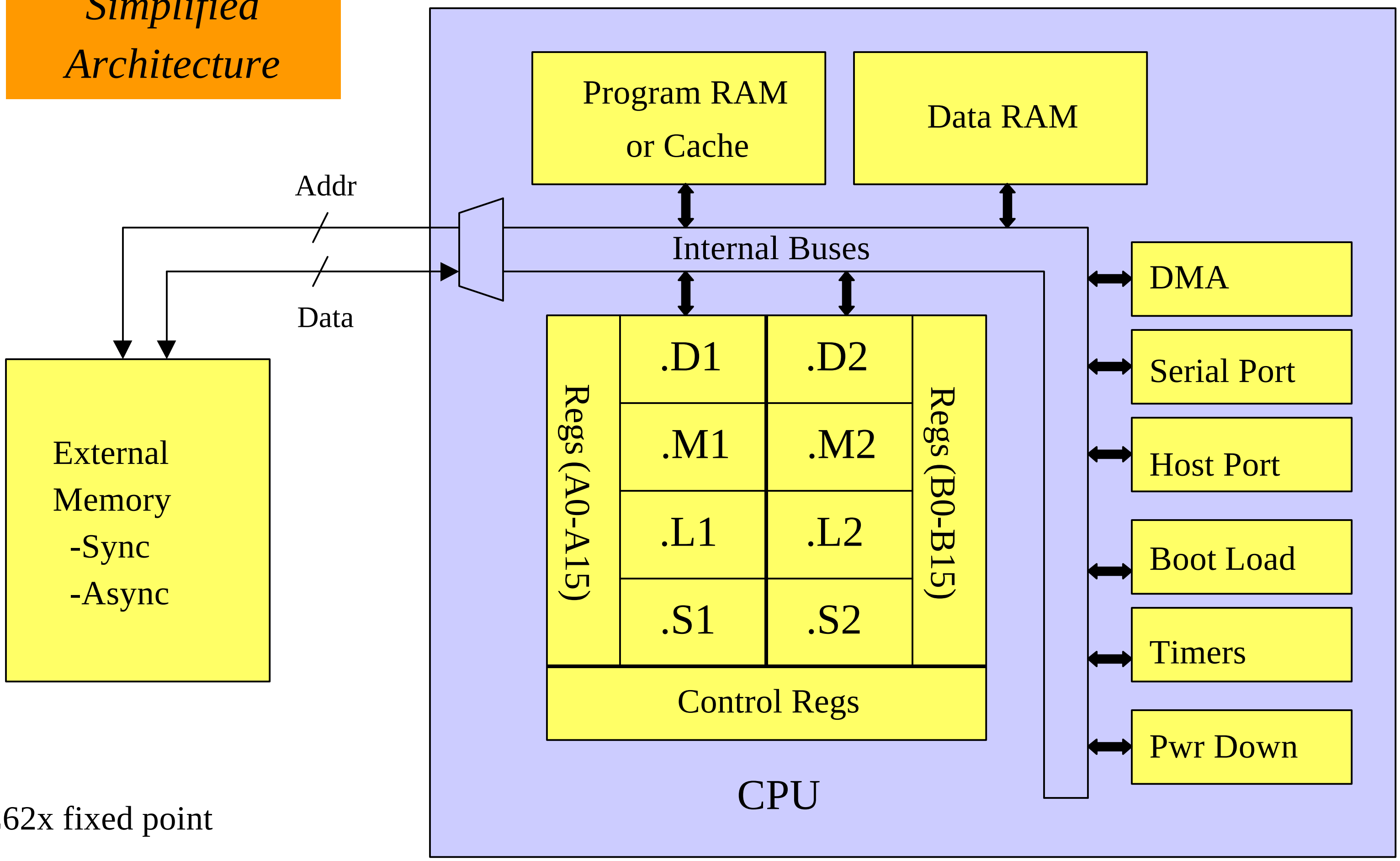


Outline

- Instruction set architecture
- Vector dot product example
- Pipelining
- Vector dot product example revisited
- Comparisons with other processors
- Conclusion

Instruction Set Architecture

Simplified Architecture



C62x fixed point

C67x floating point

Instruction Set Architecture

- Address 8/16/32 bit data + 64 bit data on C67x
- Load-store RISC architecture with 2 data paths
  - 16 32-bit registers per data path (A0-15 and B0-15)
  - 48 instructions (C62x) and 79 instructions (C67x)
- Two parallel data paths with 32-bit RISC units
  - **D**ata unit - 32-bit address calculations (modulo, linear)
  - **M**ultiplier unit - 16 bit x 16 bit with 32-bit result
  - **L**ogical unit - 40-bit (saturation) arithmetic & compares
  - **S**hifter unit - 32-bit integer ALU and 40-bit shifter
  - Conditionally executed based on registers A1-2 & B0-2
  - Work with two 16-bit halfwords packed into 32 bits

Functional Units

- .M multiplication unit
  - 16 bit x 16 bit signed/unsigned packed/unpacked
- .L arithmetic logic unit
  - Comparisons and logic operations (and, or, and xor)
  - Saturation arithmetic and absolute value
- .S shifter unit
  - Bit manipulation (set, get, shift, rotate) and branching
  - Addition and packed addition
- .D data unit
  - Load/store to memory
  - Addition and pointer arithmetic

Restrictions on Register Accesses

- Each function unit has read/write ports
  - Data path 1 (2) units read/write A (B) registers
  - Data path 2 (1) can read one A (B) register per cycle
- 40 bit words stored in adjacent even/odd registers
  - Used in extended precision accumulation
  - One 40-bit result can be written per cycle
  - A 40-bit read cannot occur in same cycle as 40-bit write
- Two simultaneous memory accesses cannot use registers of same register file as address pointers
- No more than four reads per register per cycle

Disadvantages

- No acceleration for variable length decoding
  - 50% of computation for MPEG-2 decoding on C6x in C
- Deep pipeline
  - If a branch is in the pipeline, interrupts are disabled: *avoid branches by using conditional execution*
  - No hardware protection against pipeline hazards: *programmer and software tools must guard against it*
- No hardware looping or bit-reversed addressing
  - Must emulate in software
- 40-bit accumulation incurs performance penalty
- No status register: must emulate status bits other than saturation bit (.L unit)

TMS320C62x Fixed-Point Processors

Processor	MHz	MIPS	Data (kbits)	Program (kbits)	Price	Applications
C6211	150 167	1200 1336	32 (512 kbit L2 cache)	32	$25	
C6201	167 200	1336 1600	512	512	$152 $159	EVM board
C6202	200 250	1600 2000	1000	2000	$167 $184	
C6203	250 300	2000 2400	4000	3000	n/a n/a	3G basestations modem banks

Unit price is for 100 - 999 units. N/a means not in production until 4Q99.
In volumes of 10,000, the 200 MHz C6201 is $96 per unit.

For more information: http://www.ti.com/sc/c62xdsps/

Example: Vector Dot Product

- A vector dot product is common in filtering

$$Y = \sum_{n=1}^{N} a(n)\ x(n)$$

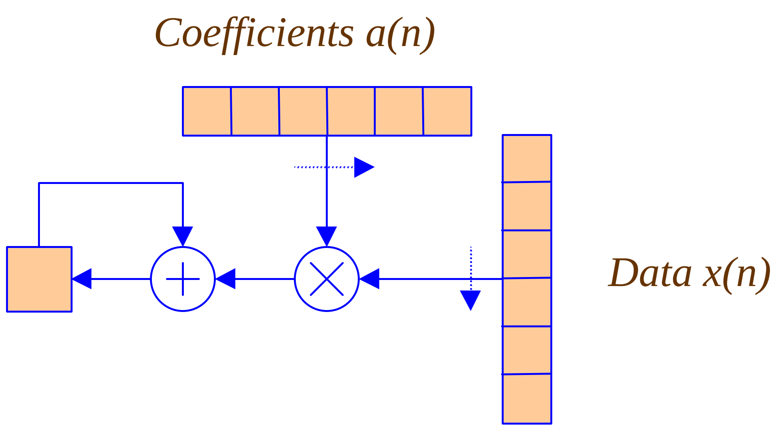
- Store $a(n)$ and $x(n)$ into an array of N elements
- C6x peak performance: 8 RISC instructions/cycle
  - Peak RISC instructions per sample: 300,000 for speech; 54,421 for audio; and 290 for luminance NTSC video
  - Generally requires hand coding for peak performance
- First dot product example will not be optimized

Example: Vector Dot Product

- Prologue
  - Initialize pointers: A5 for $a(n)$, A6 for $x(n)$, and A7 for Y
  - Move the number of times to loop (N) into A2
  - Set accumulator (A4) to zero
- Inner loop
  - Put $a(n)$ into A0 and $x(n)$ into A1
  - Multiply $a(n)$ and $x(n)$
  - Accumulate multiplication result into A4
  - Decrement loop counter (A2)
  - Continue inner loop if counter is not zero
- Epilogue
  - Store the result into Y

Reg	Meaning
A0	$a(n)$
A1	$x(n)$
A2	$N - n$
A3	$a(n)\ x(n)$
A4	Y
A5	$\&a$
A6	$\&x$
A7	$\&Y$

Example: Vector Dot Product



Using A data path only

A0	$a(n)$
A1	$x(n)$
A2	$N - n$
A3	$a(n)\,x(n)$
A4	Y
A5	$\&a$
A6	$\&x$
A7	$\&Y$

```
; clear A4 and initialize pointers A5, A6, and A7
        MVK   .S1   40,A2        ; A2 = 40 (loop counter)
loop    LDH   .D1   *A5++,A0     ; A0 = a(n)
        LDH   .D1   *A6++,A1     ; A1 = x(n)
        MPY   .M1   A0,A1,A3     ; A3 = a(n) * x(n)
        ADD   .L1   A3,A4,A4     ; Y = Y + A3
        SUB   .L1   A2,1,A2      ; decrement loop counter
[A2]    B     .S1   loop         ; if A2 != 0, then branch
        STH   .D1   A4,*A7       ; *A7 = Y
```

Example: Vector Dot Product

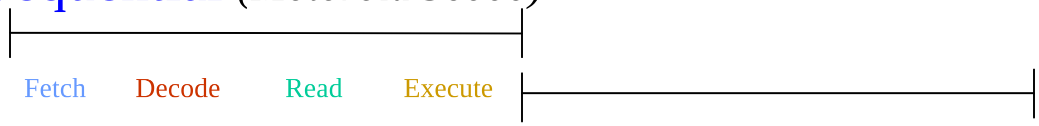
- MoVeKonstant
  - MVK .S 40,A2 ; A2 = 40
  - Lower 16 bits of A2 are loaded
- Conditional branch
  - [condition] B .S loop
  - [A2] means to execute the instruction if A2 != 0
  - Only A1, A2, B0, B1, and B2 can be used
- Loading registers
  - LDH .D *A5, A0 ;Loads half-word into A0 from memory
- Registers may be used as pointers (*A1++)

Pipelining

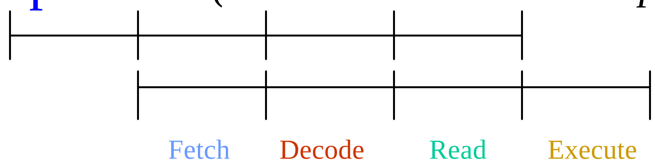
- CPU operations
  - **F**etch instruction from memory (DSP program memory)
  - **D**ecode instruction
  - **E**xecute instruction including reading data values
- Overlap operations to increase performance
  - Pipeline CPU operations to increase clock speed over a sequential implementation
  - Separate parallel functional units
  - Peripheral interfaces for I/O do not burden CPU

Pipelining

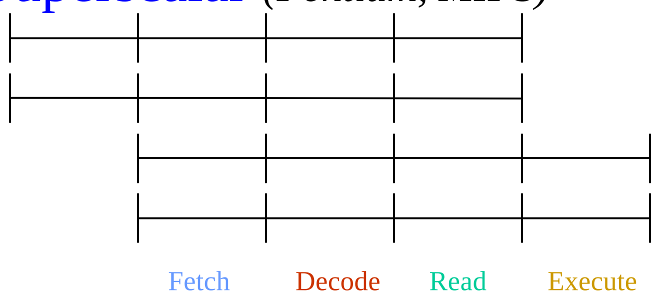
Sequential *(Motorola 56000)*



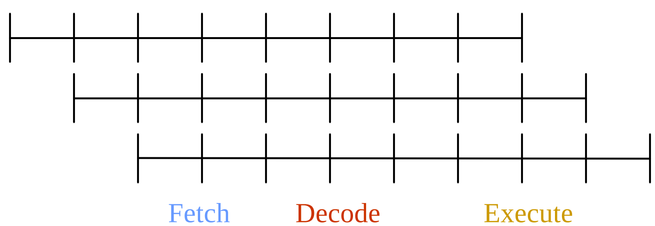
Pipelined *(Most conventional DSP processors)*



Superscalar *(Pentium, MIPS)*



Superpipelined *(TMS320C6x)*



Managing Pipelines

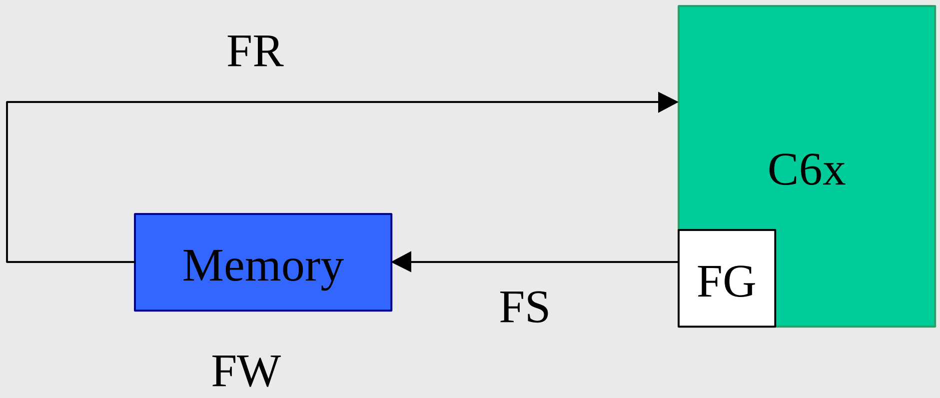
- compiler or programmer (TMS320C6x)
- pipeline interlocking in processor (TMS320C30)
- hardware instruction scheduling

TMS320C6x Pipeline

- One instruction cycle every clock cycle
- Deep pipeline
  - 7-11 stages in C62x: fetch 4, decode 2, execute 1-5
  - 7-16 stages in C67x: fetch 4, decode 2, execute 1-10
  - If a branch is in the pipeline, interrupts are disabled
  - Avoid branches by using conditional execution
- No hardware protection against pipeline hazards
  - Compiler and assembler must prevent pipeline hazards
- Dispatches instructions in packets

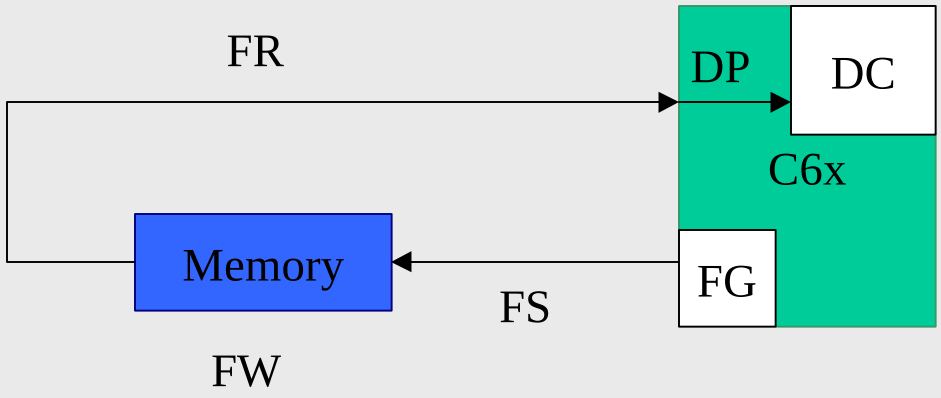
Program Fetch (F)

- Program fetching consists of 4 phases
  - generate fetch address (FG)
  - send address to memory (FS)
  - wait for data ready (FW)
  - read opcode (FR)
- Fetch packet consists of 8 32-bit instructions



Decode Stage (D)

- Decode stage consists of two phases
  - dispatch instruction to functional unit (DP)
  - instruction decoded at functional unit (DC)



Execute Stage (E)

Type	***Description***	***# Instr***	***Delay***
ISC	Single cycle	38	0
IMPY	Multiply	2	1
LDx	Load	3	4
B	Branch	1	5

Execute stage (E)

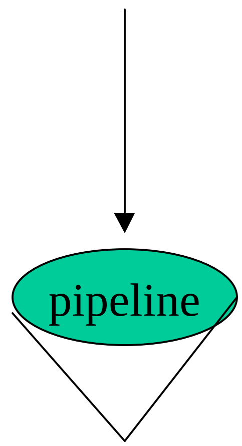
Execute Phase	***Description***
E1	ISC instructions completed
E2	IMPY instructions completed
E3	
E4	
E5	Load value into register
E6	Branch to destination complete

Vector Dot Product with Pipeline Effects

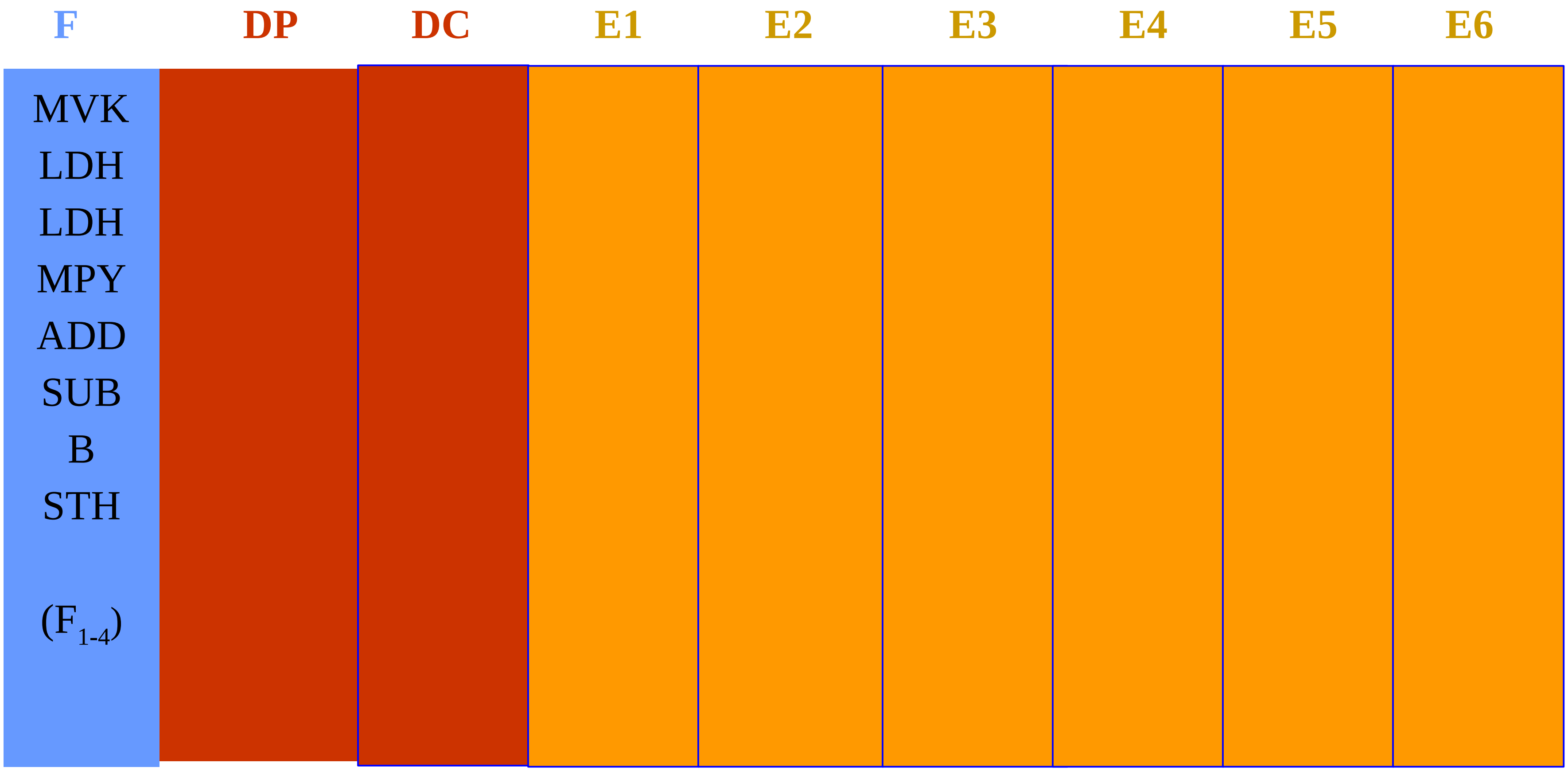
```
; clear A4 and initialize pointers A5, A6, and A7
        MVK   .S1   40,A2       ; A2 = 40 (loop counter)
loop    LDH   .D1   *A5++,A0    ; A0 = a(n)
        LDH   .D1   *A6++,A1    ; A1 = x(n)
        MPY   .M1   A0,A1,A3    ; A3 = a(n) * x(n)
        ADD   .L1   A3,A4,A4    ; Y = Y + A3
        SUB   .L1   A2,1,A2     ; decrement loop counter
[A2]    B     .S1   loop        ; if A2 != 0, then branch
        STH   .D1   A4,*A7      ; *A7 = Y
```

Multiplication has a
delay of 1 cycle

Load has a
delay of four cycles

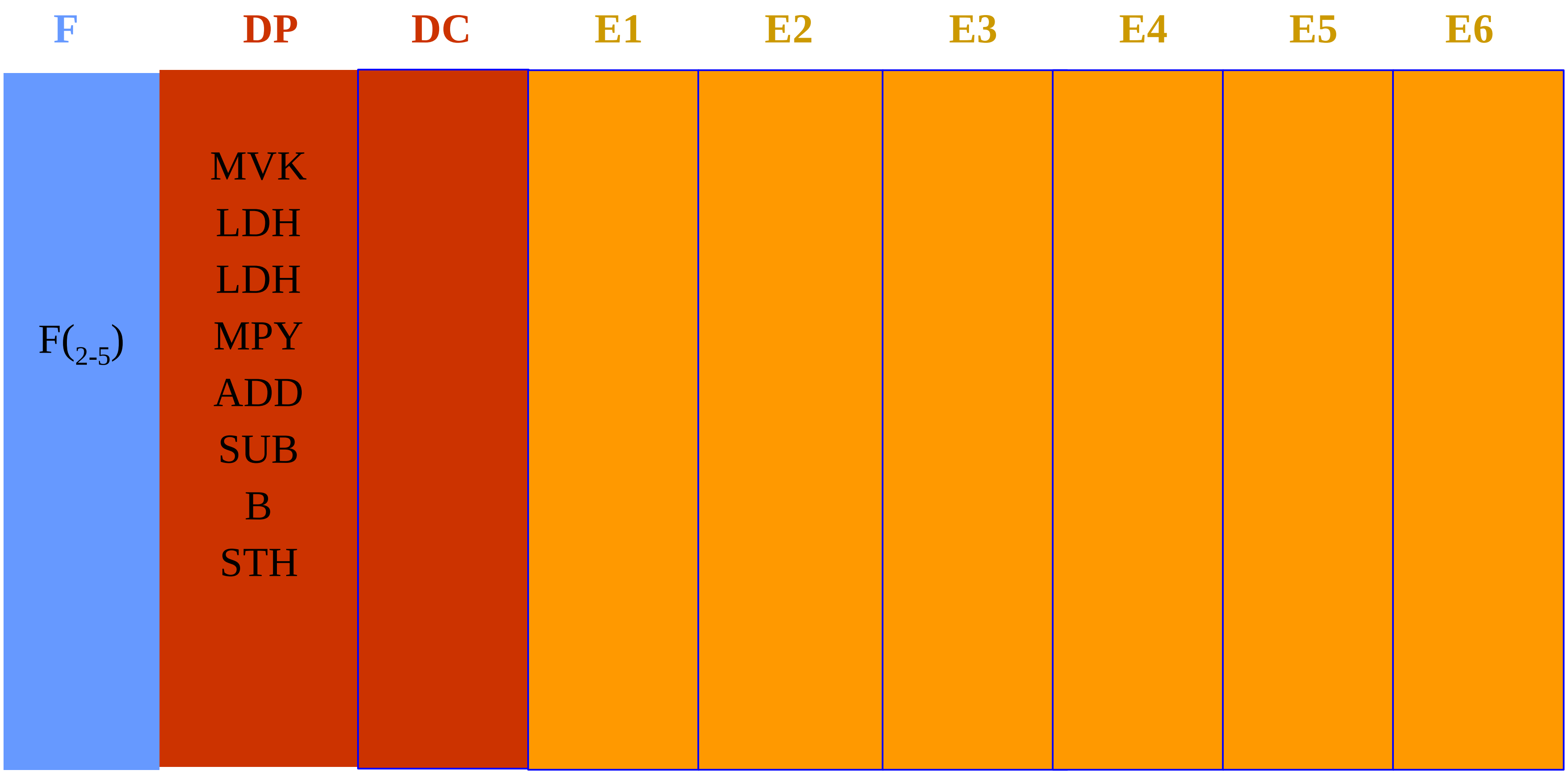


Fetch packet



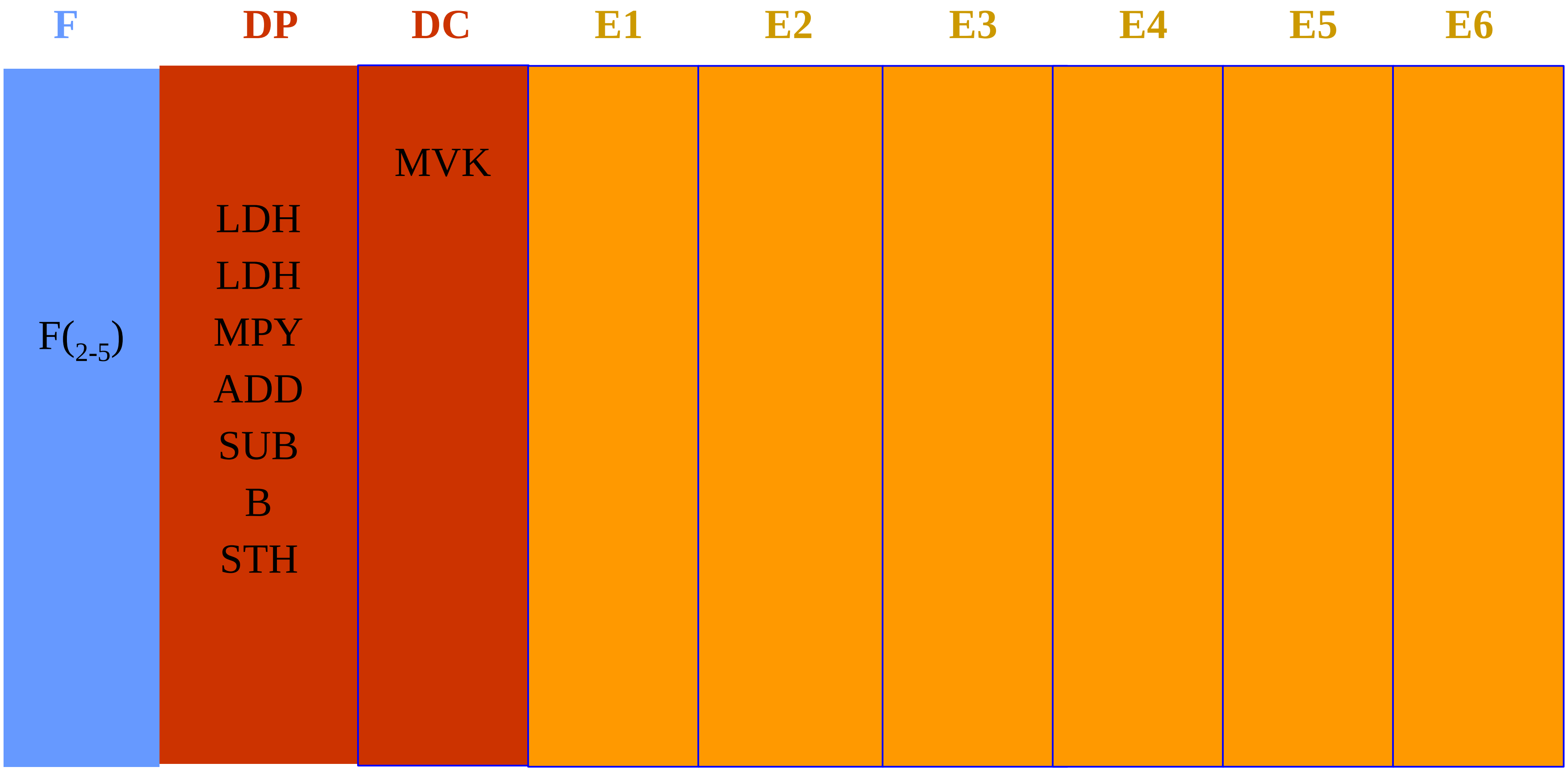
Time (t) = 4 clock cycles

Dispatch



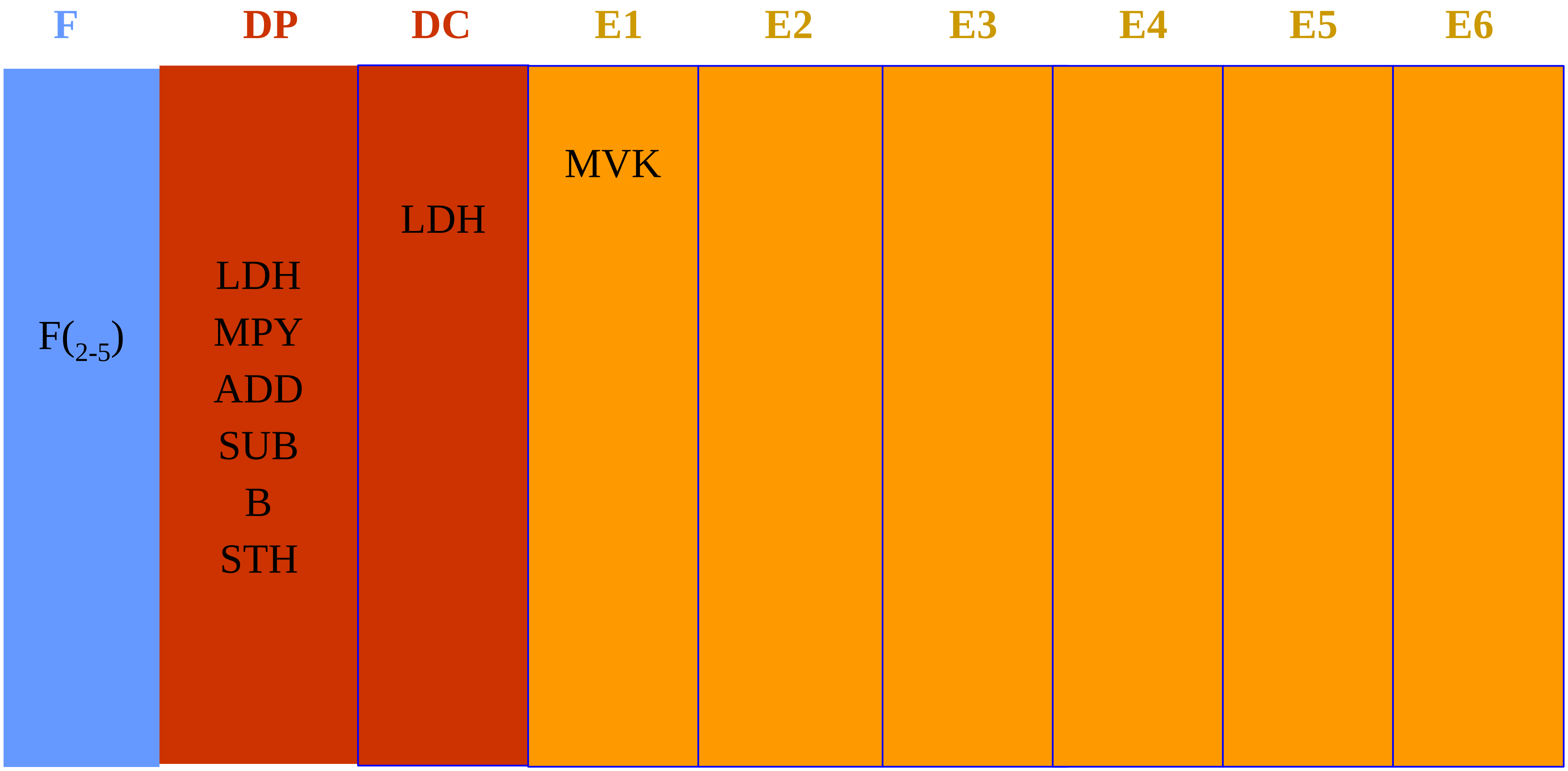
Time (t) = 5 clock cycles

Decode



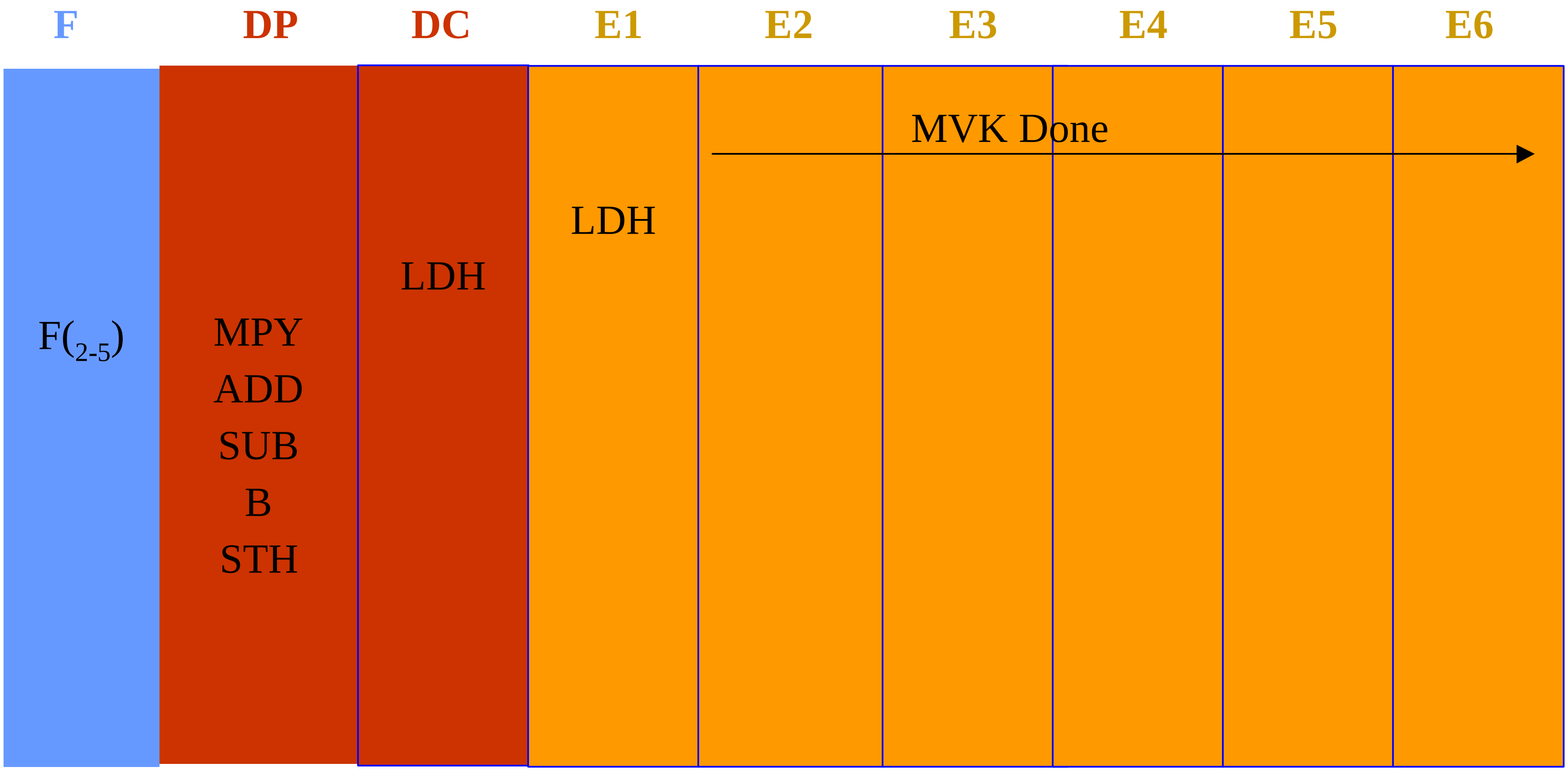
Time (t) = 6 clock cycles

Execute (E1)



Time (t) = 7 clock cycles

Execute (MVK done LDH in E1)



Time (t) = 8 clock cycles

Vector Dot Product with Pipeline Effects

```
; clear A4 and initialize pointers A5, A6, and A7
        MVK   .S1   40,A2       ; A2 = 40 (loop counter)
loop    LDH   .D1   *A5++,A0    ; A0 = a(n)
        LDH   .D1   *A6++,A1    ; A1 = x(n)
        NOP   4
        MPY   .M1   A0,A1,A3    ; A3 = a(n) * x(n)
        NOP
        ADD   .L1   A3,A4,A4    ; Y = Y + A3
        SUB   .L1   A2,1,A2     ; decrement loop counter
[A2]    B     .S1   loop        ; if A2 != 0, then branch
        NOP   5
        STH   .D1   A4,*A7      ; *A7 = Y
```

Assembler will automatically insert NOP instructions

Assembler can also make sequential code parallel

Optimized Vector Dot Product

```
; clear A4 and initialize pointers A5, A6, and A7
        MVK   .S1   40,A2        ; A2 = 40 (loop counter)
loop    LDW   .D1   *A5++,A0     ; load a(n) and a(n+1)
        LDW   .D2   *B6++,B1     ; load x(n) and x(n+1)
        MPY   .M1X  A0,B1,A3     ; A3 = a(n) * x(n)
        MPYH  .M2X  A0,B1,B3     ; B3 = a(n+1) * x(n+1)
        ADD   .L1   A3,A4,A4     ; Yeven = Yeven + A3
        ADD   .L2   B3,B4,B4     ; Yodd = Yodd + A3
        SUB   .S1   A2,1,A2      ; decrement loop counter
[A2]    B     .S2   loop         ; if A2 != 0, then branch
        ADD   .L1   A4,B4,A4     ; Y = Yodd + Yeven
        STH   .D1   A4,*A7       ; *A7 = Y
```

Retime summation

-- compute odd/even indexed terms at same time

-- utilize all eight functional units in the loop

-- put the sequential instructions in parallel

TMS320C6x vs. Pentium MMX

Processor	Peak MIPS	BDTI marks	ISR latency	Power	Unit Price	Area	Volume
Pentium MMX 233	466	49	1.14 μs	4.25 W	$213	5.5" x 2.5"	8.789 in^3
Pentium MMX 266	532	56	1.00 μs	4.85 W	$348	5.5" x 2.5"	8.789 in^3
C62x 150 MHz	1200	74	0.12 μs	1.45 W	$25	1.3" x 1.3"	0.118 in^3
C62x 200 MHz	1600	99	0.09 μs	1.94 W	$96	1.3" x 1.3"	0.118 in^3

BDTImarks: Berkeley Design Technology Inc. DSP benchmark results (larger means better) http://www.bdti.com/bdtimark/results.htm

http://www.ece.utexas.edu/~bevans/courses/ee382c/lectures/processors.html

TMS320C62x vs. StarCore S140

Feature	***C62x***	***S140***
Functional Units	8	16
multipliers	2	4
adders	6	4
other	--	8
Instructions/cycle	8	6 + branch
RISC instructions *	8	11
conditionals	8	2
Instruction width (bits)	256	128
Total instructions	48	180
Number of registers	32	51
Register size (bits)	32	40
Accumulation precision (bits) **	32 or 40	40
Pipeline depth (cycle)	7-11	5

* Does not count equivalent RISC operations for modulo addressing

** On the C62x, there is a performance penalty for 40-bit accumulation

Conclusion

- Conventional digital signal processors
  - High performance vs. power consumption/cost/volume
  - Excel at one-dimensional processing
  - Have instructions tailored to specific applications
- TMS320C6x VLIW DSP
  - High performance vs. cost/volume
  - Excel at multidimensional signal processing
  - A maximum of 8 RISC instructions per cycle

Conclusion

- Web resources
  - comp.dsp newsgroup: FAQ www.bdti.com/faq/dsp_faq.html
  - embedded processors and systems: www.eg3.com
  - on-line courses and DSP boards: www.techonline.com
- References

  - R. Bhargava, R. Radhakrishnan, B. L. Evans, and L. K. John, "Evaluating MMX Technology Using DSP and Multimedia Applications," *Proc. IEEE Sym. Microarchitecture*, pp. 37-46, 1998. http://www.ece.utexas.edu/~ravib/mmxdsp/
  - B. L. Evans, "EE379K-17 Real-Time DSP Laboratory," UT Austin. http://www.ece.utexas.edu/~bevans/courses/realtime/
  - B. L. Evans, "EE382C Embedded Software Systems," UT Austin. http://www.ece.utexas.edu/~bevans/courses/ee382c/