Texas Instruments
TMS 32C5x DSP

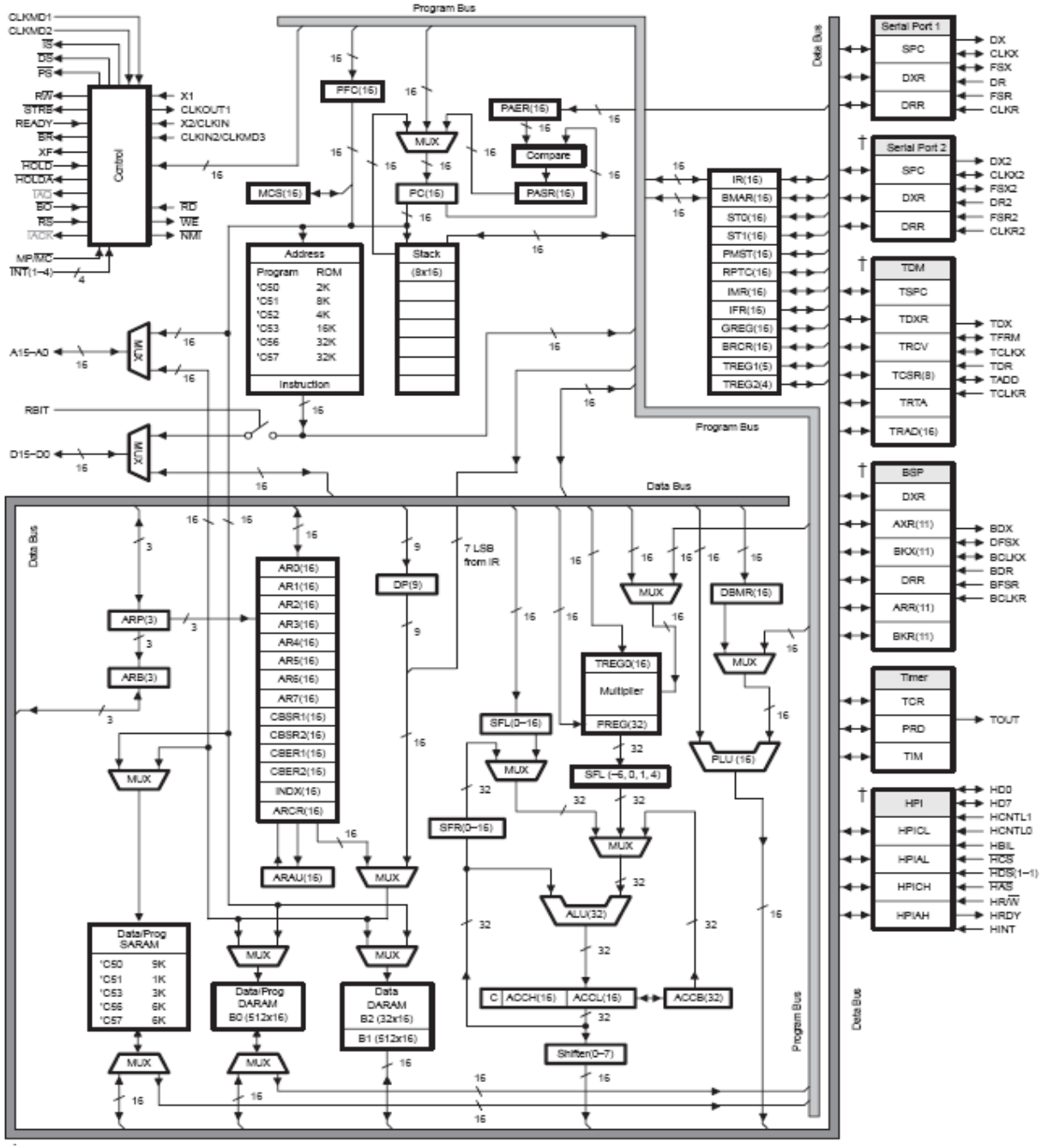
Characteristics of TI ‘C5x DSPs

TMS320 DEVICES	ON-CHIP MEMORY (16-BIT WORDS)				I/O PORTS		POWER SUPPLY (V)	CYCLE TIME (ns)	PACKAGE TYPE QFP‡
	DARAM		SARAM	ROM					
	DATA	DATA + PROG	DATA + PROG	PROG	SERIAL	PARALLEL†			
TMS320C50	544	512	9K	2K§	2	64K	5	50/35/25	132 pin
TMS320LC50	544	512	9K	2K§	2	64K	3.3	50/40/25	132 pin
TMS320C51	544	512	1K	8K§	2	64K	5	50/35/25/20	100/132 pin
TMS320LC51	544	512	1K	8K§	2	64K	3.3	50/40/25	100/132 pin
TMS320C52	544	512	–	4K§	1¶	64K	5	50/35/25/20	100 pin
TMS320LC52	544	512	–	4K§	1¶	64K	3.3	50/40/25	100 pin
TMS320C53	544	512	3K	16K§	2	64K	5	50/35/25	132 pin
TMS320LC53	544	512	3K	16K§	2	64K	3.3	50/40/25	132 pin
TMS320C53S	544	512	3K	16K§	2¶	64K	5	50/35/25	100 pin
TMS320LC53S	544	512	3K	16K§	2¶	64K	3.3	50/40/25	100 pin
TMS320LC56	544	512	6K	32K	2#	64K	3.3	35/25	100 pin
TMS320LC57	544	512	6K	32K	2#	64K + HPI‖	3.3	35/25	128 pin
TMS320C57S	544	512	6K	2K§	2#	64K + HPI‖	5	50/35/25	144 pin
TMS320LC57S	544	512	6K	2K§	2#	64K + HPI‖	3.3	50/35	144 pin

TI ‘C5x DSP CPU

Instruction Fetch →

Data Processing →



Key DSP Instructions

- MAC *pma,dma* – multiply/accumulate

  ACC + P -> ACC

  pma x dma -> P

  (repeatable with pma+1)

- MACD *pma,dma* -multiply/accumulate/delay

  ACC + P -> ACC

  pma x dma -> P

  dma -> dma+1 (move data for delay)

  (repeatable with pma+1)

Other key instructions

- T register used with multiplier

- LT dma/ind : mem ->TREG0
- LTA dma/ind : load T, ACC=ACC+P
- LTD dma/ind : load T, ACC=ACC+P,move data
- LTP dma/ind : load T, P->ACC

Low-Pass Filter

$$H(z)=\frac{Y(z)}{X(z)}=\frac{(b-1)z}{z-b}=\frac{(b-1)}{1-bz^{-1}}$$

$$Y(z)=bz^{-1}Y(z)+(1-b)X(z)$$

$$y(n)=by(n-1)+(1-b)x(n)$$

```
IN Xn,PA0 ;read x(n)
ZAP       ;A=P=0
MAC B,Yn
MAC C,Xn
APAC      ;A=A+P
SACHYn    ;save LSB
OUTYn,PA1 ;out y(n)
```

Finite Impulse Response (FIR) Filter

$$H(z) = \frac{Y(z)}{X(z)} = c_0 + c_1 z^{-1} + c_2 z^{-2}$$

$$y(n) = c_0 x(n) + c_1 x(n-1) + c_2 x(n-2)$$

Prog. Mem	Data Mem.
C0	X(n)
C1	X(n-1)
C2	X(n-2)

```
LARP AR3        ;AR3 active pointer
LAR AR3,#X0     ;point to X(n)
ZAP             ;A=P=0
MAC C0,*-       ;c0*x(n)
MAC C1,*-       ;+c1*x(n-1)
MAC C2,*-       ;+c2*x(n-2)
APAC            ;final A=A+P
SACHY           ;save LSB
```

PID Controller

$$a(n) = a(n-1) + c_0 e(n) + c_1 e(n-1) + c_2 e(n-2)$$

a(n)=control action
e(n) = nth error
c0,c1,c2 constants

Prog. Mem	Data Mem.
C0	e(n)
C1	e(n-1)
C2	e(n-2)

```
IN E0,PA0        ;new error e(n)
LARP AR3         ;AR3 active pointer
LAR AR3,#X0      ;point to e(n-2)
ZAP              ;A=P=0
MAC   c2,*-      ;c2*e(n-2)
MACD c1,*-       ;+c1*e(n-1), move e
MACD c0,*        ;+c0*e(n), move e
APAC             ;final A=A+P
SACH An          ;save LSB
OUT An,PA1
```

Basic sum of products

$$Y = \sum_{k=0}^{N} c_k x(k)$$

Prog. Mem	Data Mem.
C0	X(0)
C1	X(1)
C2	X(2)
….	
CN	X(N)

```
LARP AR3          ;AR3 active pointer
LAR AR3,#X0       ;point to X0
ZAP               ;A=P=0
RPT #N
MAC C0,*+         ;sum of products
APAC              ;final A=A+P
SACHY             ;save LSB
```

Basic sum of products

$$Y = \sum_{k=0}^{N} x(k)y(k)$$

B0 RAM	B1 RAM
x(0)	y(0)
x(1)	y(1)
....	
x(N)	y(N)

```
CNFP          ;B0 prog memory
LARP AR3      ;AR3 active pointer
LAR AR3,#Y    ;point to y(0)
ZAP           ;A=P=0
RPT #N
MAC X0,*+     ;sum of products
APAC          ;final A=A+P
SACHY         ;save LSB
CNFD          ;B0 back to data memory
```