Chapter 18: Programmable DSPs

Keshab K. Parhi and Viktor Owall

DSP Applications

DSP applications are often real time but with a wide variety of sample rates

- **High rates**
  - **Radar**
  - **Video**
- **Medium rates**
  - **Audio**
  - **Speech**
- **Low rates**
  - **Weather**
  - **Finance**

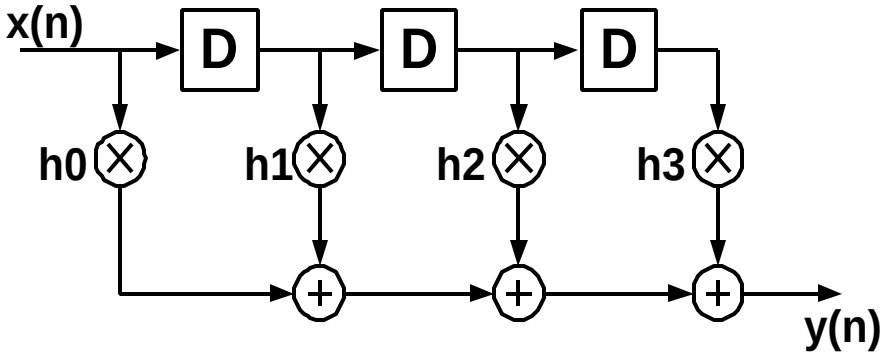
...with different demands on

- **numeric representation**
  - **float or fixed**
  - **and nmber of bits**
- **Throughput/speed**
- **Power/energy dissipation**
- **Cost**

DSP features

Fast Multiply/Accumulate (MAC)

- **FIR**
- **FFT**
- **etc.**



- **Multiple Access Memories**
- **Specialized addressing modes**
- **Specialized execution control (loops)**
- **Specialized interfaces, e.g. AD/DA**

Addressing Modes

- **Implied addressing**
  P=X*Y; operation sets location
- **Immediate data**
  AX0=1234
- **Memory direct**
  R1=Mem[101]
- **Register direct**
  sub R1, R2
- **Register indirect**
  A0=A0+ *R5
- **Register indirect with increment/decrement**
  A0=A0+ *R5++
  A0=A0+ *R5--

Standard DSP Alternatives

PCs or Workstations

- **Non-real time**
- **low requirements**

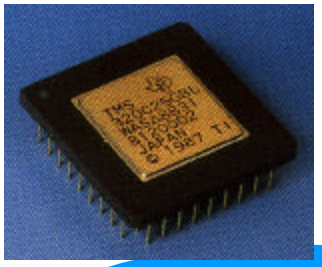
General purpose microprocessors

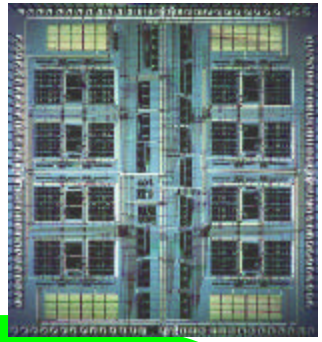
- **slower for DSP applications**
- **might be one mproc. there anyway**

Custom

- **perfomance**
- **low cost at volume**
- **High development cost**

Standard Processors vs. Special Purpose

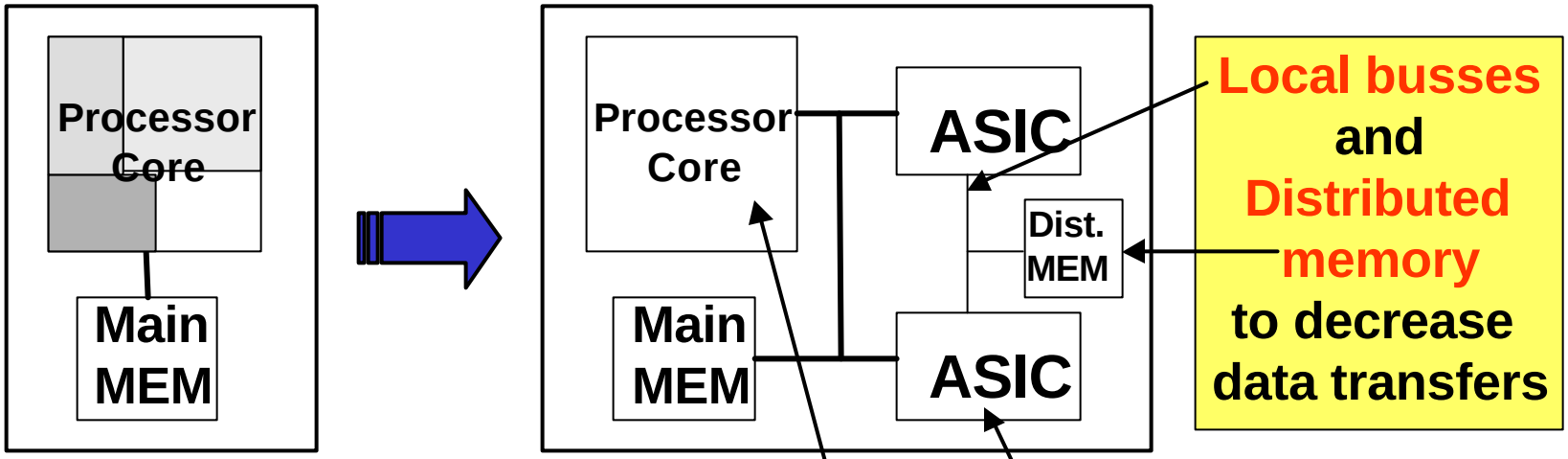




- Programmable
- Low Design cost
- Standard Interface
- Good supply of tools

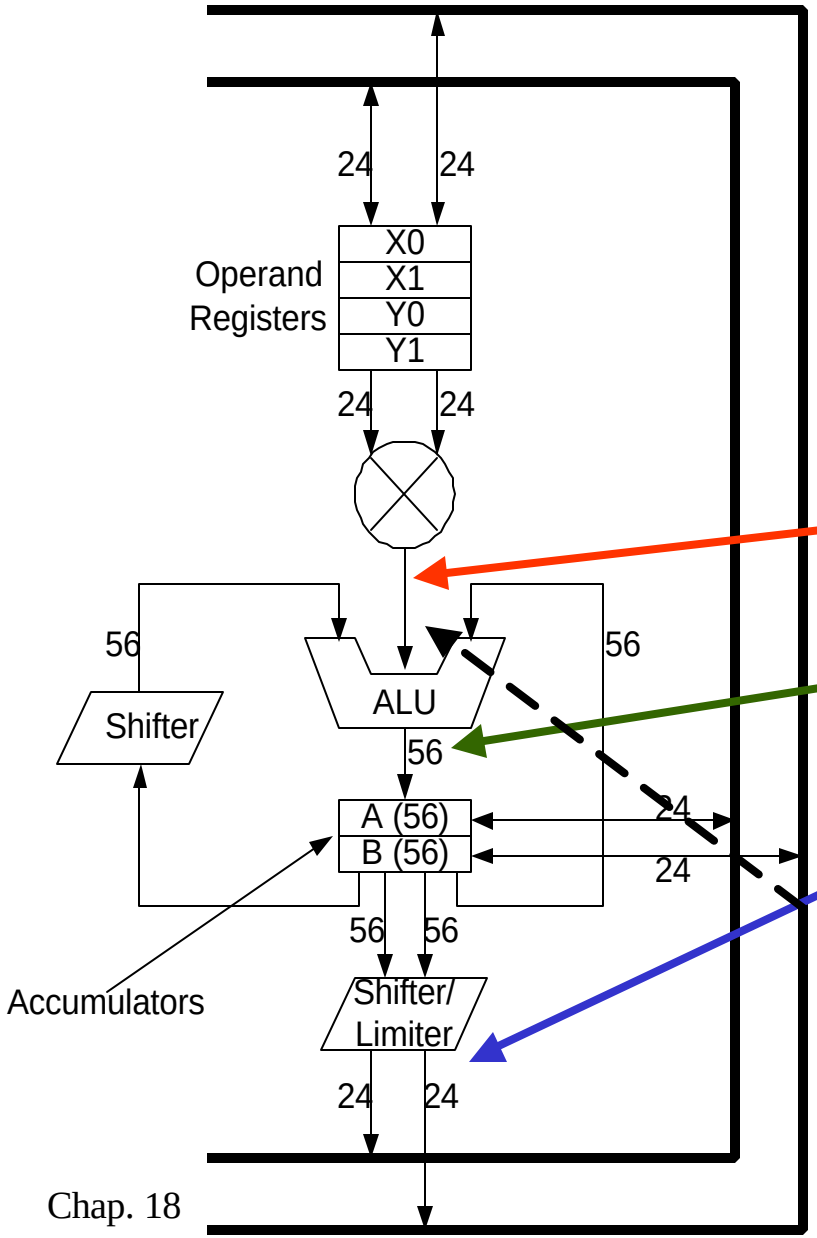
- High Calculation Capacity
- Low Power
- User defined Interface
- Variable Wordlength
- Low Price at Volume

Architectural Partitioning



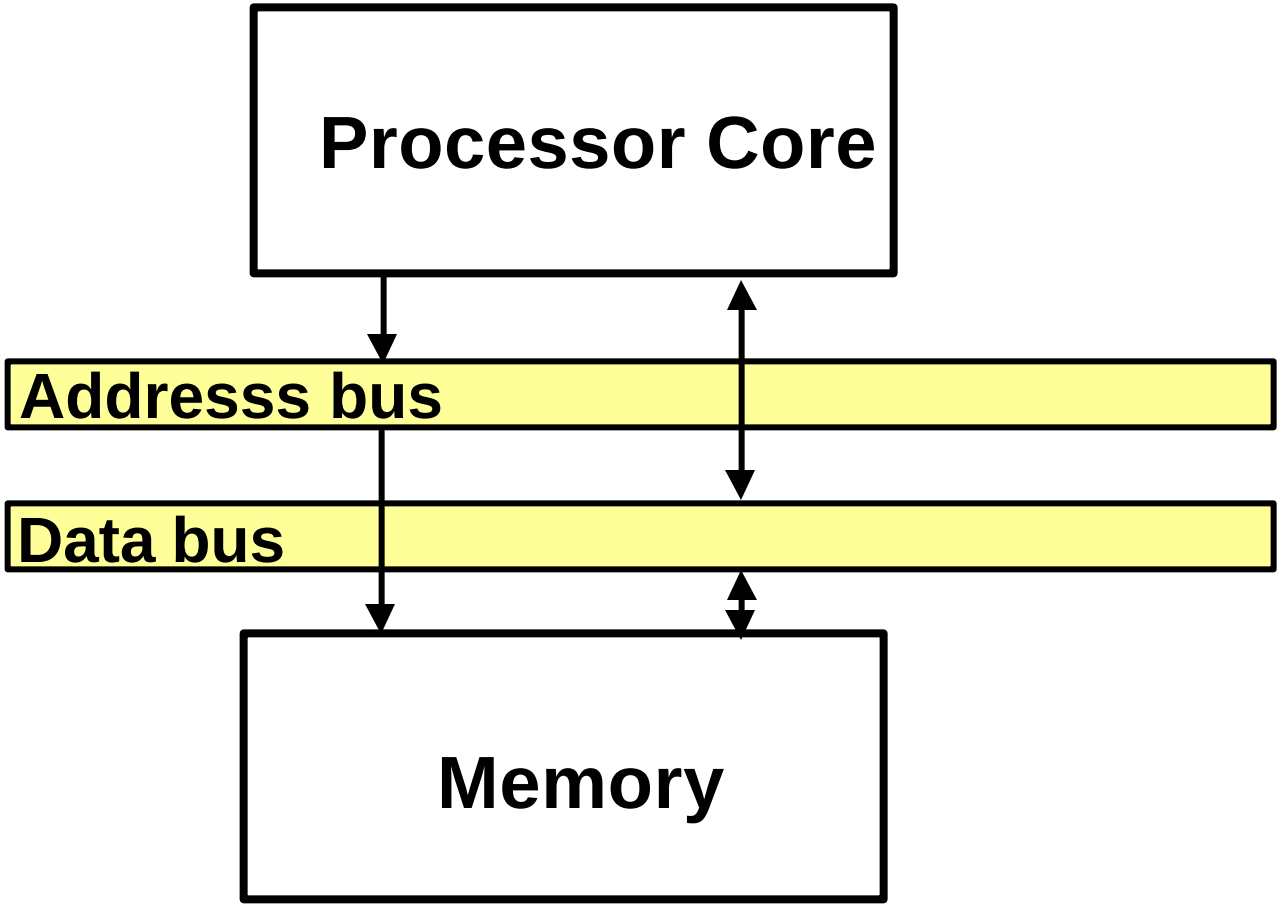
Fixed point DSP

Motorola DSP56000x

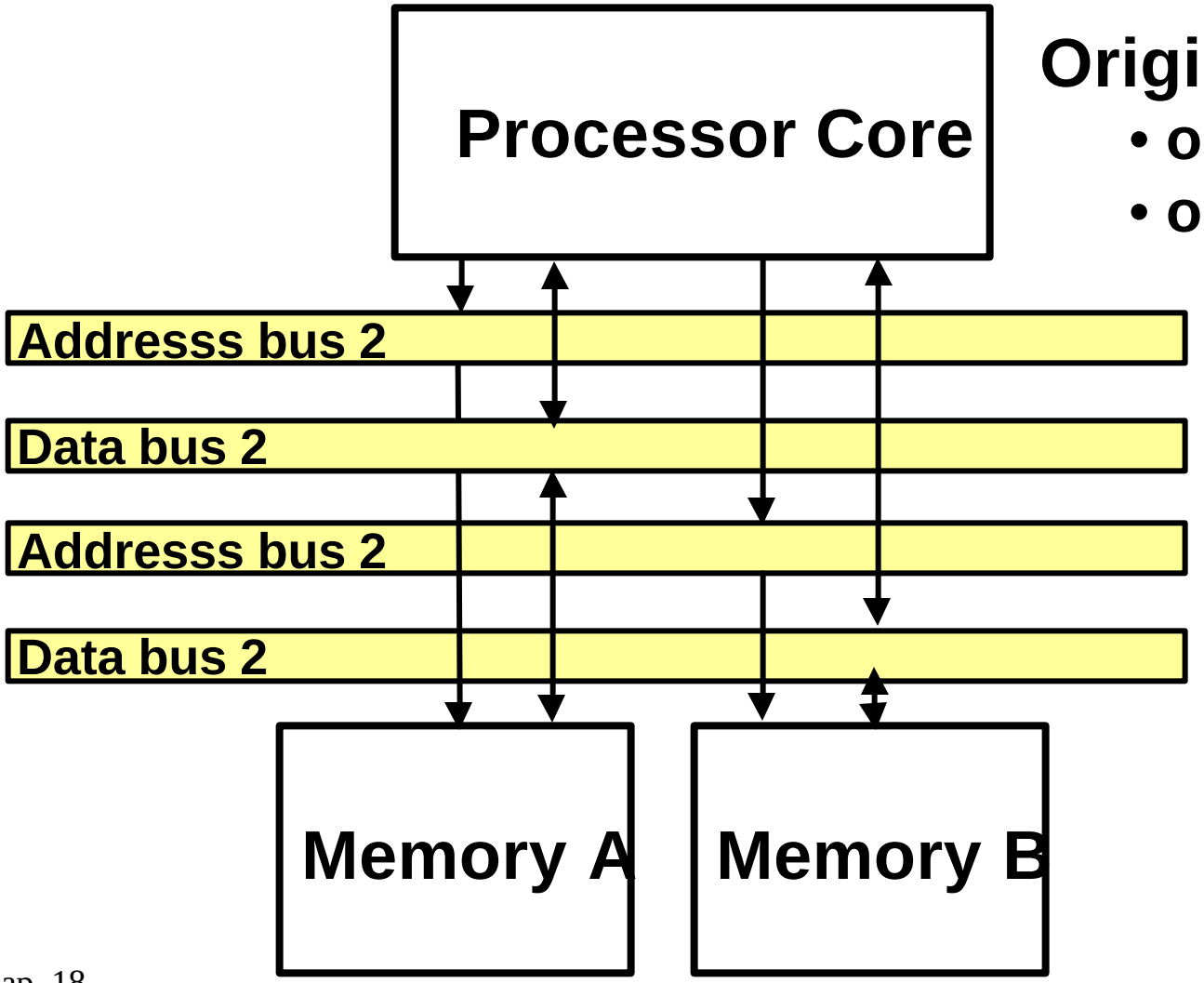


- **Usually DSP has single cycle multiplier, may be pipelined**
- **Double wordlength out**
- **+ guard bits**
- **scaling**
- **Altenative is mult with reduced wordlength output, e.g. 24**

Memory Structures, von Neuman



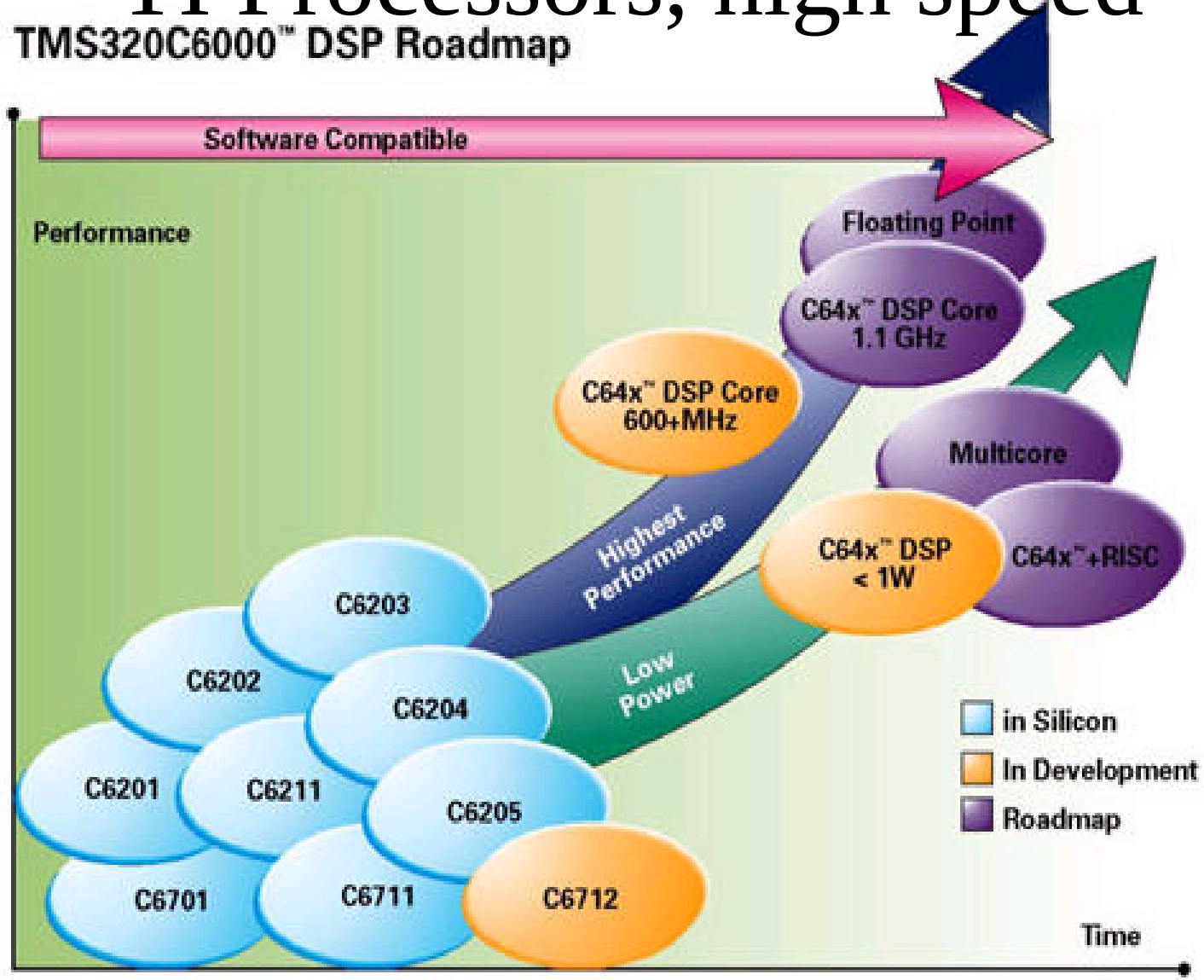
Memory Structures, Harvard



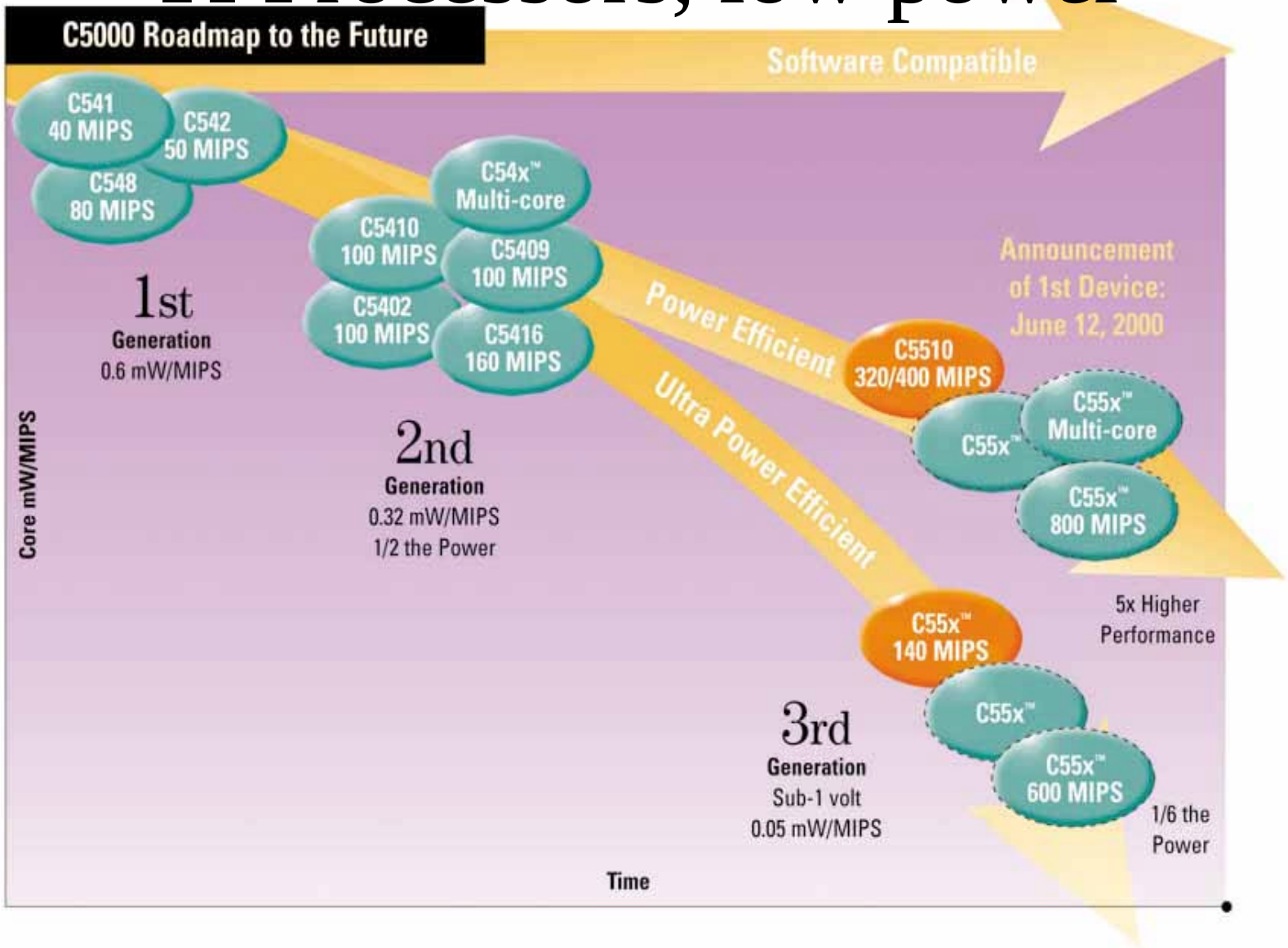
Original Harvard

- **one data**
- **one program**

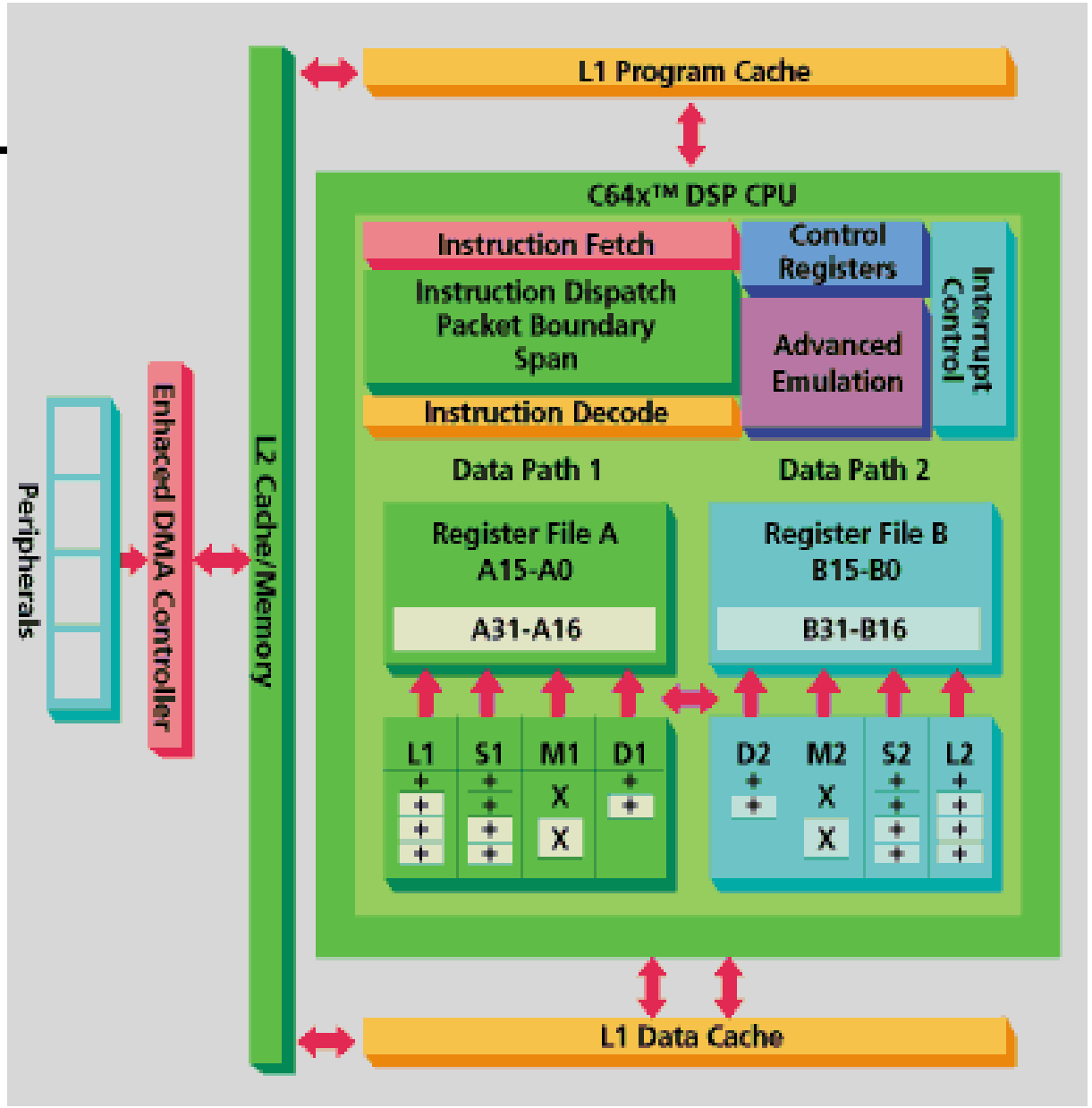
TI Processors, high speed



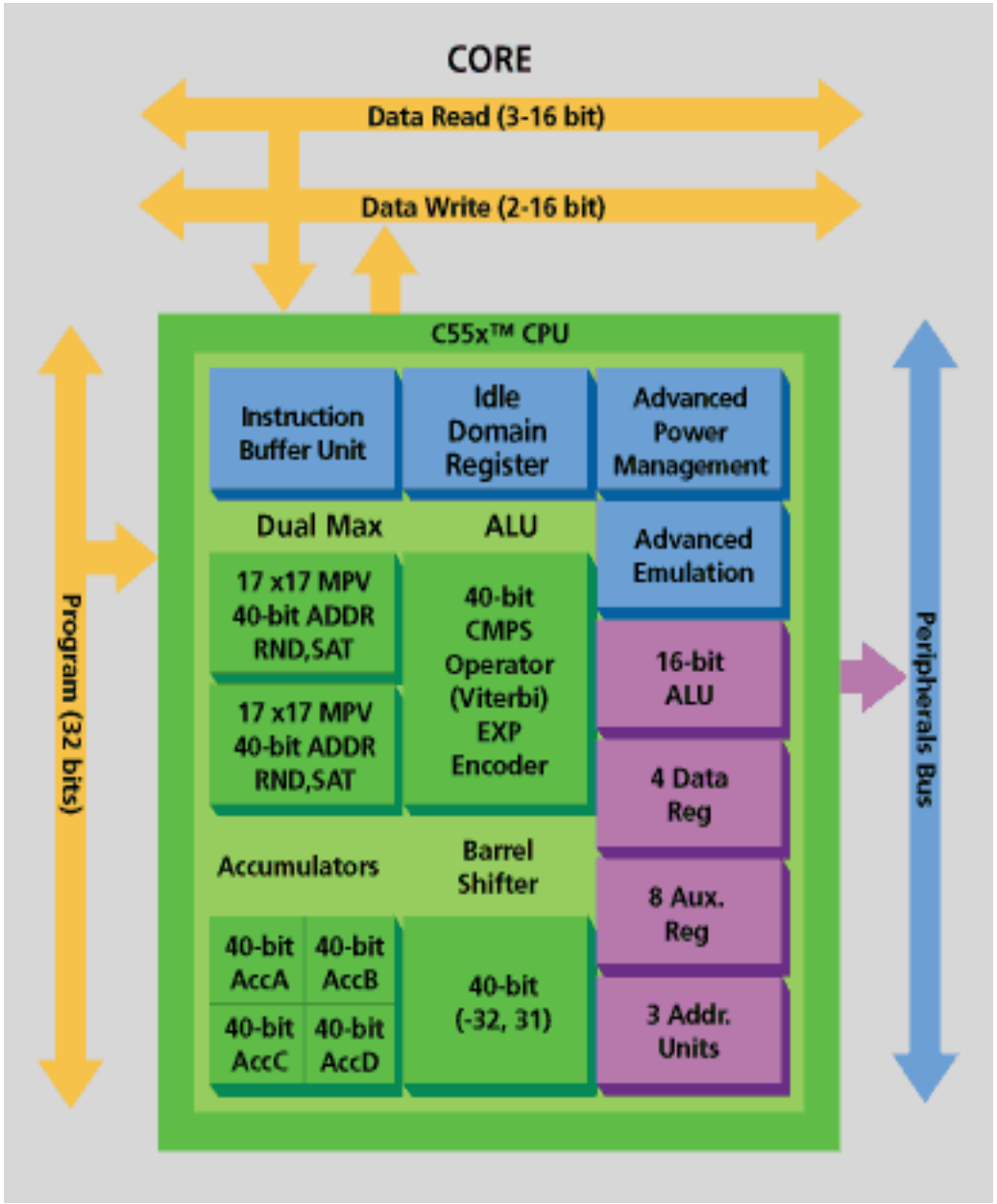
TI Processors, low power



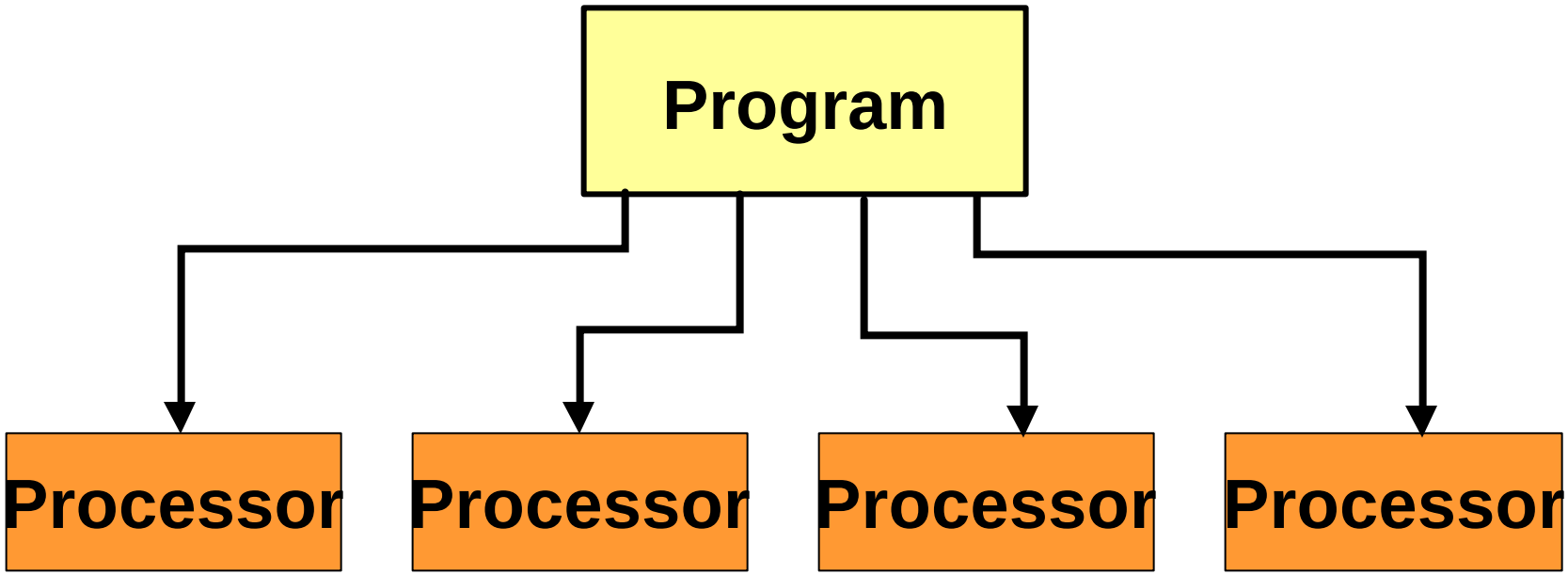
TI, C64


L1 Program Cache
C64x™ DSP CPU
Instruction Fetch
Instruction Dispatch
Packet Boundary
Span
Instruction Decode
Control
Registers
Advanced
Emulation
Interrupt
Control
Data Path 1
Data Path 2
Register File A
A15-A0
A31-A16
Register File B
B15-B0
B31-B16
L1
S1
M1
D1
D2
M2
S2
L2
X
X
X
Peripherals
Enhaced DMA Controller
L2 Cache/Memory
L1 Data Cache

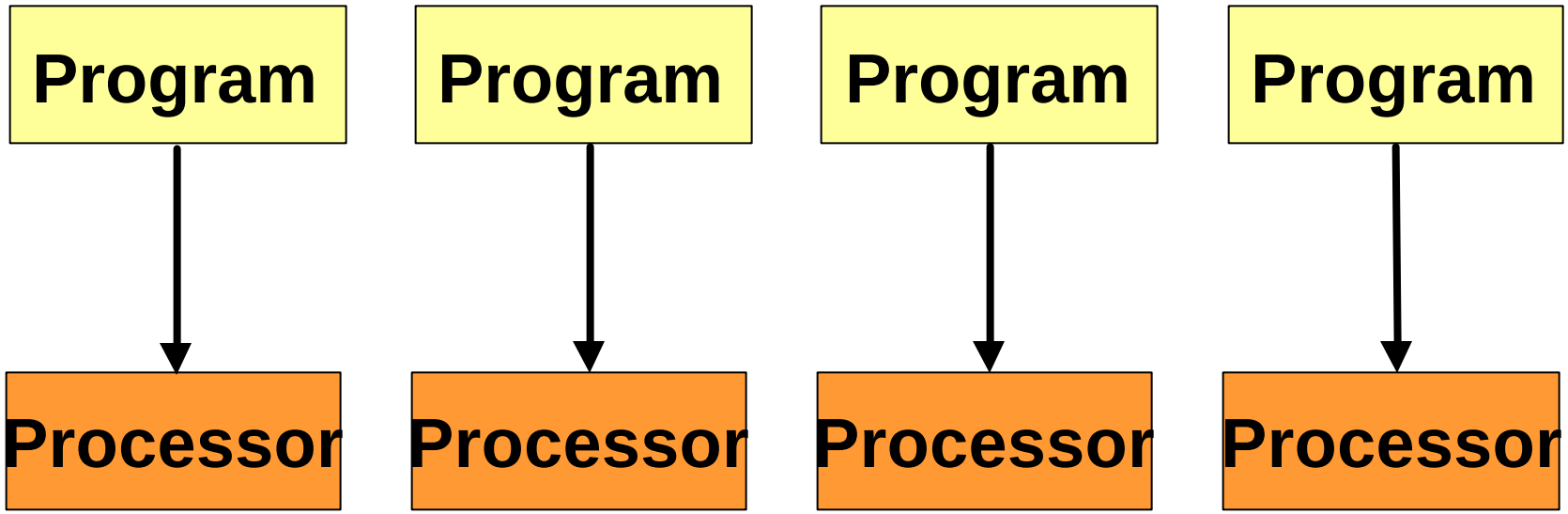
TI, C55


CORE
Data Read (3-16 bit)
Data Write (2-16 bit)
C55x™ CPU
Instruction Buffer Unit
Idle Domain Register
Advanced Power Management
Dual Max
ALU
Advanced Emulation
17 x17 MPV 40-bit ADDR RND,SAT
17 x17 MPV 40-bit ADDR RND,SAT
40-bit CMPS Operator (Viterbi) EXP Encoder
16-bit ALU
4 Data Reg
Accumulators
Barrel Shifter
8 Aux. Reg
40-bit AccA
40-bit AccB
40-bit AccC
40-bit AccD
40-bit (-32, 31)
3 Addr. Units
Program (32 bits)
Peripherals Bus

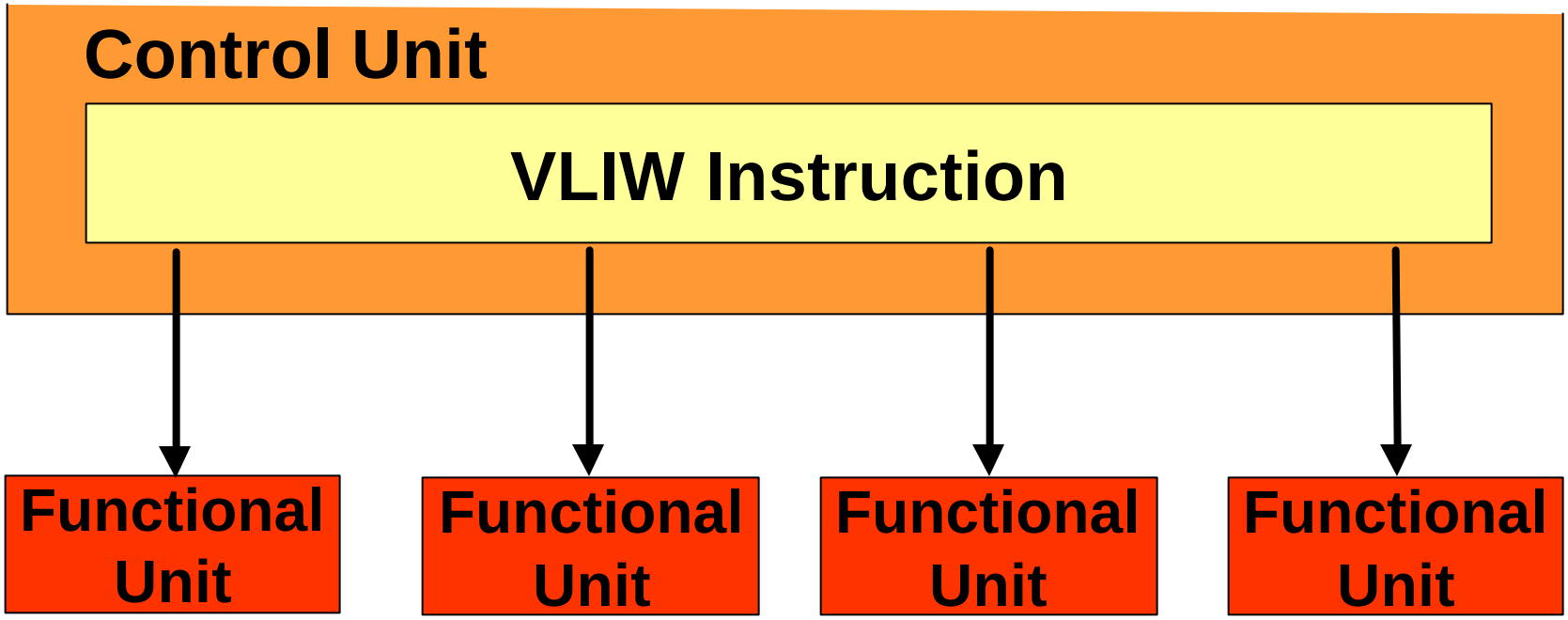
Processor Architectures
SIMD –
Single Instruction Multiple Data



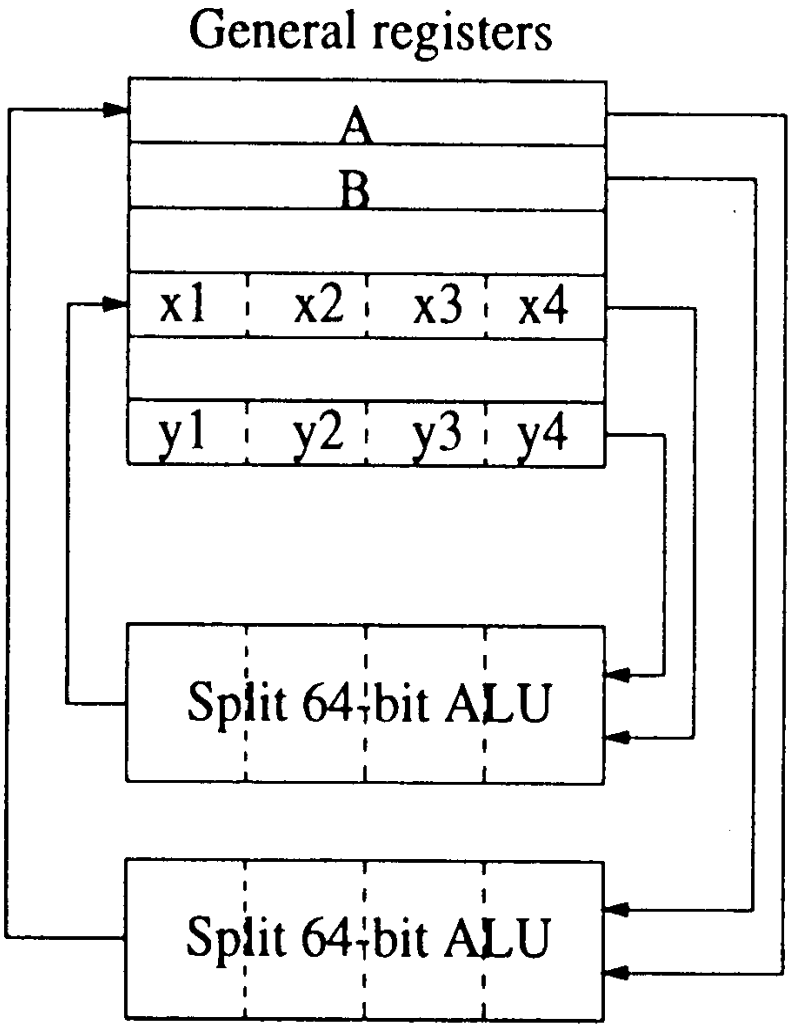
Processor Architectures
MIMD –
Multiple Instruction Multiple Data



Processor Architectures
VLIW –
Very Long Instruction Words

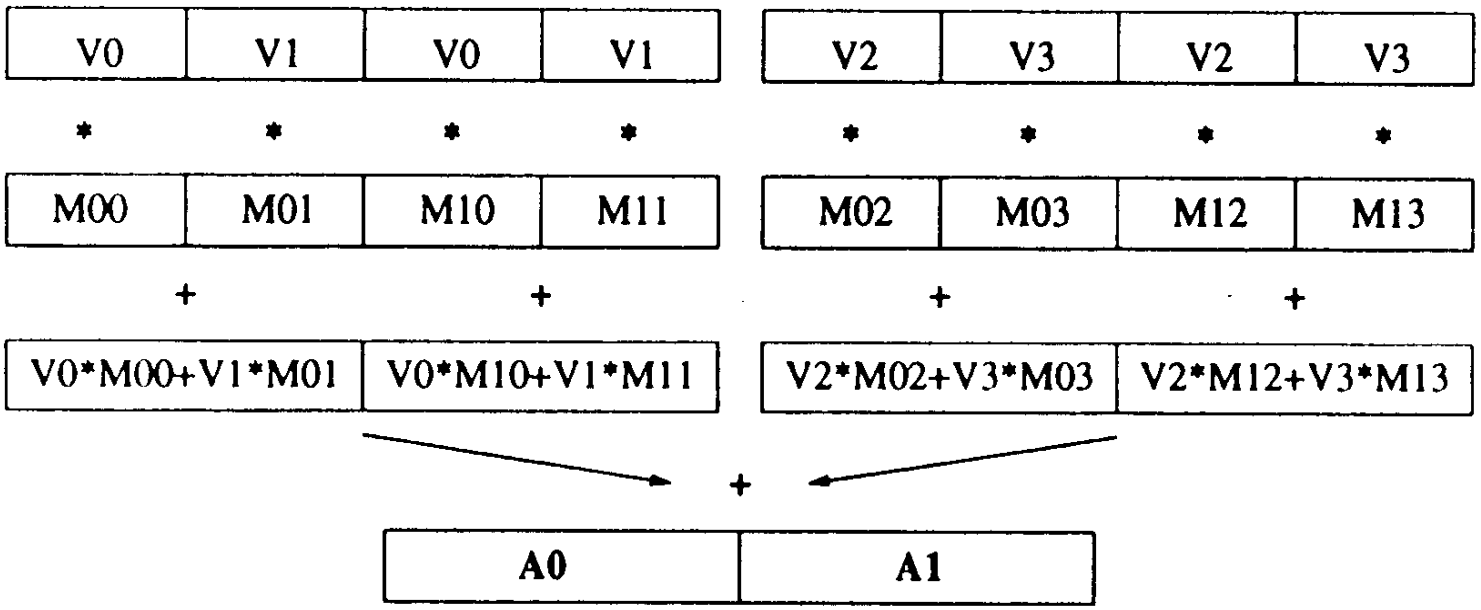


Split Processors



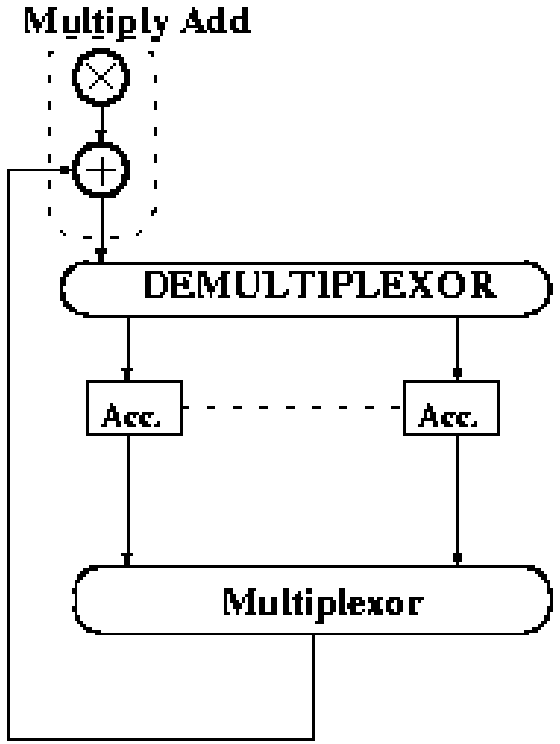
Functional units can be split into submodules, e.g. for images (8bits)
TI320C80,
1 RISC
4 x 32bit DSP which can be split into 8bit modules

Vector Processors



Low Power MMAC Multiplier Multiple Accumulator

V. Sundararajan and K.K. Parhi, "A Novel Multiply Multiple Accumulator Component for Low Power PDSP Design", Proc. of 2000 IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Vol. 6, pp. 3247-3250, Istanbul, June 2000



MMAC architecture: the number of output iterations that can coexist is equal to the number of Accumulators

Low Power MMAC

Schedule 16-tap FIR 4 acc. MMAC

Cycle #	acc$_0$	acc$_1$	acc$_2$	acc$_3$
1	$a_{15} \cdot x(4j-15)$	-	-	-
2-3	$a_{14} \cdot x(4j-14)$	$a_{15} \cdot x(4j-14)$	-	-
4-6	$a_{13} \cdot x(4j-13)$	$a_{14} \cdot x(4j-13)$	$a_{15} \cdot x(4j-13)$	-
7-10	$a_{12} \cdot x(4j-12)$	$a_{13} \cdot x(4j-12)$	$a_{14} \cdot x(4j-12)$	$a_{15} \cdot x(4j-12)$
11-14	$a_{11} \cdot x(4j-11)$	$a_{12} \cdot x(4j-11)$	$a_{13} \cdot x(4j-11)$	$a_{14} \cdot x(4j-11)$
15-18	$a_{10} \cdot x(4j-10)$	$a_{11} \cdot x(4j-10)$	$a_{12} \cdot x(4j-10)$	$a_{13} \cdot x(4j-10)$
19-22	$a_9 \cdot x(4j-9)$	$a_{10} \cdot x(4j-9)$	$a_{11} \cdot x(4j-9)$	$a_{12} \cdot x(4j-9)$
23-26	$a_8 \cdot x(4j-8)$	$a_9 \cdot x(4j-8)$	$a_{10} \cdot x(4j-8)$	$a_{11} \cdot x(4j-8)$
27-30	$a_7 \cdot x(4j-7)$	$a_8 \cdot x(4j-7)$	$a_9 \cdot x(4j-7)$	$a_{10} \cdot x(4j-7)$
31-34	$a_6 \cdot x(4j-6)$	$a_7 \cdot x(4j-6)$	$a_8 \cdot x(4j-6)$	$a_9 \cdot x(4j-6)$
35-38	$a_5 \cdot x(4j-5)$	$a_6 \cdot x(4j-5)$	$a_7 \cdot x(4j-5)$	$a_8 \cdot x(4j-5)$
39-42	$a_4 \cdot x(4j-4)$	$a_5 \cdot x(4j-4)$	$a_6 \cdot x(4j-4)$	$a_7 \cdot x(4j-4)$
43-46	$a_3 \cdot x(4j-3)$	$a_4 \cdot x(4j-3)$	$a_5 \cdot x(4j-3)$	$a_6 \cdot x(4j-3)$
47-50	$a_2 \cdot x(4j-2)$	$a_3 \cdot x(4j-2)$	$a_4 \cdot x(4j-2)$	$a_5 \cdot x(4j-2)$
51-54	$a_1 \cdot x(4j-1)$	$a_2 \cdot x(4j-1)$	$a_3 \cdot x(4j-1)$	$a_4 \cdot x(4j-1)$
55-58	$a_0 \cdot x(4j)$	$a_1 \cdot x(4j)$	$a_2 \cdot x(4j)$	$a_3 \cdot x(4j)$
59-61	—	$a_0 \cdot x(4j+1)$	$a_1 \cdot x(4j+1)$	$a_2 \cdot x(4j+1)$
62-63	-	-	$a_0 \cdot x(4j+2)$	$a_1 \cdot x(4j+2)$
64	-	-	-	$a_0 \cdot x(4j+3)$