Introduction to the Digital Signal Processor Design

Dake Liu
Associate Professor at Electronic Components
Department of Physics and measurement technology
Linköping University, Linköping, Sweden

Introduction to the course for today

Introduce the basic concept

DSP Applications

Focus on HW implementation

Behavior process flow and Architecture

BUS and Memory Architecture

Datapath

Control path

Peripherals

Development flow & environment

Introduction to Digital Signal Processors

1. Why touch Design of DSP Processors

The market view

- ☛ At least three DSP processors used in a mobile phone.
- . More than 150 million mobile phone sold last year.
- . It means > $2 billion used for DSP's inside mobile phones.
- ☛ Sales of user programmable DSP chips in 98 was $3.9 billion.

The applications

- . The communication is just DSP + Control + transmission
- . DSP makes high performance home electronics possible
- . DSP is the key component in industrial applications
- . DSP is the key component for medical, military, and so on
- . A cheap, easy, and reliable solution based on VLSI technique

2. ***General*** algorithms for signal processing:

Is the Kernel algorithms executing in DSP processors

Define the Kernel to formalize the problem

- Convolution based algorithm -- FIR

$$y(i) = \sum_{k=0}^{N-1} h(k)x(i-k)$$

- Convolution based algorithm -- IIR

$$y(i) = \sum_{k=0}^{M-1} a(k)y(i-k) + \sum_{k=0}^{N-1} b(k)x(i-k)$$

- Minimal distance based algorithm

$$y(i) = \frac{1}{N}\sum_{k=0}^{N-1} [|x(i) - r_k(i)|]^{1(2)}$$

- Signal modeling based algorithm

$$a_{ik}^{new} = a_{ik} - \frac{a_{ij} a_{nk}}{a_{nj}}$$

- Control algorithms (approaching)

$$d = k_1 x(n) + k_2 [x(n) - x(n-1)] + k_3 \sum_{i=0}^{\infty} x(n-i)$$

- Numerically stable rotation based algorithm

$$a'_{ik} = a_{ik} cos\Theta - a_{jk} sin\Theta$$
$$a'_{jk} = a_{ik} sin\Theta - a_{jk} cos\Theta$$

A DSP Behavior flow

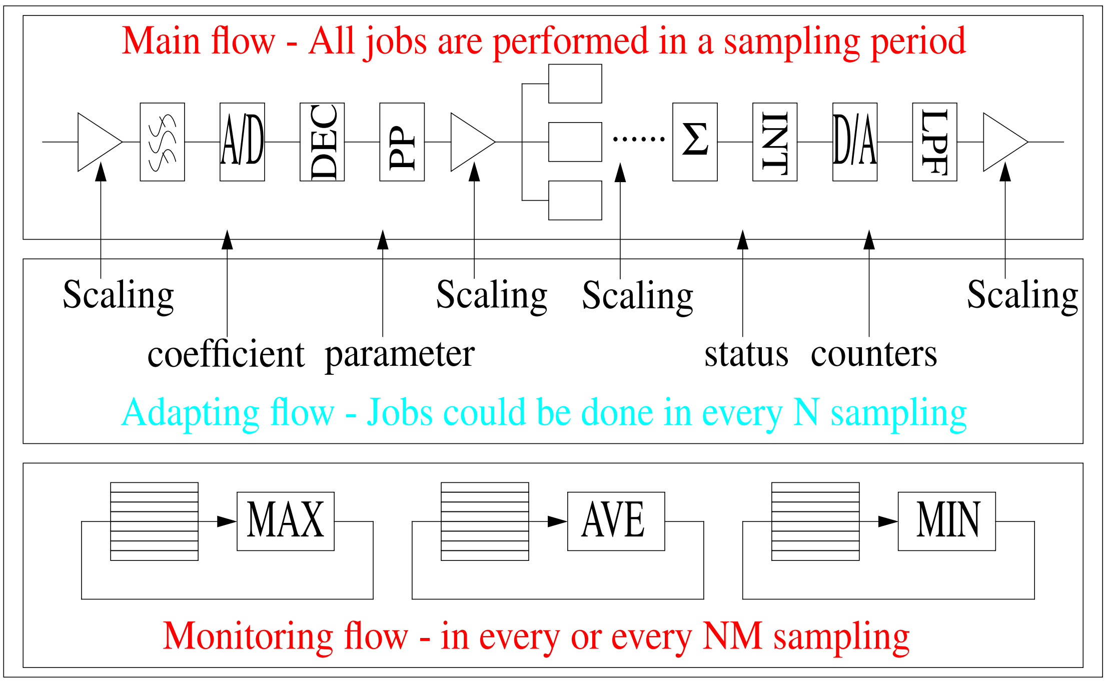
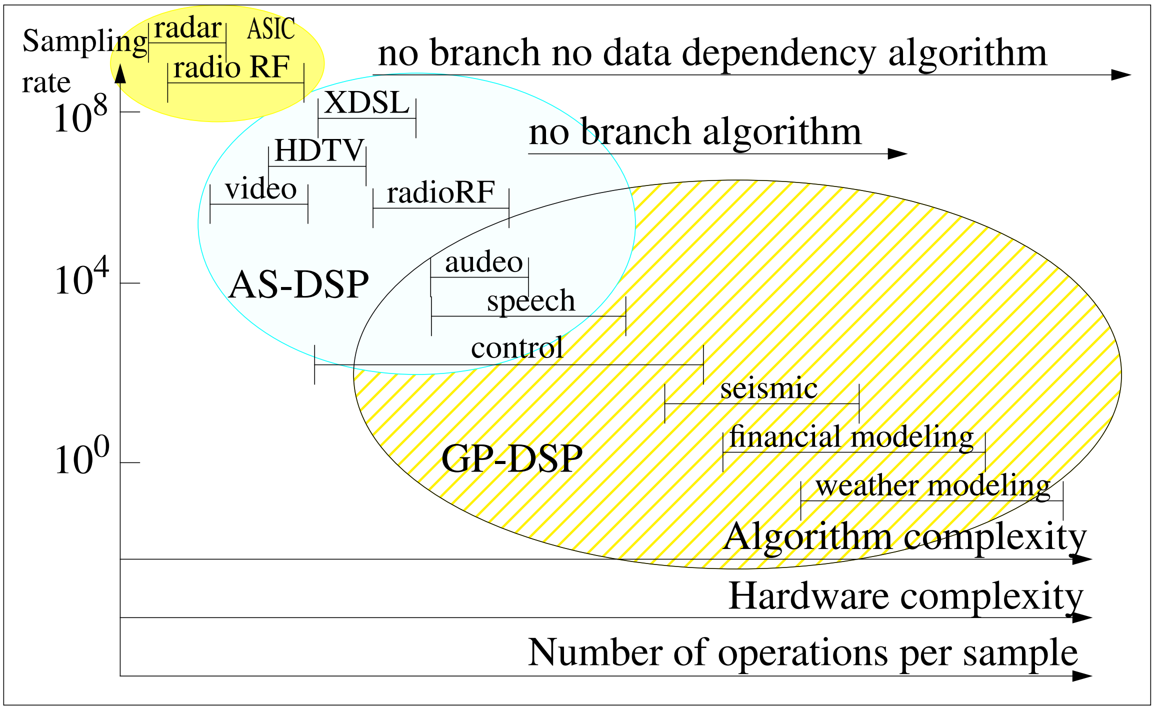


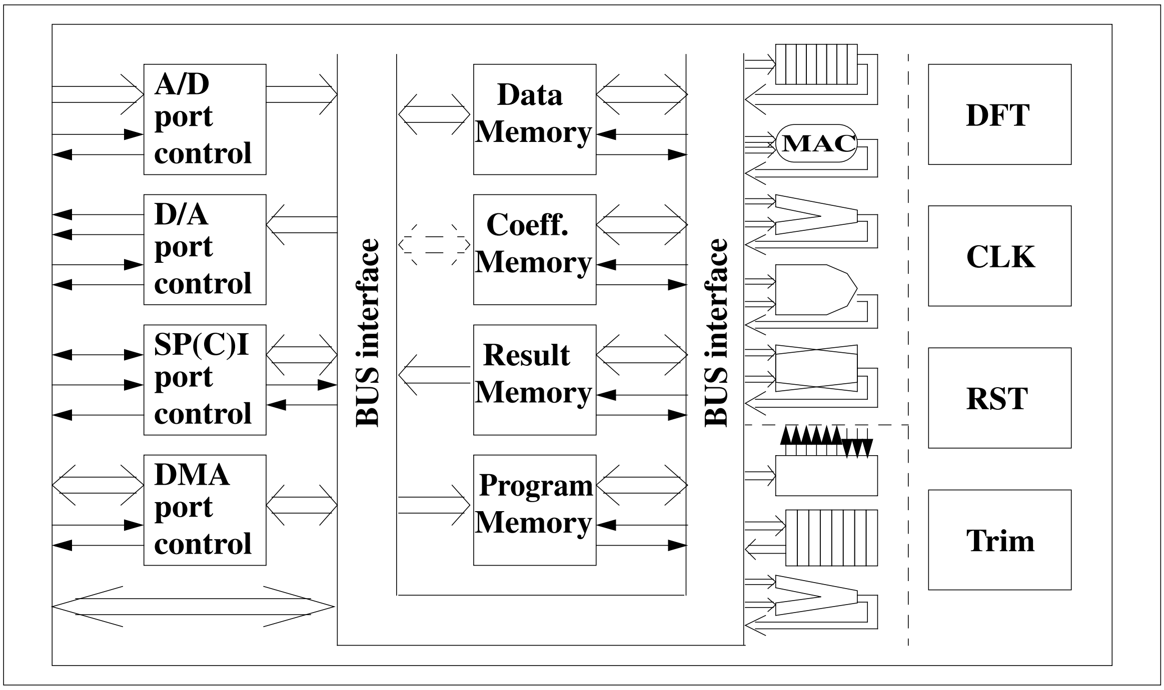
Table 1: Implementations of DSP Processors

HW	Algorithm driven Architecture	Appli. specific processor	General purpose processor
Design method	Mapping to HW	Mapping to AIS	Mapping to AIS
Application	Simple	complicated AS	General
Design cost	Very high	Not very high	Low
Low volm cost	Very high	high	low
High volm cost	low	low	high
TTM	Very long	X00k g / day	Short
Flexibility	Poor	Good	Very good
Performance	Very high	High	V.S. Price
Power	low	medium	?
Industr. activity	<1%	~35%	65%

Applications



A DSP Processor in general



Features of DSP Processors

1. Fast MAC (Multiplication and accumulation):
   We often measure the possible number of MAC ops per cycle.
2. Multiple access memory architecture
   Memory access is the bottleneck of DSP performance
   We need multi-bus structure
   We need multi-memory structure
   We need multi-port memory structure
3. Specialized addressing modes
   Memory access is the bottleneck of DSP performance
   Addressing mode to give higher performance
   Low power addressing model
   Implied addressing model used for saving the instruction memory
4. Specialized execution control
   Loop control, fast in-out control, integrated multi op instructions

A way of measure

. MIP: Very old concept, still used to fool customers

. MMACPS: Not easy to use, give you the real performance

. High multiplexed DSP processor: $n_{OP} > 100$

. Low multiplexed DSP processor: $n_{OP} < 100$

. Power: mW/MIP or mW/MMAC

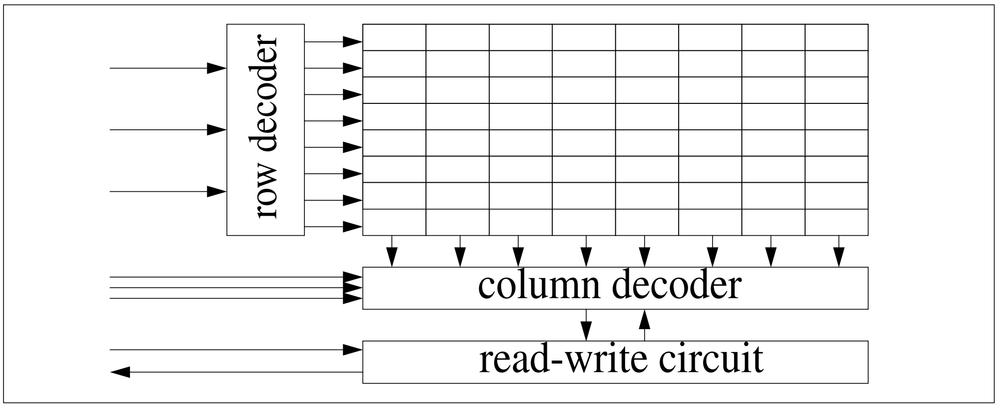
. AIS: General parameter process, special DSP supporting

. Addressing: HW looping, step, bit inversel, fast branch

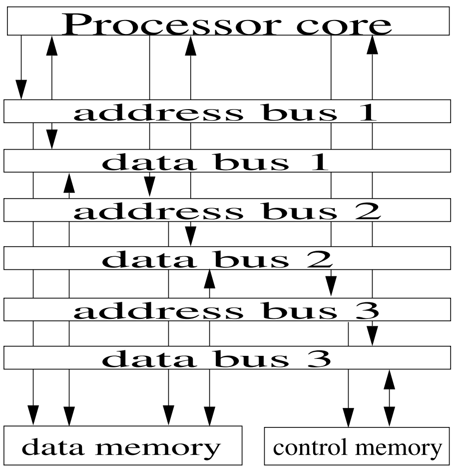
. Peripheral: Performance, compatibility, pin cost

Bus and memory architecture

1. Basic memory component:
   We mean a single port Static Random Access Memory (SRAM)
   ROM is not discussed because:
      Embedded EPROM request a special silicon technology.
      Simple fixed ROM limited the flexibility
2. General architecture of a Single port SRAM (64X1 bits)



3. Requirements:
   a). To support arithmetic-intensive algorithms.
   b). Multiple operands per instruction (weak data dependency).
   c). Read and write integrated in one cycle.
4. Improved Harvard Architecture



Harvard architecture Traditional.

Use one independent memory-bus set for the Data

Use one independent memory-bus set for the Coefficient

Use one independent memory-bus set for the program

Datapath

Definition:

Data path: is where arithmetic computations take place.
The HW from the data input bus to the data output bus.

Data path components:

1. Multiplier:

Basic concepts.
   a). OPA and OPB are native size (or native size plus one).
   b). result width is twice of the native size.

Integer or fractional multiplier:
   a). Why: for both arithmetic-fractional and integer-controls.
   b). Alignment problem when combine both multipliers.
   c). Overflow problem when using integer multiplier.

Signed/unsigned:

2. Accumulator and Accumulator register:

Basic concepts.

Accumulator is a full adder with its register as one operand input.

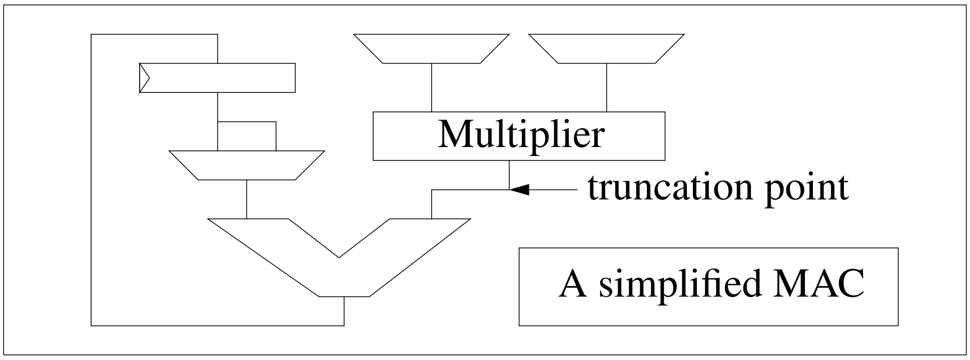
Accumulator register: storing the intermediate result

It performs the core function together with the multiplier.

ACR = ACR + MA*MB

for convolution based and rotation based algorithms

MAC Multiply-Accumulator



Digital Signal Processor Design

Overflow:

The magnitude of the ACC sum exceed the maximum value

Key DSP feature: Relative result is enough! (no need absolute)

Scaling:

☞Implied scaling by right shifting the result coming to ACR,

☞Dedicated MAC scaling instruction,

Guarding:

extra HW on the left of sign bits,

better than scaling, because of the higher precision

Saturation (MSB side):

If $A \geq 1$ then A = MAXP, if $A<1$ then A = MAXN

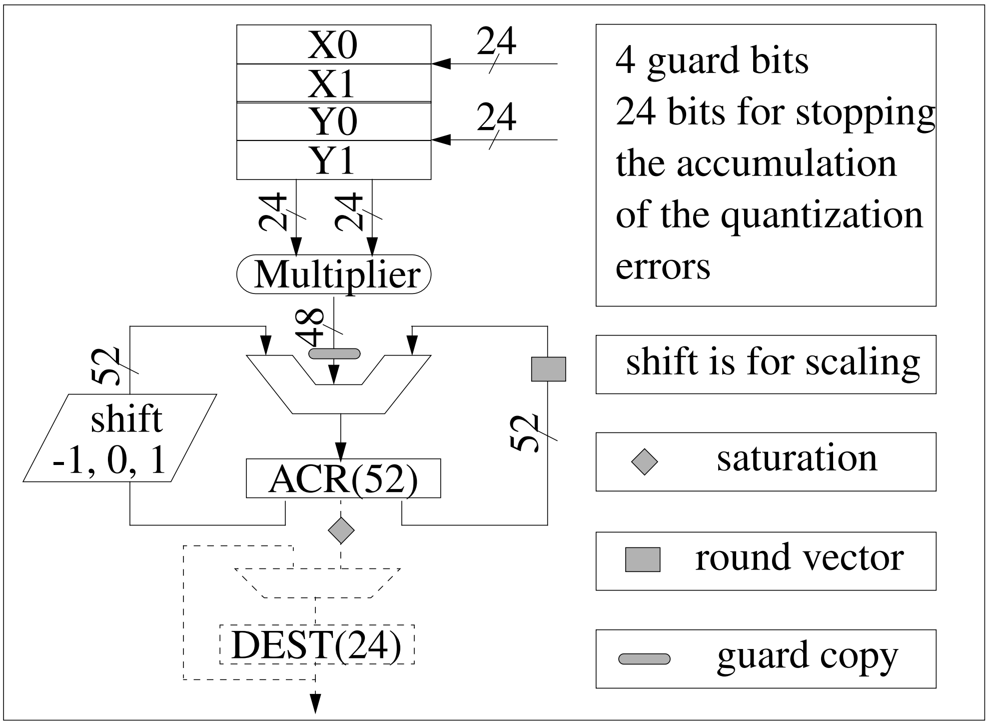
Truncation for the final result of the MAC (LSB side)

just cut the expended word to the native width

Rounding for the final result of the MAC (LSB side)

change the LSB according to the bit being cut off.

An example of a typical MAC



3. Arithmetic and Logic Unit (ALU):

Basic concepts.

Similar to the integer ALU in a CPU

AU (Arithmetic Unit):

Add, Subtract, Absolute, Compare, Negation.

Carry in and carry out for double precision

SU: (Shift Unit) Combinational logic

Arithmetic right shift, Logic left and right shift: for controls:

LU (Logic Unit):

Pre-op: Invert; Logic op: or and xor: Post-op: Invert, dest.

General purpose register file.

Buffer between the data path and the memory bank

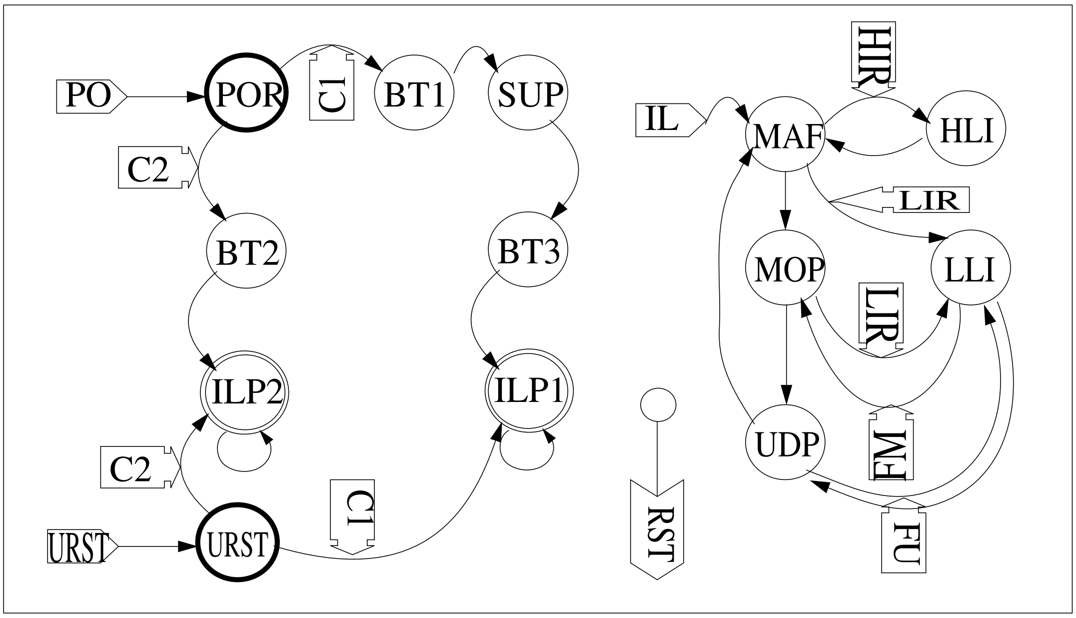
Integration of ALU (or ASLU).

Operand control, functional integration, result selection

Fast data path integration, low power applications

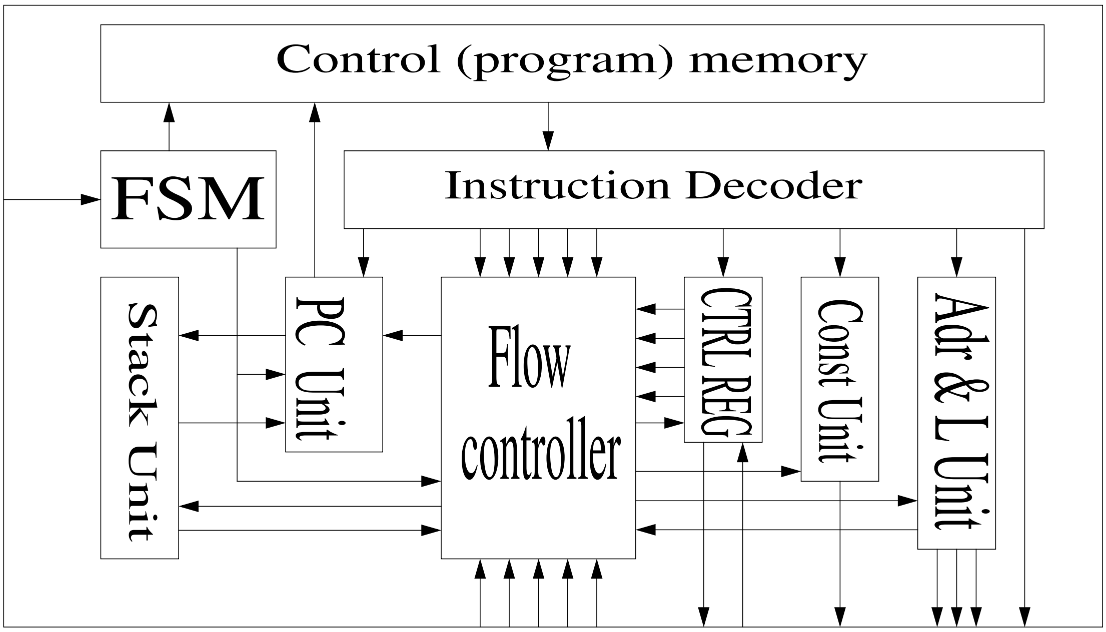
Control path

An example: overview of a control flow:



Digital Signal Processor Design

Example: An overview of a control path block diagram



1. Control path fundamentals
   - An infinite loop execution
   - General functional description
   - Program flow control
   - Pipeline scheduling and Hazard management
   - Interruption and exception
2. Hardware looping
   - Single instruction loop and multiple instruction loop
   - PC and Program flow control
   - memory address supporting
   - Modulo address for the data memory
   - Loop Stack and Nesting
3. Branch
   - Branch hazard management
   - Fast branch and implied prediction

Peripherals of a DSP chip

1. BUS Hierarchy

a). Module internal bus (micro code controlled)

. For example, Bus inside ALU, in general purpose register file

b). DSP Processor memory bus (micro code controlled)

. Bus between processor core and memories, short, fast

. multiple bus in DSP processor, strobbed to local module buss

c). I/O peripheral bus (configuration and interruption based)

. lengthy, link to many devices, balance cost and performance,

. following standard because of unknown devices connected

d). Backbone bus (program controlled, often standard bus):

. Connections between boards and between magazines.

. Controlled by a bus control board or a BUS controller.

e). System BUS (protocol based, application independent)

. To inter-connect your system to the outside of the world.

2. Three measure of communications:

a). latency: the time taken for an operation
. From event application to done acknowledgment
. Short response time: large number of small unrelated access
. Latency is not important for a long stream of data DMA e.g.
b). I/O bandwidth: the rate at which operations are performed
.Definition is based on your application: do not be fooled!
.How much ***data*** moved through the system in a unit time.
.How many I/O operations can we do per unit time (***control***)
c). cost: the impact these operations have on the execution time of the program (define your own benchmark)
. HW cost: cost of pins, cost of complexity, cost of memory
. Resource cost: cost of DSP core usage, cost of internal bus
. Development and maintains const: coding time, debugging.

3. Communications modes:
a). Send and forget (fixed timing):
. No request no acknowledgment, no accompanied control.
. Sync: Instruction controlled, receiver can not be busy.
b). strobe (fixed timing):
. A select control pulse to release data at a precise instance
. Data is enabled - disabled by a valid signal
. No acknowledgment is requested
c). handshake protocol based (variable timing):
. Consist of a series of steps both parties agree
. Allows convenient and variable response time.
d). FIFO (variable timing):
. An asynchronous buffer memory is used
. A interface between synchronous and asynchronous

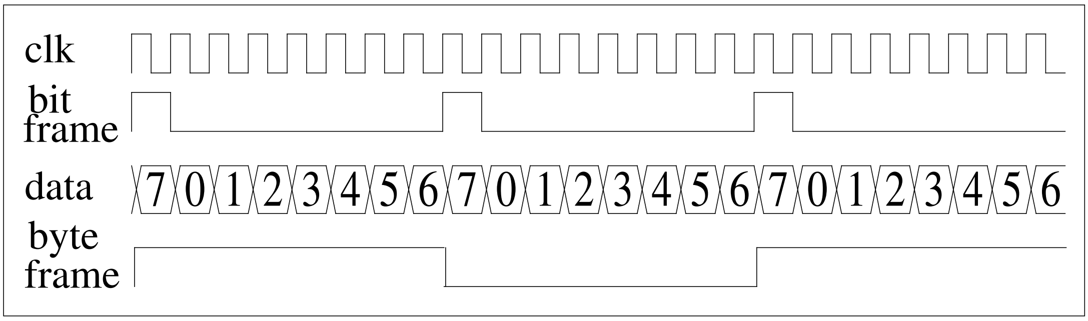
4. Synchronous bus:
   . pre-defined clock for the data transmission
   . Advantage: controlled by a small FSM, simple, fast.
   . Disadvantage: Limited at the same clock, short distance
5. Asynchronous bus:
   . Is not clocked! handshaking protocols are used.
   . Longer latency, not fast because of the acknowledgment.
6. DSP Solutions:
   a). Instead of trading off requirements and performances,
      . DSP implement different interfaces for different purposes.
   b). Application specific:
      . No common rules can follow, follow the requirements
   c). Special feature of message (not data) carried on I/O:
      . Sampling based

7. Serial Synchronous Interface

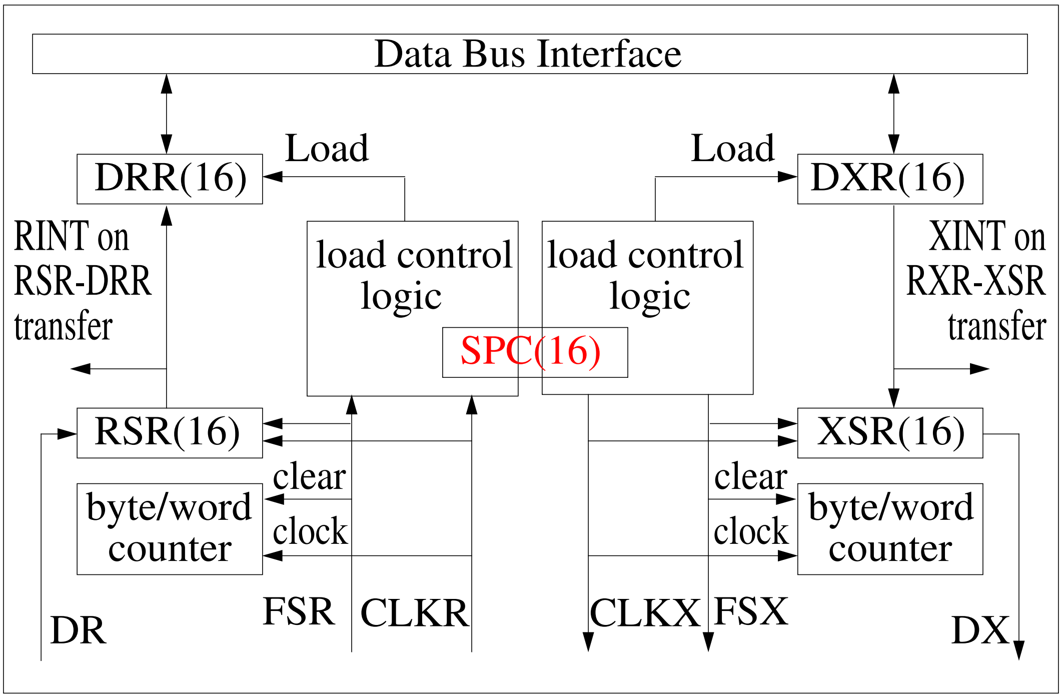
. General:
   Serial interface: Transmit/receive data one bit at a time.
   use less interface pins.

. Applications:
   communication with the micro controller and other DSPs.
   interface between DSP and AD, DA, and other components.

. Frame synchronization
   - Bit length frame sync (MSB first, one word per frame):



8. Serial Peripheral Ports (SPI):

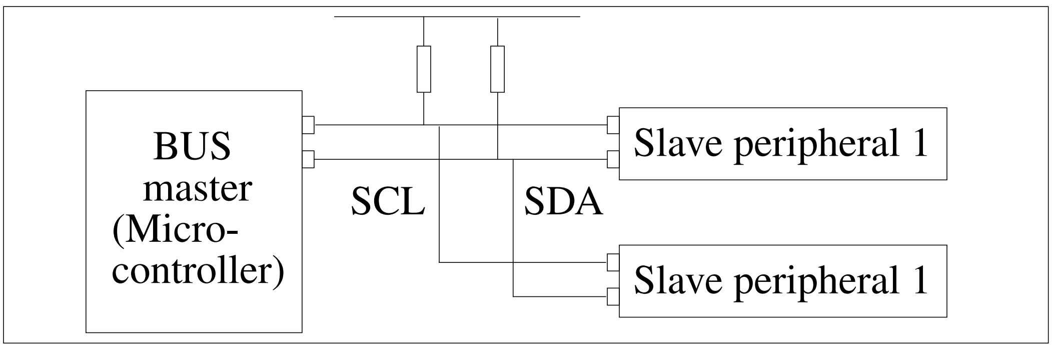


9: I^2C bus

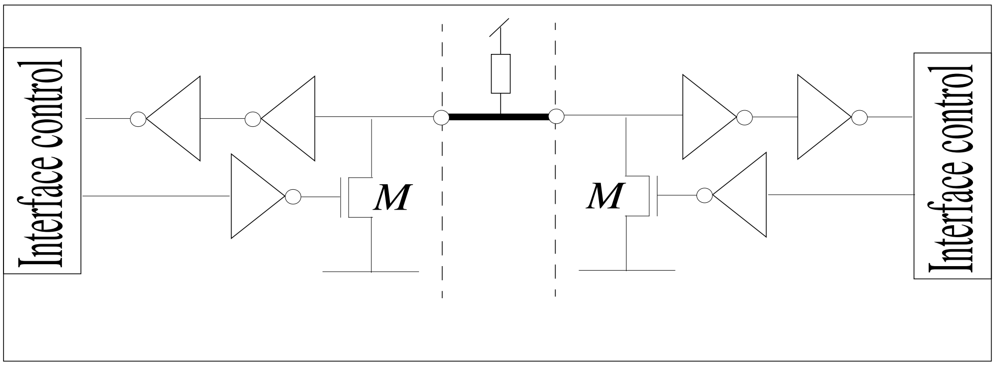
1. General:

Developed by Phillips originally, **Inter IC or I^2C bus**.
Bi-directional two wire only hardware protocol.
Software protocol for integrated address and data-transfer.
Multiple master bus including collision detection and arbitration
The maximum bus capacitance is 400pF.

2. Interface:



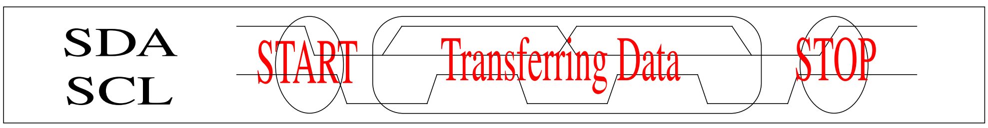
. SCL pin provides reference clock and hardware handshaking.
. SDA pin ensures the serial data is clocked out using SCL.
. Low speed and limited number of loads: no problem
. High speed and heavy line load: use a special buffer chip
. Open drain driver request pull up resistors:



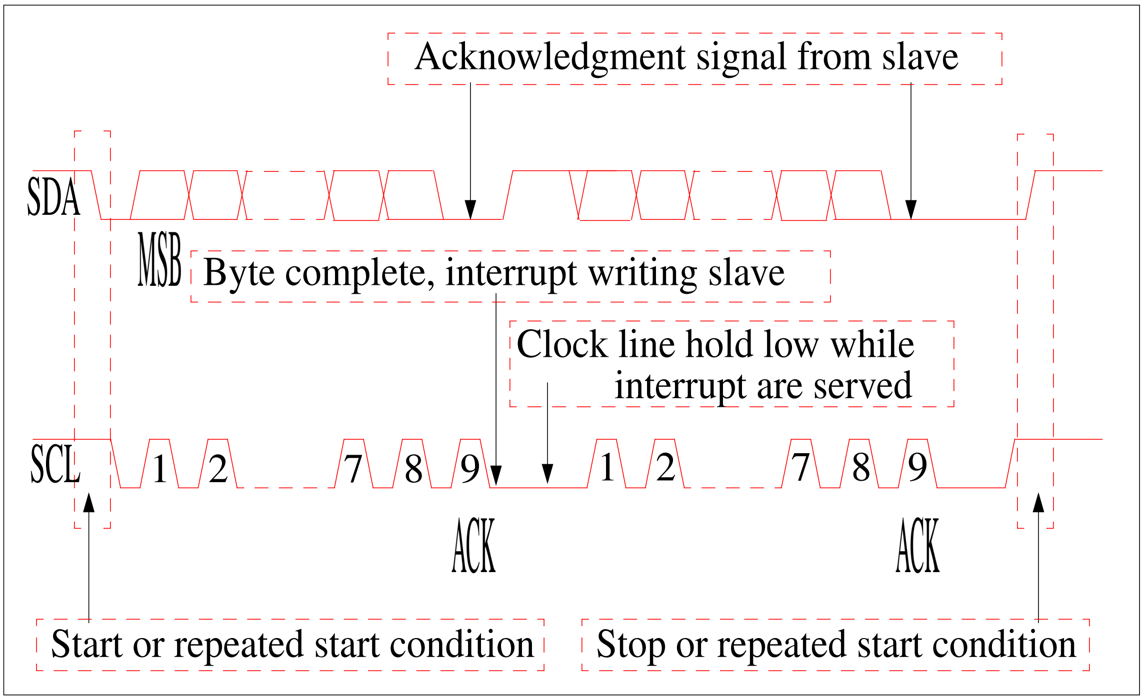
. When driving, the line connected to the ground
. When no driving, the output pulled up.

Digital Signal Processor Design

5. Hardware Signaling
- Clock signal generation:
  The master's responsibility!
- One clock pulse is generated for each data bit transferred
  Data on the SDA line must be stable when clock SCL is high
- Start and stop conditions:
  Must be unique, transparent to the data transmitted.
  Start condition: A H to L transition on SDA while SCL is low.
  Stop condition: A L to H transition on SDA while SCL is low.
  Start and stop conditions are always generated by the master.
- Bus status:
  The BUS is busy after the start condition,
  The BUS is free a certain time after the stop condition.

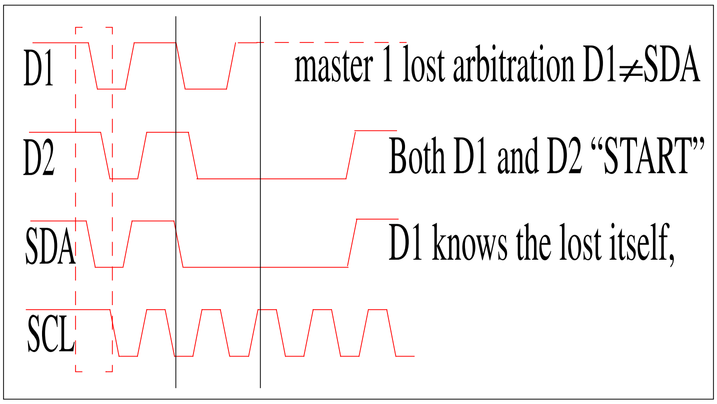


7. A complete data transfer over I^2C:



9. Clock generation
   - The master generate their own clock on the SCL.
     Data is valid during the HIGH period of the clock.
   - Clock synchronization is performed using wire AND
     all high is high, anyone of the device pull it low, it is low.

10. Arbitration
   - When two masters request a start the same time:



a. Both master 1 & 2 want to trans.
b. Both START as the line is free.
c. Both send data to SDA.
d. SDA arbitrate when SCL is high.
e. The one data is high lost right.
f. Because D1 differ from SDL.
g. No data lost during arbitration.

DSP Development Environment

Algorithm Behavior - Bit true, not yet cycle true

Architectural Behavior - Both bit and cycle true

Instruction Level - Compiler

Register Transfer Level - Design entry and simulator

Back end - Silicon implementations

Conclusion for today: Compare the DSP Processor with CPU

Table 2: Compare DSP with CPU

	DSP	CPU
Job	Based on Fixed infinite loop(s)	Who knows
Data type	1. Fixed point. 2. Fractional	Floating and integer
Data path Component	MAC and ALU	Floating Unit, Fixed Unit, etc.
Data size	Much higher than native	Equal to native
Data alignment	MSB	LSB integer, MSB floating
Data overflow	Guards and saturation	Carry out or exception interrupt
Data exception	skip it and keeps the old	exception interruption
Structure	Multi data, not multi instruction issue	Multi instruction issue
AIS Statistics	MAC with higher probability	MOVE with higher probability
Instruction	Cycles per instruction is fixed	Cycles per instruction can't be fixed (memory access)
Scheduling	Fixed pipe	Dynamic pipe because of cache (dynamic retire)
Data memory	Multiple Data memory and multiple access	One Data memory supported by cache
Control Memory	Simple, mostly single	Cache supported
Data Addressing	Dedicated Data address generate	Level hierachy
Program addressing	Dedicated loop and branches	Segment level hierachy

Table 2: Compare DSP with CPU

	DSP	CPU
BUS	Multi local and simple buses	Level hierachy global complicated bus system
Branch	Simple no prediction	Event based prediction
Control Path	Direct decoding	micro code based execution
Control except	Skip job in a sample period	interrupt
Results	relatively result with absolute precision	absolutely result with relatively precision