LG3: SoC System Design (SD)


- LG3.1 - SD System On Chip: System Design
- LG3.2 - SD Microcomputer Processor Bus
- LG3.3 - SD Microcontroller
- LG3.4 - SD Switch/LED I/O
- LG3.5 - SD Programmed I/O
- LG3.6 - SD Interrupt Structure
- LG3.7 - SD GPIO
- LG3.8 - SD Keyboard Controller
- LG3.9 - SD Counter/Timer

- LG3.10 - SD Video Controller
- LG3.11 - SD DMA Controller
- LG3.12 - SD Network Device
- LG3.13 - SD Bus Bridge
- LG3.14 - SD Clock Tree
- LG3.15 - SD Clock Domain Crossing
LG3.1 - SD: System On Chip: System Design

System on a Chip = SoC design.

Platform Chip: Example Virata Helium 210
www.icfinder.co.kr/pdfview.asp?id=2965.
LG1.1 - SD: Platform Chip Description

Our platform chip has two ARM processors and two DSP processors. Each ARM has a local cache and both store their programs and data in the same offchip DRAM.

The left-hand-side ARM is used as an I/O processor and so is connected to a variety of standard peripherals. In any typical application, many of the peripherals will be unused and so held in a power down mode.

The right-hand-side ARM is used as the system controller. It can access all of the chip's resources over various bus bridges. It can access off-chip devices, such as an LCD display or keyboard via a general purpose A/D local bus.

The bus bridges map part of one processor's memory map into that of another so that cycles can be executed in the other's space, albeit with some delay and loss of performance. A FIFO bus bridge contains its own transaction queue of read or write operations awaiting completion.

The twin DSP devices run completely out of on-chip SRAM. Such SRAM may dominate the die area of the chip. If both are fetching instructions from the same port of the same RAM, then they had better be executing the same program in lock-step or else have some own local cache to avoid huge loss of performance in bus contention.

The rest of the system is normally swept up onto the same piece of silicon and this is denoted with the 'special function peripherhal.' This would be the one part of the design that varies from product to product. The same core set of components would be used for all sorts of different products, from iPods, digital cameras or ADSL modems.
A microprocessor logic symbol and simplified internal structure.

This device is a bus master: or initiator in ESL terms.
LG3.2c - SD: A D8/A16 Computer

Memory Map decoder circuit
Often a ‘PAL’ single chip device.

Data bus (8 bits)
Address bus (16 bits)

Register File (including PC)
Control Unit
Execution Unit + ALU
(Micro-)Processor

D0-7

1 K Byte ROM
Read Only Memory
A0-9
Enb

16 kByte Static RAM
A0-13
R/Wb

UART Serial Port
Rs232 Serial Connection

Memory
A0-7

UART ENABLE_BAR
RAM_ENABLE_BAR
ROM_ENABLE_BAR

R/Wb

LG3.2c -SD: Memory Address Mapping and Decode

<table>
<thead>
<tr>
<th>Start</th>
<th>End</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>03FF</td>
<td>EPROM</td>
</tr>
<tr>
<td>0400</td>
<td>3FFF</td>
<td>Unused images of EPROM</td>
</tr>
<tr>
<td>4000</td>
<td>7FFF</td>
<td>RAM</td>
</tr>
<tr>
<td>8000</td>
<td>BFFF</td>
<td>Unused</td>
</tr>
<tr>
<td>C000</td>
<td>C001</td>
<td>Registers in the UART</td>
</tr>
<tr>
<td>C002</td>
<td>FFFF</td>
<td>Unused images of the UART</td>
</tr>
</tbody>
</table>

module address_decode(abus, rom_cs, ram_cs, uart_cs);
    input [15:14] abus;
    output rom_cs, ram_cs, uart_cs;
    assign rom_cs = (abus == 2’b00); // 0x0000
    assign ram_cs = (abus == 2’b01); // 0x4000
    assign uart_cs = !(abus == 2’b11); // 0xC000
endmodule
LG3.3 - SD: : A Basic Micro-Controller

Introduced 1989-85.

Such a micro-controller has an D8/A16 architecture and would be used in a mouse or smartcard.
LG3.4 - SD: Switch/LED Interfacing

a) Example of electronic wiring for switches and LEDs.

b) Example of memory address decode and simple LED and switch interfacing for programmed I/O (PIO) to a microprocessor.
LG3.5 - SD: Programmed I/O

Input and output operations where a program on a processor makes read or write operations to the device.

Inefficient - too much polling for general use.

Interrupt driven I/O much better.

```c
#define IO_BASE 0xFFFFC1000 // or whatever
#define U_SEND 0x10
#define U_RECEIVE 0x14
#define U_CONTROL 0x18
#define U_STATUS 0x1C

#define UART_SEND() (*((volatile char *)(IO_BASE+U_SEND)))
#define UART_RECEIVE() (*((volatile char *)(IO_BASE+U_RECEIVE)))
#define UART_CONTROL() (*((volatile char *)(IO_BASE+U_CONTROL)))
#define UART_STATUS() (*((volatile char *)(IO_BASE+U_STATUS)))

char uart_polled_read()
{
    while (UART_STATUS() & UART_STATUS_RX_EMPTY) continue;
    return UART_RECEIVE();
}

char uart_polled_write(char d)
{
    while (!(UART_STATUS() & UART_STATUS_TX_EMPTY)) continue;
    UART_SEND() = d;
}
```
LG3.6 - SD: Interrupt Structure

Receiving device: Keep int enabled: dev interrupts when data ready.

Transmit device: Enable int when s/w output queue non-empty: dev interrupts when h/w output queue has space.

Enhancement: Vectored interrupt tells processor which device and priority.
Micro-controllers have a large number of GPIO. Platform chips have a few.

**Ex:** Show how to wire up a push button and write a device driver that counts pressed.
LG3.9 - SD: Keyboard Scanning Controller

In practice, scan more slowly and use extra register on asynchronous input pressed.

Could use a separate microcontroller to scan keyboard.

This keyboard scanner generates an interrupt on each key press.

Standard PC keyboard generates an output byte on press and release and implements a short FIFO.
Re-load register accommodates poor interrupt latency.

Timer (illustrated) : counts pre-scaled system clock.

Counter: counts external input pulses (e.g. car rev counter).

Four to eight, versatile, configurable counter/timers provided in one block.

All registers also configured as bus slave read/write resources.

Interrupt cleared by host programmed I/O to host_op.
LG3.10 - SD: Video Controller: Framestore

Uses private SRAM instead of main system RAM (share instead with staging FIFO?).

The pixel clock rate, H/W dimensions and synch pulse widths are normally programmable.

The RAM cannot be read. Moreover it has two address ports: re-code with one arbitrated port?
This controller just block copies: may need to keep src and/or dest constant for device access.

Typically, a multi-channel DMA controller is provided.

Or just use another (simple) processor?

DMA controllers may be built into devices: SoC bus master ports needed.

```verilog
reg [31:0] count, src, dest, datareg;
reg int_enable, active, intt, rwbar;

always @(posedge clk) begin
  if (hwen && addr==0) begin
    { int_enable, active } <= wdata[1:0];
    int <= 0; rwbar <= 1;
    end
  if (hwen && addr==4) count <= wdata;
  if (hwen && addr==8) src <= wdata;
  if (hwen && addr==12) dest <= wdata;
  if (active && rwbar && m_ack) begin
    datareg <= m_rdata;
    rwbar <= 0;
    src <= src + 4;
    end
  if (active && !rwbar && m_ack) begin
    rwbar <= 1;
    dest <= dest + 4;
    count <= count - 1;
    end
  if (count==1 && active && !rwbar) begin
    active <= 0;
    intt <= 1;
    end
  end
  assign m_wdata = datareg;
  assign m_ren = active && rwbar;
  assign m_wen = active && !rwbar;
  assign m_addr = (rwbar) ? src:dest;
  assign interrupt = intt && int_enable;
```
LG3.12 - SD: Network Device

Example: Ethernet, USB, Firewire, 802.11.
For high throughput should likely be bus master or use a DMA channel.
More importantly: DMA requires less staging RAM or FIFO in device.
Shared RAM pool: statistical multiplexing gain.
The device driver will set up a circular buffer or linked list of buffers.
LG3.13 - SD: Bus Bridge

Cycles slaved on one side are mastered on the other.

Need not be symmetric, or have flat address space.

Busses can be dissimilar.

Writes posted (internal FIFO).

(The ‘busses’ on each side use multiplexors and not tri-states.)

System bandwidth ranges from 1.0 to 2.0 bus bandwidth: inverse proportion to bridge crossing cycles.
LG3.14 - SD: Clock Tree

Clock sourced from a lower-frequency external (quartz) reference.
Multiplied up internally with a phase-locked loop.
Dynamic frequency scaling (future topic): programmable PLL ratio.
Skew in delivery is minimised using a balanced clock distribution tree.
Physical layout: fractal of H’s, ensuring equal wire lengths.
Inverters are used to minimise pulse shrinkage (duty-cycle distortion).
LG3.15 - SD: Clock Domain Crossing

Like a bus bridge, but different clocks on each side.

- Have one signal that is a guard or qualifier signal for all the others going in that direction.
- Make sure all the other signals are settled in advance of guard.
- Pass the guard signal through two registers before using it (metastability).
- Use a wide bus (crossing operations less frequent).

```verilog
input clk; // receiving domain clock
input [31..0] data;
input req;
output reg ack;
reg [31:0] captured_data;
reg r1, r2;
always @(posedge clk) begin
  r1 <= req;
  r2 <= r1;
  ack <= r2;
  if (r2 && !ack) captured_data <= data;
```

Simplex: can never be sure about the precise delay.

Need protocol with insertable/deletable padding symbols that have no semantic meaning.

100 percent utilisation impossible.

The four-phase handshake limits utilisation to 50 % (or 25 if registered at both sides)

Duplex: cannot reply on any precise timing relationship between the two directions. Protocol must rely on sequencing or explicit transaction tokens.