Chapter 1

Abacus-1 System-Level Design

This section contains relatively detailed design of the entire system, including sequencer, I/O interfaces, and host workstation interfaces. The topics of this section have largely been ignored by academic researchers, with most of the attention going to processing element and interconnection design. However, system interfaces are precisely those elements most likely to cause bottlenecks, and therefore no chip design is truly complete until its surrounding system is sketched out.

The Abacus-1 system consists of the sequencer, the PE array, the input board, the output board, and the video input board. They are coupled by a 32-bit bus.

1.1 Sequencer

The traditional role of the sequencer is to deliver microinstructions to the PE array. It has been necessary because word-level conventional instructions such as add or compare must be converted to several micronstructions. Also, the array typically executes much faster than

System Picture

the sequencer, due to the simplicity of its components. The sequencer acts as a bandwidth amplifeir between the host and PE array. The host specifies higher-level macroinstructions that are interpreted by the sequencer to produce the microinstructions of the array. The operational model is that the host computer executes scalar code on its own. When it is time to perform a parallel operation, the host sends a request to the sequencer to execute a block of instructions.

Although the simplest model of the sequencer is little more than an adress counter, and a microcode dispatcher, its complexity invariably grows. The first observation is that loops are common, and the extra latency of repeated block issues by the host is substantial. Thus, a loop counter is added. Unconditional loops become too restrictive, and the sequencer can now sample the global OR flag from the array. Conditional loops often require a single scalar value obtained from the array (such as the active pixel count) to be compared with some other quantity, and the overhead of transferring this value to the host for comparison is too high. Now registers are added to the sequencer, as well as simple program structure. The sequencer begins to resemble a conventional microcontroller.

This process is of course familiar as the wheel of reincarnation, as more and more functionality migrates between the main computing system and the controller. This migration occurs as a result of technological changes, as more silicon area becomes available, as new algorithms are developed, or as subsystem speeds change. It occurs in IO controllers, graphics controllers, vector processors, etc.

The Abacus-1 sequencer design is a new point in the evolutionary cycle. It is based on recognition of the following technological factors:

- Memory is cheap. There is no need to interpret instructions at runtime when the microinstructions themselves can be stored.
- Contemporary microprocessors already internally run faster than the PE array, and their external bus speeds are within a factor of two of array speeds.

The sequencer integrates the conventional roles of the host and sequencer. It executes

```
ADD  R3, R1, R2             ;  some scalar operatoin
MULT R4, R4, R5             ;  another scalar operation
LD   R0, 0xF1000000         ;  NETSOURCE := EAST
LD   R0, 0xF1000001         ;  NET_OUT   := P1
LD   R0, 0xF1000002         ;  P2        := P2 + NET_IN
LD   R0, 0xF1000003         ;  NETSOURCE := WEST
LD   R0, 0xF1000001         ;  NET_OUT   := P1
LD   R0, 0xF1000002         ;  P2        := P2 + NET_IN
ADD  R5, R3, R2
```

Figure 1-1: Example of interleaved scalar and parallel code. The parallel code implements a 2-point horizontal boxcar filter. Notice the instruction memory reuse. In actual code, the reuse will not be as frequent, since instructions are grouped in pairs.

scalar code intermixed with parallel code at a much finer level than before. Specifically, it executes parallel instructions by performing a read operation to a set of addresses. A specialized circuit watches the address bus, detects the read, and converts it to a parallel instruction address (PIADDR). The PIADDR is applied to the parallel instruction memory (PIMEM), which is then sent to the PE array. If no read request is issued during a particular cycle, the buswatcher sends a NOP (no operation) instruction to the array. This ability allows variable timing on the part of the sequencer due to cache effects, for example.

The sequencer consists of a conventional medium-speed DSP chip, the TI C40. It has uniformed cycle times as it does not use caches. The DSP chip is limited to clock rates of 40 MHz, leading to 3:1 blocking: for every instruction address, three instructions are fetched from the memory. Controller versions of modern fast microprocessors such as the PowerPC or the Alpha support 66 MHz bus speeds, and will allow 2:1 blocking.

1.2 PE Board

This section presents the design of a full 16 processor board. It includes a part count that demonstrates the uselessness of ECL.

Layer	Name	Orientation	Purpose
1	TOP	Random	SMT/via connectors.
2	V_{TT}	Plane	ECL termination voltage. Tied to 2V.
3	X1	Horizontal	Routing
4	Y1	Vertical	Routing
5	V_{CC}	Plane	Positive power supply. Tied to 5V.
6	X2	Horizontal	Routing
7	Y2	Vertical	Routing
8	V_{EE}	Plane	Negative power supply. Tied to 0V.
9	X3	Horizontal	Routing
10	Y3	Vertical	Routing
11	V_{CE}	Plane	1 V power supply. Tied to 1V.
12	BOT	Plane	Negative power supply. Tied to 0V.

Table 1.1: Board Stackup

Board Stackup. The board organization is the same as used in the testing environment. The organization is offset stripline, with X-Y pairs of routing between two planes.

The ECL circuits are operated in offset mode, between 0 and 5 volts, instead of XX and XX.

Bottom is covered by interposer clamp block, so no routing of PE chip traces is possible.

Top is currently unused because of propagation differences and robustness reasons, but could be used for routing clock signals and I/O signals.

Differential signals: Chip I/O, clock, global OR, length matched, run adjacent. Capacitance is increased, but whatever affects one trace, affects the other trace.

The PE board consists of 16 PE chips and associated SIMMS.

- Instructions are received on board by differential line receivers (6 quint receivers MC100E116). Each receiver fans out to four buffers.
- A row of PE chips shares 30 instruction wires. These wires are driven by single-ended ECL drivers (MC100E122, 9-bit buffers). There are four of these chips per row. The wires are terminated to a -2V (+3 in offset ECL) supply by 50 ohm resistors. These serve both as pulldowns and as line terminations.

Part	Desc	Qty	Comment
E122	9-bit buffer	16	4 chips/instruction bus
E451	6-bit D register, diff-in	3	1 reg/ chip (GOR)
		3	1 reg/ chip (DATAI)
E452	5-bit D register, diff	4	1 reg/ chip (DATAO)
E101	Quad 4-input OR/NOR	5	1 gate/chip + 1 to sum (GOR)
E116	Quint diff receivers	6	30 instruction bits
Total		34	

Table 1.2: PE Board: ECL Chip Requirements

- The ECL reference voltage is produced by a MC100E111 clock chip.
- Global OR differential inputs go to 3 6-bit D registers (MC100E451). These feed into a quad 4-bit OR, which feeds into another 4-bit OR. The outputs of that gate are differential and are sent offboard.
- DataO differential outputs go to differential D FFs (MC100E452). Differential outputs from the D regs go offboard.
- DataI inputs arrive differentially from off-board into 6-bit D registers (MC100E451) and are sent to the PE board single-ended.

Could this be reduced with custom chips? A single custom glue chip would have 60 outputs and 30 inputs for instruction delivery; 8x2 = 16 inputs and 16 outputs for DATAOUT; 16 inputs and 2 outputs for GLOBALOR; 16 inputs and 8 outputs for DATAIN. This adds up to 164 outputs. We would need two such chips for the entire board.

Power Consumption. EDRAM parts are estimated to have a static column read current of 105 mA/chip at 16ns cycle times, or about 500 mW. 16 of these chips, forming two SIMMS, consume 8 Watts.

1.3 I/O Boards

This section describes the design of the input and output boards. Since the boards are fairly symmetrical in design, only the output (array to sequencer) board is described.

The first challenge is to receive the large number of ECL level signals from the PE boards. There are 16 signals per board and 16 boards in the system, for a total of 256 signals at a data rate of 125 Mbits/second. The received signals must then be converted from a bit-serial to bit-parallel form. This is more complicated than a simple serial-to-parallel conversion since a single data word is scattered across several serial transmissions (four in the case of a 16-bit four by four arrangement). After the pixels are reformatted, they must be stored in memory as to appear as a linearly accessible frame buffer, so that efficient loops on the sequencer can access them.

All IO from a group of 16 chips comprising a board is processed by a collection of circuitry called a reformat unit. A munge unit is composed of a 32-deep, 16-wide ECL FIFO, two ECL/CMOS level converters, a SHMUX unit, and two SRAMs. The SHMUX is composed of two FPGAs and is effectively a chain of shift registers and multiplexers. The multiplexer outputs are connected to two SRAM chips. The reformatting proceeds as follows:

1. The FIFO is enabled for input and reads in the 32-bit burst.

2. The FIFO is enabled for output. The ECL signals are converted to CMOS levels and entered into the munger.

3. The controller sequences the multiplexers inside the mungers to select the appropriate bit groups for output. At the same time, appropriate addresses are presented to the SRAM so that the bitwise reformatted data are image-wise reformatted.

Notice that the data transfer is pipelined, as the FIFO can be loaded at the same time as the SHMUX is trasferring data to the SRAMs.

The 16 reformat units on the O board are controlled by the format controller. The controller consists of a small state machine, a counter, and an address lookup table. The table contains addresses to be presented to the SHMUX units for appropriate reformatting.

The sequencer accesses the board memory by presenting the desired address to the format controller, which maps the address to the appropriate SRAM chip based on the prepro-

Chip	Quantity
Differential receivers	4
FIFO	1
Level convert	3
FPGA	2
SRAM	2
Total	12

Table 1.3: Chip requirements, output board. Notice that if output enable inputs are not available, several tristate buffer chips would be required.

grammed image size table. This address translation adds two cycles of latency to sequencer memory accesses, but allows fast random access.

The total chip count for the board is 192 plus the 8 or so chips implementing the format controller for a total of 200. Assuming 1 inch chip pitch, and a double-sided surface mount design, each board is about ten inches on a side, which is quite reasonable.

The input board symmetrically reverses the reformatting process by loading the SHMUX chips from the SRAMs, loading the SRAM from the other side, and then executing burst transfer operations. If need be, the functionality of the two boards can be combined into one with a few multiplexers and extra wiring demands.

We estimate that a single custom VLSI chip can replace two reformat units, for a substantial improvement in integration and cost. The major limitation is the number of pins required: 16 input pins and 32 output pins.