A High Performance VLSI CMOS Arithmetic Processor Chip

Dr. Glenn Culler

CHI Systems, Inc.
Goleta, California

Ed Greenwood
Dave Harrison

Motorola Inc., Government Electronics Group
Scottsdale, Arizona

INTRODUCTION

The described Arithmetic Processor Unit (APU) uses the efforts of Motorola's three micron silicon gate CMOS semiconductor fabrication technology and CHI System's array processor architecture technology. CHI Systems Inc. has developed architectures for array processor systems (as well as the hardware and software elements) for over ten years. In 1979, CHI Systems was developing their fifth iteration of an array processor system, called the CHI-5.[1] The U.S. Defense Advanced Research Projects Agency (DARPA) was sponsoring development of the CHI-5. Motorola was seeking to use the CMOS process for low-power and medium-speed linear algebra processors. DARPA proposed the natural combination of these two technologies. Emphasis of this program, sponsored by DARPA, has been to obtain the best deliverable silicon gate CMOS APU capability for a fixed investment. Thus, yield and production cost have been secondary factors. Processing throughput and maximum functions obtainable on the chip were desired features.

[1]Patented - U.S. Patent Number 4,287,566 dated Sept. 1, 1981

FUNCTIONS ON CHIP

The APU architecture was derived from the CHI-5 system. The CHI-5 uses much parallel computation, thus address calculations and data operations are performed in a single system clock cycle. The APU uses three primary 16-bit data busses: bidirectional X data bus, bidirectional Y data bus, and unidirectional R bus. These three busses are shown in Figure 1. The CHI-5 system has a distinct 16-bit memory structure associated with each of these busses. The R bus fulfills the need for a coefficient read-only bus. Three specialized 16-bit adders (F, G, and H) are used in conjunction with four 16-bit accumulator registers (T, U, V, and W), as shown in Figure 1.

The multiplier is a single pipe delay multiplier. Hence, the input multiplier registers (MA and MB) and the 33-bit concatenated output product registers (MPL and MPR) are provided, as shown in Figure 1. MPL contains the most significant bits. The multiplier performs integer and fractional two's complement arithmetic. The fractional two's complement operation provides either a 31-bit product or a 16-bit convergent rounded product. In

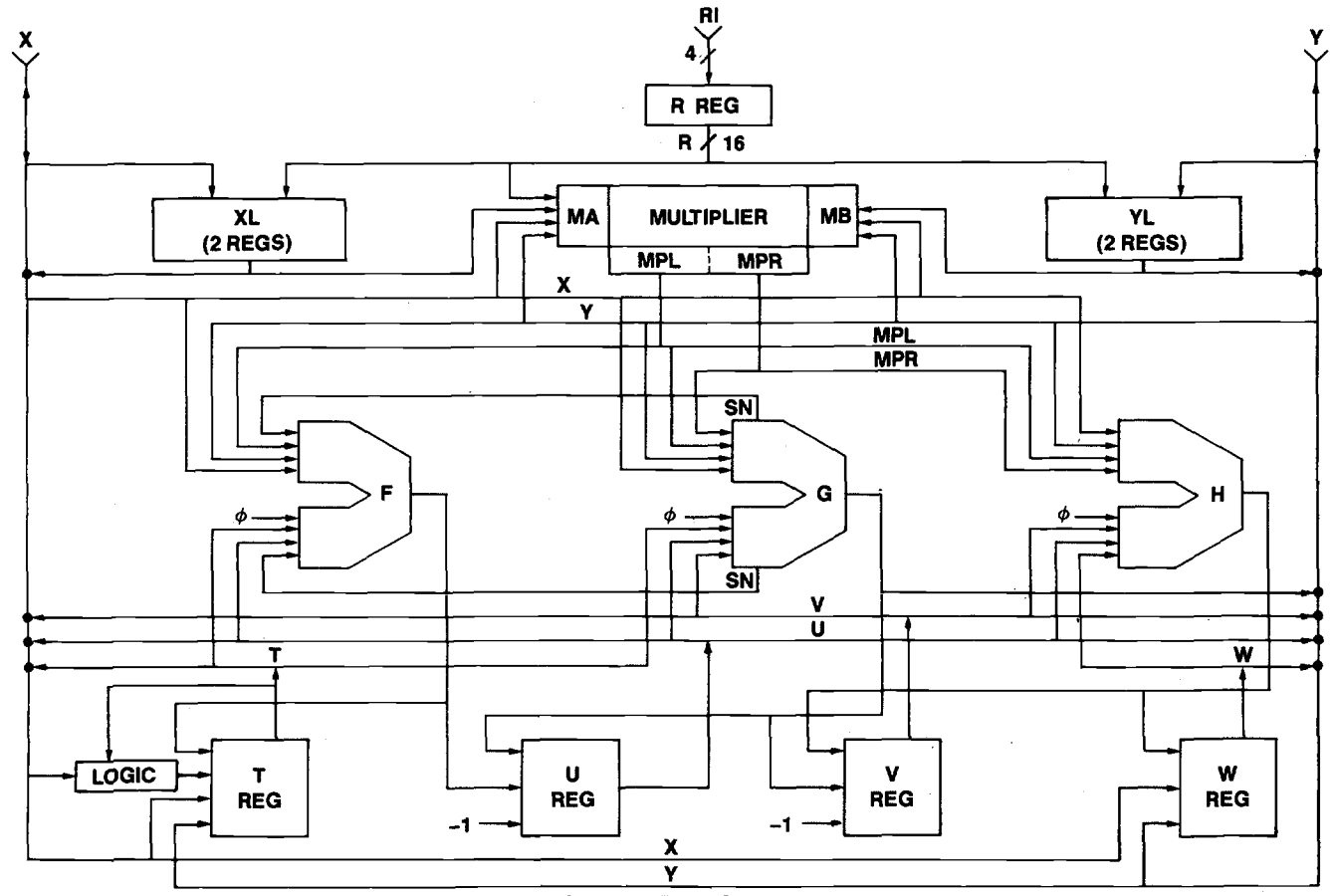

Figure 1. Basic APU Structure

CH 1746-7/82/0000 - 1053 $ 00.75 © 1982 IEEE

order to expand (via microcode) the multiplier's capability to more significance, special types of multiplier arithmetic are provided in the hardware. The integer and fractional multiplier modes provide multiplication of a 16-bit two's complement number times a 16-bit positive binary number. The integer mode also allows multiplication of two 16-bit positive binary numbers. All of the above types of multiplication are performed in a single APU instruction cycle.

Most components of the APU are interconnected to one or both of the internal busses (X-Y as shown on Figure 1) for input and/or output. In addition, there are a series of connections to the multipliers and adders which do not use the X-Y busses. In particular, the multiplier has inputs from its local registers (XL1, XL2, YL1, and YL2) and the R bus (through a single pipe delay register). The adders have inputs on the A side from the multiplier registers MPL and/or MPR; and the adder's B side can select from a subset of accumulator registers. Each adder's sum can feed two specific accumulators directly; the G adder can also drive the Y bus. The interconnection structure of the adders and accumulators is such that they can be used to provide an internal pipeline configuration of up to four stages, capable of being fed by either the X or Y bus and being read by the opposite bus.

This powerful internal topology of register/adder connectivity provides the microcode designer with the structure to optimize an array processing task for maximum throughput that utilizes only one multiplier. Thus, the test of utilization of this structure for the microcode designer is that one microcode cycle per multiply operation for inner loops can be achieved.

The additional features of the adders allow coupling for 32-bit and 48-bit arithmetic operations. Also, each adder contains an extra bit (or 17th bit). This extra bit provides a conditional shift feature. Testing and correction of adder overflows are provided by this capability. When overflow occurs, the 16th and 17th bit shall have different values. When a conditional shift add is programmed, the overflow condition is checked after the add operation; and then the result is right shifted one bit if an overflow has occurred. The overflow condition is latched into a 1-bit register for use by the APU. This feature is useful in floating point number operations. A logic operation is also provided into the T register, using the X and T values for inputs.

Several specialized logic functions are also provided to enhance algebraic array processing and floating point arithmetic. These features are unary operations, which are called RSP, DSCL, LSP, SIZE, DECM, and TEST.

The TEST logic evaluates the G adder's output, the H adder's output, or the GH adder's concatenated output. The logic determines if the data is less than zero or equal to zero. The conditions are output from the chip. This feature allows the branch on conditional data operation. The DECM function performs a 16-bit decimate (bit reversal) of the U register onto the X bus. Certain types of transforms on arrays of data will find this function useful for address calculations. A SIZE register allows the microcode designer the capability of deriving the most significant bit position seen on an array of values in the U register, V register, or concatenated UV registers. This function is useful for block scale floating point operations. The LSP operator provides the multiplier required to normalize a value in the concatenated UV register (obviously U itself is an available operation). The DSCL operation provides a count of the number of places shifted by LSP. Hence, DSCL provides the exponent delta value. The TEST function can also test DSCL's output to determine if the LSP value was determined from the most significant 16 bits or the least significant 16 bits (for double precision operations). The RSP operator provides a multiplier required to align two floating point mantissa's. It assumes that W (used to generate the RSP value) contains the positive difference of the two numbers' exponents. This feature eases the addition of floating point numbers.

Address controllers for the X and Y memory elements are provided on the APU chip. Two identical 16-bit data controllers (for XA and YA) are used. Two 16-bit adders and two 16-bit registers are used in each controller. The block diagram of the X address controller (XA) is shown in Figure 2. This address controller structure provides a base address (Y) with an index (XC) and an offset variable (XD). The increment/decrement function provides for column displacement for two-dimensional array addressing directly. Thus, two-dimensional mask convolution direct addressing is provided, as shown in Figure 3. Mask size is limited to twice the maximum value of XD plus one (nine in the APU). The choice of X or Y data as the source for XC allows X memory to be used as a buffer for arrays that have dimensions greater than two. Of course, the Y address controller can be used by the microcode designer for an additional simultaneous address controller.

Note that the combination of the X and Y address controllers and the data arithmetic unit required isolation of the external X bus from the APU X bus (called XS). And similarly, a YS bus exists, isolated from the Y bus. Thus, external memory operations (in conjunction with the address controller can exist while internal APU data processing uses the extension of the X and Y busses (XS and YS).

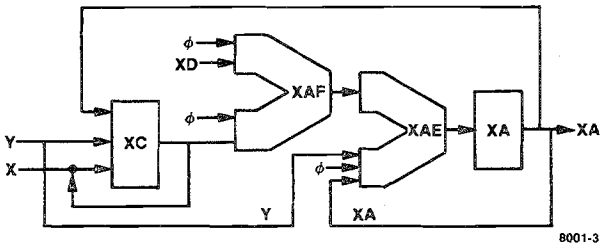

Figure 2. X Address Controller

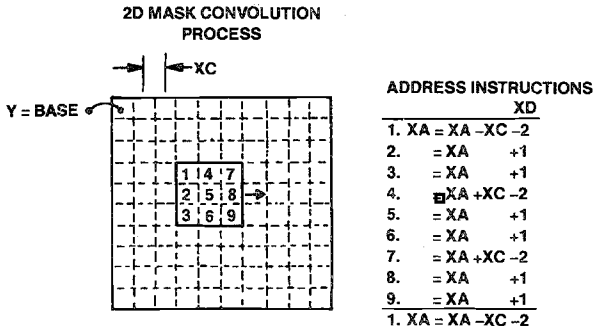

Figure 3. 2D Mask Convolution Addressing 8001-4

RATIONALE FOR CHIP FUNCTIONS

Obviously, the APU chip was designed to implement the basic multiply/add/accumulate functions. But the local mutliplier storage (XL/YL), the special unary logic functions, and the address controllers were designer choices for implementation on the chip. The XL/YL registers were a high priority for implementation so that bus contention would not limit throughput. We desire to use the multiplier with 100 percent efficiency in inner processing loops. The only question was the number of registers to be implemented. Two of each was found to be adequate for benchmarks that were reviewed. Similarly, the special logic functions were a high priority since they directly affected inner loop processing throughput.

The address controllers for X and Y memory were certainly an appendage to the arithmetic processor. They were considered for implementation, along with on-board program memory (RAM or mask ROM) or X/Y memory RAM or a system controller. This was the primary tradeoff area for on-chip versus off-chip functions.

Obviously, the chip contains no RAM or ROM memory, a popular function on newer processor chips. X, Y, R, and program memory were considered for implementation on the chip. However, the rationale for no memory on the chip was as follows:

1. The semiconductor industry will continue to provide denser and faster CMOS memories (PROM, ROM, and RAM).
2. Build a chip that will take advantage of Item 1 above, and do not let the chip built be outdated because of Item 1 above.
3. Provide an APU such that the application designer can utilize the type and amount of memory as required by the application.

Thus, we envision a system that could use PROM, ROM, or RAM program memory. Obviously, the RAM program memory would have to be down-loaded by a host. Since 80 bits are used for APU instruction, a 96- to 128-bit wide instruction word is envisioned for systems. Obviously, not many program micro instructions (i.e. <100) could be put on the chip.

X/Y data memory on-board the chip would be a desirable feature. But if some X/Y memory is on-board and some is off-board, then the system application designer would be constrained to use a fixed type of RAM. We envision the use of either NX1 or NX8 RAM's (or maybe NX16 in the future) for different applications.

If the system controller (program control, DMA, and I/O control) were added to the chip, the pin count would have been greater than 150 and there

would not be area for the address controllers. Since program controllers and DMA controllers were being offered by semiconductor firms, but data controllers were not available, we chose to implement the X/Y data controller function.

DESIGN LAYOUT METHODOLOGY

The APU was divided into five design sections: multiplier, adders and registers with test, float and size logic, XL/YL registers, address controllers[2]; and the control and decode logic for each of the previous sections. A photograph of metal and poly layers with partitioned data busses and logic sections is shown in Figure 4.

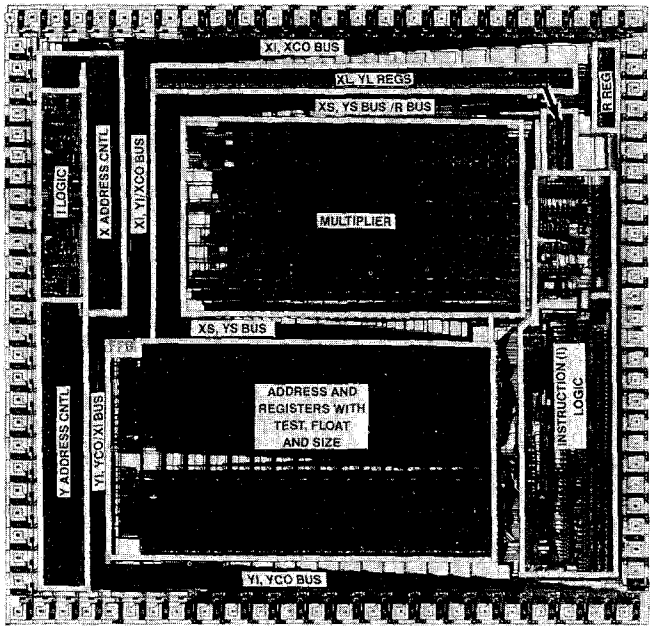

Figure 4. APU Chip Layout

Each address controller was designed in a bit slice arrangement that was repeated 16 times for each address bit. The slice for the address controller has two custom cells.

The XL/YL registers required one custom cell to be drafted. This section is simple to lay out, but its placement in the APU chip is important because the section has considerable bus interfaces.

All control and decode sections contain the 4-to-1 time division demultiplexing of a 20-bit-wide input instruction field into an 80-bit-wide instruction field. This multiplexing of the instructions was used to reduce chip pin count. This 80-bit-wide word is decoded into 184 control lines that control the APU operation.

The control logic is designed in three sections, using a standard cell approach with a minimum number of cells. These cells were designed to drive the input capacitance of the large buffers. Only four cells are used in this section. These buffers were designed to have as low a profile as possible so as to match up in vertical layout height to the large number of control and clock lines in the bit slice, thus lowering the area of interconnection between them.

Table I summarizes the allocation of chip area dedicated to the logic and busses for the APU chip.

The 17 × 17 single cycle pipelined array multiplier uses the Burks, Goldstein, Von-Neumann algorithm implemented in an array tree. The multiplier has 241 full adders, 31 half adders, 256 nand gates, and 32 exclusive-or gates. The multiplier layout required 13 macro cells, which used a fast array adder as the basic building block. This adder cell, with its programmability, is used throughout the APU whenever a full adder is required. Five of the 13 cells were custom designed (drafted at 1000X), while the remaining eight cells were modifications of the basic five and were created on a CALMA graphics system. In the layout, the partial products flow downward through the layout on polysilicon. Approximately 7 percent of the multiplier area is for interconnection of the array tree. The power, control, and data shift lines are routed on metal horizontally through the layout.

The adders and registers with special unary function logic was the most challenging layout because it has a large number of busses. To minimize the effects of the busses on the chip area, the logic was designed in a bit slice arrangement. The inputs and outputs of the bit slice are XS data bus, YS data bus, YI (XI and YI are the X/Y busses that are used by the address controllers) data bus coming from the bond pads, and the control lines coming from the control and decode logic.

The bit slice has 22 macro cells stacked vertically and the stack is repeated 16 times across the layout. Then with considerable modification, a 17th bit slice is added to the MSB end. This section required a total of 33 macros, 21 of which were explicitly designed while the other 12 were modifications of the basic bit slice macros.

Like the multiplier, the data flow is vertical on polysilicon while the power busses and control are horizontally routed in metal. The main data busses in the bit slice are XS, YS, MPL, and MPR. This is in comparison to the internal data busses for T, F, U, G, V, W, and H; which route vertically on polysilicon between macros and do not run the entire height of the bit slice. The main data busses do run from top to bottom of the bit slice. The bit slice also has a large number of control lines and power busses which run in

Table I. Allocation of Chip Area

Function	Area (square mils)	Number of Transistors	Density (square mils/transistor)	% of Chip Area
Multiplier	15484	10349	1.50	16
Bit Slice	18849	10000	1.88	21
XAC	2584	1952	1.32	3
YAC	2584	1952	1.32	3
XL, YL	2553	1824	1.34	3
Control	10269	3506	2.94	11
I/O (Bond Pads)	18038	978	18.44	20
Data Busses	15022	0	0	17*
Control Busses	2444	0	0	3*
Available Area	2770	-	-	3
Totals	90597	30561	2.98	100

*Does not include busses within each section.

metal across its width. The four main data busses alone account for 11 percent of this function's area.

Review of the layout (after completion) has allowed us to better critique our design approach. We have estimated that a fully custom layout (with the same design rules) would have resulted in a chip with 80 percent of the present area. However, this would have required at least twice the design man-hours. Had we used standard cells with computer placement and routing tools (that are known to us today), the chip size would not have been feasible to build. Future tools being discussed and their utilization on a chip of this type have yet to be proven and are good for present-day coffee break soliloquys.

The 100-pin device is expected to have a power dissipation of 250 milliwatts at a cycle time of 250 nanoseconds. The design goal for the clock cycle time is 200 nanoseconds. The power supply voltage is 5 volts.

BENCHMARKS

A set of linear algebra processes that represent the type of applications envisioned for the APU were coded. The dot product (or inner product) for a 32-bit result or for a 48-bit result in fixed point arithmetic or block scale floating point arithmetic can be performed in one clock cycle per data point. Sixteen-bit mantissa and 32-bit mantissa floating point operations were evaluated. All floating point operations use a 16-bit exponent.

A portion of the processes that were benchmarked are shown in Table II. All execution times are based on a 250-nanosecond cycle time.

Table II. Process Benchmarks

Process	Execution Time
32-bit mantissa floating point addition	1.5 to 2.5 μs
32-bit mantissa floating point multiply	0.75, 1, or 1.75 μs
Complex conjugate pole-zero pair filter with gain	1.25 μs/point
512 complex point FFT	2560 μs
512 real point FFT	1152 μs
DARPA speech LPC analysis, pitch analysis, and synthesis (total)	$\leq$5263 μs

SYSTEM APPLICATIONS

Starting with the APU chip, one can design computer systems literally of any type presently known and use no other arithmetic facility than that provided by the APU. That is to say, it is a true part, not just a piece of a special purpose system that fell short of the mark of placing the entire system on one chip.

The small-scale microprocessor-based computers that form the basis of the personal computer market provide user communication that is very successful for program preparation and small-scale problem solving. But when any significant scientific computing is required, the arithmetic capability is simply not adequate. An enormous enhancement of performance will result by using the APU as an accelerator.

Special purpose computers are organized by the concept of performing particular algorithms with minimum hardware. Because their design must be so tightly coupled to one algorithm, a separate design is required for each case of interest. Consequently, the ease of carrying out such a design is an important issue in the cost/effectiveness of such systems. But with the 16-bit microprocessors presently available, together with the APU chip and wide-word RAM's, such designs will be achieved by just connecting major components in simple configurations.

ARRAY PROCESSOR APPLICATION

At the low end of the performance scale, we envision array processor systems with a small inventory of supporting memory chips controlled by off-the-shelf sequencers. To keep the memory chip count low, successive reads for micro-instruction source will be used, as well as successive reads and writes for the vector memories.

In the intermediate range of performance, a full complement of supporting memory chips will be used, with "size" determined by how many words are available in each of the memory modules:

X,Y	Vector memories
PH	Micro-instruction ROM
PS	Micro-instruction RAM
D	System memory and R source data

The high-performance capability can be achieved by using several APU chips to provide higher level parallelism of arithmetic operations and to extend arithmetic precision. Of course, even more memory chips are required to keep the memory accesses up to the extended APU performance.

To achieve the super-performance level, we consider micronets of array processors with each node having general purpose control, a local operating system, and a micronet communication capability over a multiple connected network.

An array processor system is shown in Figure 5 that utilizes the APU chip. The system employs the APU, two array memories (X and Y), a partitioned data memory (DR and DL) that could be shared with a host computer, a system controller, transceivers, and a clock. The system controller would provide program controller functions, memory control functions, and I/O control functions.

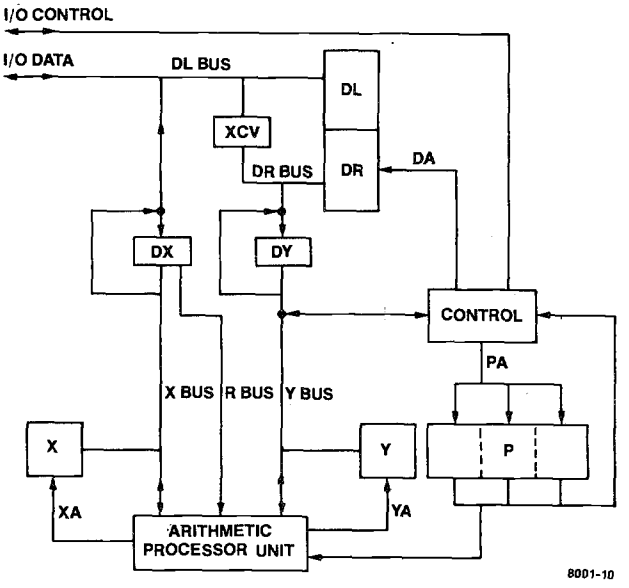

Figure 5. Array Processor Module