Elite Engineering Inc.
Pinpoint Engineering Inc.

August 2006

The BLITZCORE Processor

Introducing the Blitzcore™ Processor

- **Revolutionary Bus Design.** Beyond the limitations of the front side bus (FSB), the Blitzcore utilizes a new design for ultra fast throughput.
- **Chip Level Parallel Processing.** Solving the issues of parallel processing on a chip level.
- **Combinatorial Engine.** This proprietary logic circuit optimized for real-time compression and encryption can run 30 to 40 times faster than a typical processor because it is not waiting for instructions.
- **Applications for the Intelligence Community and Homeland Security.** High speed imaging, compression and encryption with real time analytics and advanced autonomy algorithms for Unmanned Aerial Vehicles (UAVs), video management systems, and the aerospace industry.

TABLE OF CONTENTS **PAGE**

Executive Summary 3

Blitzcore Processor 4

Blitzcore ALU 5

Parallel Processing 6

Combinatorial Engine 12

Manufacturing Process 16

Prototypes 20

HD-UAV ....

Security DVR ....

Wireless HD ....

HD-IPTV ....

Executive Summary

Elite Engineering Inc. is a research and development firm based in Colorado. The firm develops intellectual property (IP) cores licensed to new projects and established companies based on applications of the proprietary Blitzcore™ Processor technology.

Elite was organized to develop and produce IP cores and application specific integrated circuit (ASIC) chip design in four key areas:

- defense and military applications
- media applications
- wireless technologies
- medical imaging

The firm's area of expertise is ultra high speed digital design. The background of the firm's engineers includes development within the strict tolerances of the aerospace industry and the the diligent requirements of medical projects requiring FDA-audits.

The Blitzcore Processor was developed in partnership with Pinpoint Engineering. The design uses aspects of chip based parallel processing and a revolutionary internal bus design that eliminates bottlenecks and enables tremendous data throughput.

The firm is developing a unique parallel processing technique in its discret logic design for real-time compression and encryption. The firm is solving key issues of cache coherency related to parallel processing on a chip level. The Blitzcore doubles as a RISC core processor with a limited number of instructions to handle any type of general purpose processing.

The firm is currently considering several candiates for their board of directors and advisory panel. The management of Elite is also being assembled.

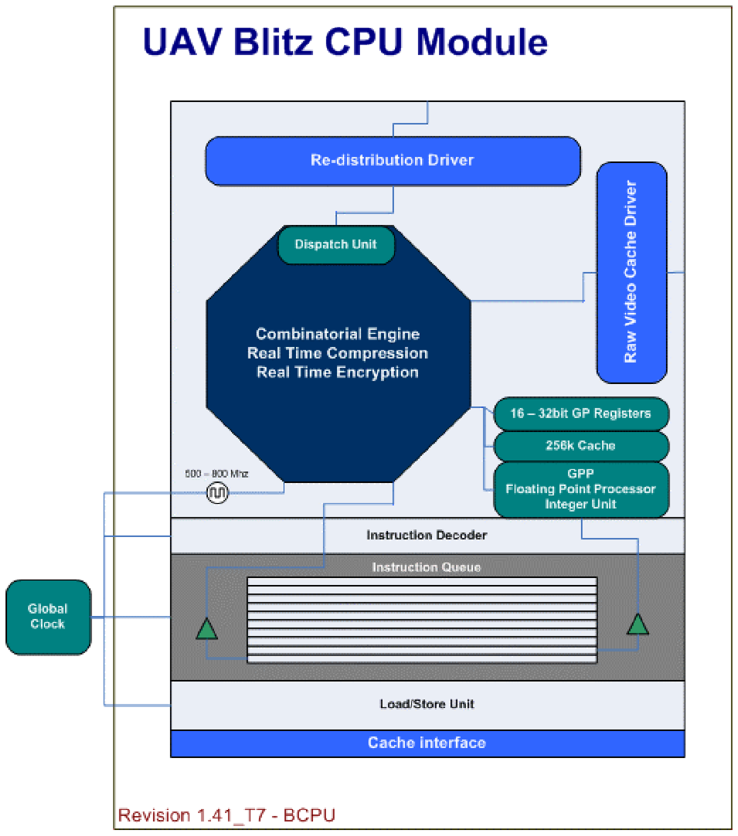
Revenue comes from project management fees, firm development fees, licensing fees from intellectual property, and ownership in companies launched.

Blitzcore Processor

The Blitzcore central processing unit (CPU) is an ultra high throughput chip design developed to process enormous amounts of data on a order of several magnitudes greater than other microprocessors. The Blitzcore utilizes 6 to 16 separate uniquely modified RISC core processors.

The processors today still employ the same Harvard Architecture method of processing data and uses a single bus to transport raw data and compressed data to and from its locations. The processor while becoming more and more complex will still always lack the type of throughput that our ASIC chips can provide. The frontside bus is the gatekeeper to the chip, requiring all data to pass through it.

The Blitzcore uses an instruction set specifically designed for processing raw video with a limited number of clock cycles. Lowering the number of clockcycles is an important factor in increasing the speed and efficiency of the design.



Pinpoint Engineering Inc.

Blitzcore ALU

The Blitzcore CPU utilizes a proprietary arithmetic logic unit (ALU). A typical ALU of a computer's CPU is a part of the execution unit, a core component of all CPUs. ALUs are capable of calculating the results of a wide variety of basic arithmetical computations. Virtually all modern computer ALUs use the two's complement binary number representation. Early computers used a wide variety of number systems, including one's complement, sign-magnitude format, and even true decimal systems, with ten tubes per digit.

Mathematician John von Neumann proposed the ALU concept in 1945, when he wrote a report on the foundations for a new computer called the EDVAC (Electronic Discrete Variable Automatic Computer). Later in 1946, he worked with his colleagues in designing a computer for the Princeton Institute of Advanced Studies (IAS). The IAS computer became the prototype for many later computers. In the proposal, von Neumann outlined what he believed would be needed in his machine, including an ALU.

Von Neumann stated that an ALU is a necessity for a computer because it is guaranteed that a computer will have to compute basic mathematical operations, including addition, subtraction, multiplication, and division. He therefore believed it was "reasonable that [the computer] should contain specialized organs for these operations".

Most of the computer’s actions are performed by the ALU. The ALU gets data from processor registers. This data is processed and the results of this operation are stored into ALU output registers. Other mechanisms move data between these registers and memory. A Control Unit controls the ALU, by setting circuits that tell the ALU what operations to perform.

Most ALUs can perform the following operations:

- Integer arithmetic operations (addition, subtraction, and sometimes multiplication, though this is more expensive)
- Bitwise logic operations (and, not, or, xor)
- Bit-shifting operations (shifting or rotating a word by a specified number of bits to the left or right, with or without sign extension)

Many standard ALUs do not handle integer division or any floating point operations since they can be emulated in software. However, several algorithms do exist for implementing division in hardware. The Blitzcore CPU can handle all math functions including floating point operations.

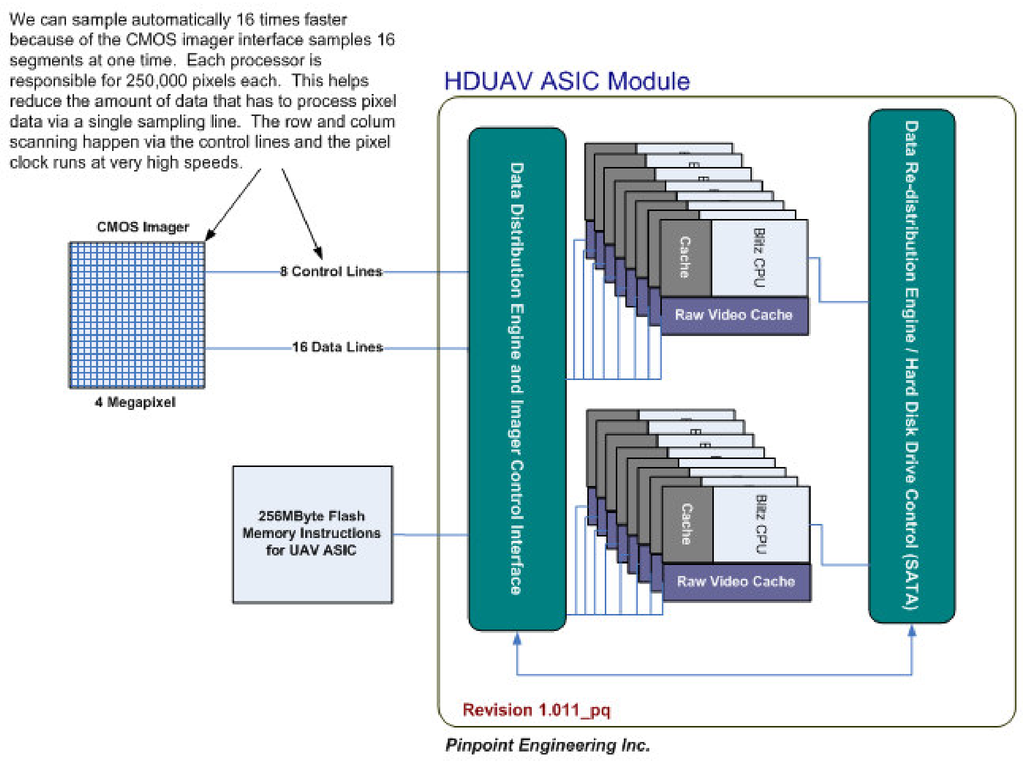
The ALU inside the Blitzcore Processor and the Combinatorial Engine are designed specifically for high throughput of special codec algorithms. These designs are currently being tested and updated everyday before we apply for our patents in this area. The ALU’s are always designed by different CPU manufacturers to suit their specific needs. We have done the same but we maintain that our ALU can run 2 – 4 times faster than a typical CPU’s ALU. We attribute this to the unique design of getting the information to the ALU faster and calculating the data quicker.

Parallel Processing

The Blitzcore Processor is an IC level parallel processor but we design the Blitzcore Processor into all of our ASIC chips based on our specific customer needs. For example the ASIC Chip for the UAV utilizes 16 Blitzcore Processors to increase the Compression and Encryption Process. We consider the Compression Speed to be real time. We are able to process data from a CMOS Imager at a rate of 100 – 10,000 Frames Per Second (FPS) depending on the electronic shutter speed of the camera. Most Cameras that process data at this speed are not able to be kept up with by conventional embedded systems or high-speed PCs. We are currently designing a UAV Chip that will utilize Pinpoint Engineering's proprietary Blitzcore Processors and the front-end design will utilize a Patent Pending Camera Interface owned by Elite Engineering and Pinpoint Engineering Inc. This front-end design directly interfaces to a CMOS imager that we are currently designing with a manufacture of this type of device. Our proprietary CMOS Imager will allow us to make full use of our processor high-speed compression/encryption technique.

The chip has a very specific technique to process video data at a high rate of speed. This chip will ultimately be very useful in many areas in the aerospace industry. The main target for this project is the UAV market.

The combinatorial engine that Pinpoint Engineering has developed will be developed into prototype "B" for demonstrations for our larger customers and clients. The combinatorial engine is explained in its own section. Each Blitzcore processor has its own engine optimized for sampling and compressing video data from a CMOS imager.


We can sample automatically 16 times faster because of the CMOS imager interface samples 16 segments at one time. Each processor is responsible for 250,000 pixels each. This helps reduce the amount of data that has to process pixel data via a single sampling line. The row and colum scanning happen via the control lines and the pixel clock runs at very high speeds.
HDUAV ASIC Module
CMOS Imager
4 Megapixel
8 Control Lines
16 Data Lines
256MByte Flash Memory Instructions for UAV ASIC
Data Distribution Engine and Imager Control Interface
Cache
Blitz CPU
Raw Video Cache
Cache
Blitz CPU
Raw Video Cache
Data Re-distribution Engine / Hard Disk Drive Control (SATA)
Revision 1.011_pq
Pinpoint Engineering Inc.

The Blitzcore CPU is our main processor that processes all the data but is given all of its instructions on what to do by the Distribution Engine. The Blitzcore CPU has very limited instructions that allow it to be optimized for use with the UAV ASIC. Each Blitzcore CPU has the capability of running almost all Codecs used on the market today. We will evaluate our final customers needs and find which compression technique they prefer the best and then optimize the instructions to handle the Codecs our customers require.

The Distribution Processor section of the UAV ASIC is far more complex than it looks in the block diagram. Its responsibilities include sending out the information from the CMOS Imager, Configuring all processors to be RISC or Combo Engine, Control of CMOS Imager and run embedded programs from flash memory. This processor is not a typical microprocessor it has a proprietary instruction set that is more dense then the RISC Processor because of its duties. It can be programmed to run like a general purpose processor, although the amount of instruction is not as large as and IBM™ Processor is lies somewhere between our RISC and their processor. The CMOS Imager is a modified semiconductor device that is designed specifically for use with our UAV ASIC Chip. It is capable of being sample in 16 separate locations at one time thus speeding up our frame rate immediately by 16 times. Each Blitzcore CPU has the capability of handling direct data from the CMOS Imager but the data is facilitated by the Distribution processor. It will receive instructions on when to start sampling the data and it knows what to do automatically for compression and encryption.

Concurrency was first exploited in computing to better utilize or share resources within a computer. Modern operating systems support context switching to allow multiple tasks to appear to execute concurrently, thereby allowing useful work to occur while the processor is stalled on one task. This application of concurrency, for example, allows the processor to stay busy by swapping in a new task to execute while another task is waiting for I/O. By quickly swapping tasks in and out. Giving each task a "slice" of the processor time, the operating systems can allow multiple users to use the system as if each were using it alone (but with degraded performance).

Most modern operating systems can use multiple processors to increase the throughput of the system. The UNIX shell uses concurrency along with a communication abstraction known as pipes to provide a powerful form of modularity: Commands are written to accept a stream of bytes as input (the consumer) and produce a stream of bytes as output (the producer). Multiple commands can be chained together with a pipe connecting the output of one command to the input of the next, allowing complex commands to be built from simple building blocks. Each command is executed in its own process, with all processes executing concurrently. Because the producer blocks if buffer space in the pipe is not available, and the consumer blocks if data is not available, the job of managing the stream of results moving between commands is greatly simplified. More recently, with operating systems with windows that

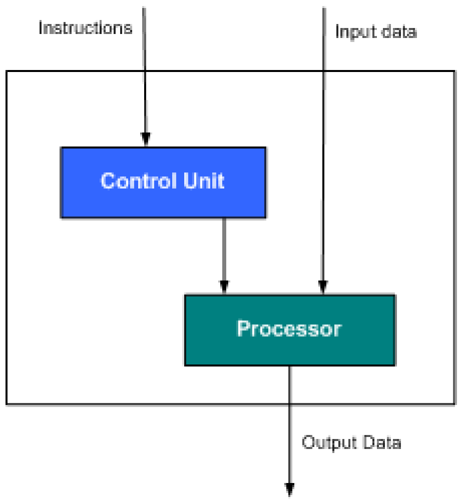
Invite users to do more than one thing at a time, and the Internet, which often introduces I/O delays perceptible to the user, almost every program that contains a GUI incorporates concurrency.

Although the fundamental concepts for safely handling concurrency are the same in parallel programs and operating systems, there are some important differences. For an operating system, the problem is not finding concurrency-the concurrency is inherent in the way the operating system functions in managing a collection of concurrently executing processes (representing users, applications and background activities such as print spooling) and providing synchronization mechanisms so resources can be safely shared. However, an operating system must support concurrency in a robust and secure way: Processes should not be able to interfere with each other (intentionally or not) , and the entire system should not crash if something goes wrong with one process. In a parallel program, finding and exploiting concurrency can be a challenge, while isolating processes from each other is not the critical concern it is with an operating system. Performance goals are different as well. In an operating system, performance goals ar normally related to throughput or response time, and it may be acceptable to sacrifice some efficiency to maintain robustness and fairness in resource allocation. In a parallel program, the goal is to minimize the running time of a single program.

Our parallel processing technique is very unique in that the I/O for accessing the CMOS Imager data is initial stage of compression. The front-end of our processor is setup to scan the data directly into the first stage compressed raw video cache. Each processor goes into the second stage of compression on its own particular real-estate for 1/16th of the 4 Megapixel CMOS Imager. The compression technique is so fast that it allows for the use of the word "real-time". However real-time is something that we are far beyond in terms of throughput and managing and analyzing the data available to us from the CMOS Imager.

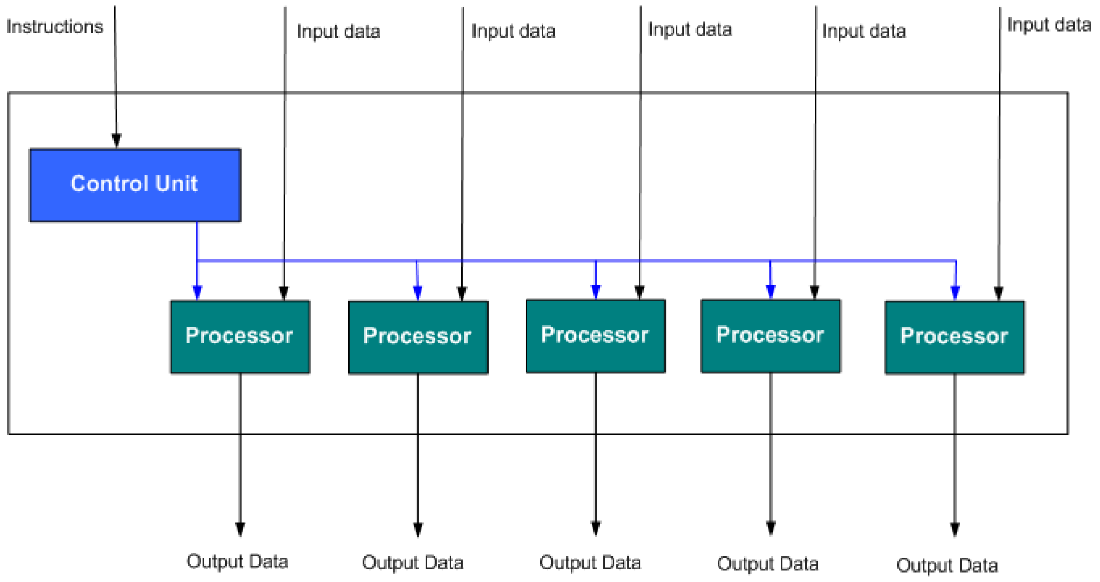
By far the most common way to characterize these architectures is Flynn's taxonomy. He categorizes all computers according to the number of instruction streams and data streams they have, where a stream is a sequence of instructions or data on which a computer operates. In Flynn's taxonomy, there are four possibilities: SISD, SIMD, MISD, and MIMD.

Single Instruction, Single Data (SISD). In a SISD system, one stream of instructions processes a single stream of data, click on the arrow to the right to see an example of the SISD approach. This is the common von Neumann model used in virtually all single-processor computers.



Single Instruction, Single Data architecture

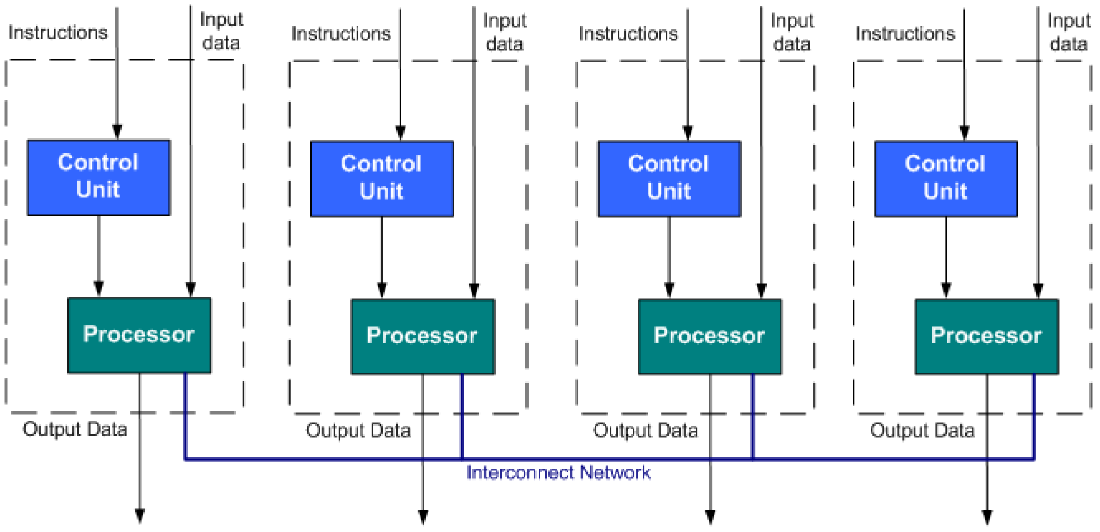
Single Instruction, Multiple Data (SIMD). In a SIMD system, a single instruction stream is concurrently broadcast to multiple processors, each with its own data stream. Click on the arrow at the end of this paragraph to see and example. The original systems from Thinking Machines and MasPar can be classified as SIMD. The CPP DAP Gamma II and Quadrics Apemille are more recent examples; these are typically deployed in specialized applications, such as digital signal processing, that are suited to fine-grained parallelism and require little interprocess communication. Vector processors, which operate on vector data in a pipelined fashion, can also categorized as SIMD. Exploiting this parallelism is usually done by the computer.



Single Instruction, Multiple Data architecture

Multiple Instruction, Single Data (MISD). No well-known systems fit this designation. It is mentioned for the sake of completeness.

Multiple Instruction, Multiple Data (MIMD). In a MIMD system, each processing element has its own stream of instructions operating on its own data. This architecture is the most general of the architectures in that each of the other cases can be mapped onto the MIMD architecture. The vast majority of modern parallel systems fit into this category. Click on the arrow to see and example.

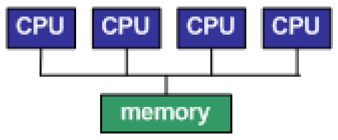


Multiple Instruction, Multiple Data architecture

The HD-UAV ASIC falls into two categories SIMD and MIMD. The chips is specifically designed for the Aerospace Industry to handle highspeed image compression and encryption but also to handle autonomy control over the UAV Drone itself. The purpose of this chip is to be able to handle the media requirements of a High Resolution Camera and while compressing and encryption are happening simultaneously, it also capable of making intelligent decisions about its flight path and possible collision detection based on video data.

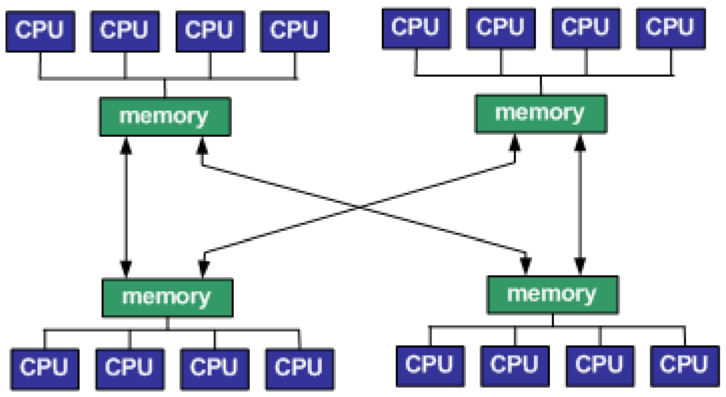
The MIMD category of Flynn's taxonomy is too broad to be useful on its own; this category is typically decomposed according to memory organization.

In a shared-memory system is called SMPs (symmetric multiprocessors). Click on the arrow to the right to see and example. All processors share a connection to a common memory and access all memory locations at equal speeds. SMP systems are arguably the easiest parallel systems to program because programmers do not need to distribute data structures among processors. Because increasing the number of processors increases contention for the memory, the processor/memory bandwidth is typically a limiting factor. Thus, SMP systems do not scale well and are limited to small number of processors.



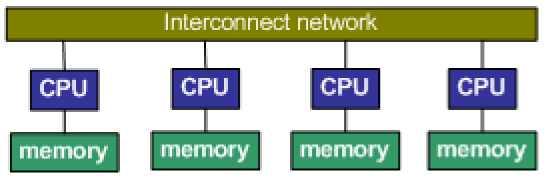
The Symmetric Multiprocessor (SMP) architecture

The other main class of shared-memory systems is called NUMA (nonuniform memory access). The memory is shared and is uniformly addressable from all processors, but some blocks of memory may be physically more closely associated with some processors than others. This reduces the memory bandwidth bottleneck and allows systems with more processors. As a result, the access time from a processor to a memory location can be significantly different depending on how "close" the memory location is to the processor. To mitigate the effects of nonuniform access, each processor has a cache, along with a protocol to keep cache entries coherent. Hence, another name for these architectures is cache-coherent nonuniform memory access systems (ccNUMA). Logically, programming a ccNUMA system is the same as programming an SMP, but to obtain the best performance, the programmer will need to be more careful about locality issues and cache effects.



The nonuniform memory access (NUMA) architecture

In a distributed-memory system, each process has its own address space and communicates with other processes by message passing (sending and receiving messages).



The distributed-memory architecture

Depending on the topology and the technology used for the processor interconnection, communication speed can range from almost as fast as shared memory (in tightly integrated supercomputers) two orders of magnitude slower (for example, in a cluster of PCs interconnected with an Ethernet network). The programmer must explicitly program all the communication between processors and be concerned with the distribution of data.

Distributed-memory computers are traditionally divided into two classes: MPP (massively parallel processors) and clusters. In an MPP, the processors and the network infrastructures are tightly coupled and specialized for use in a parallel computer. These systems are extremely scalable, in some cases supporting the use of many thousands of processors in a single system.

Clusters are distributed-memory systems composed of off-the-shelf computers connected by an off-the-shelf network. When the computers are PCs running Linux OS, these clusters are called Beowulf clusters. As off-the-shelf networking technology improves, systems of this type are becoming more common and much more powerful. Clusters provide an inexpensive way for an organization to obtain parallel computing capabilities.

We have described the various parallel computing and processing that is used on systems. The HD-UAV ASIC is a chip that has a lot of what we have been describing built into a single chip, but without all the draw backs to each system. We have optimized our chip using our own proprietary instruction set and utilizing some of the various parts of architectures described on the previous paragraphs.

Combinatorial Engine

In digital circuit theory, combinational logic (also called combinatorial logic) is a type of logic circuit whose output is a function of the present input only. This is in contrast to sequential logic, in which the output depends not only on the present input but also on the history of the input. In other words, sequential logic has memory while combinational logic does not.

Combinational logic is used in computer circuits to do boolean algebra on input signals and on stored data. Practical computer circuits normally contain a mixture of combinational and sequential logic. For example, the part of an arithmetic logic unit, or ALU, that does mathematical calculations is constructed in accord with combinational logic, although the ALU is controlled by a sequencer that is constructed in accord with sequential logic.

Our Combinatorial Engine is a unique proprietary design that allows us to optimize the strict use of our device for real-time compression/encryption. This type of design can run 30 – 40 times faster than a typical processor because it does not have the waiting for instructions to tell the microprocessor what to do. It is specifically waiting for our RISC processor to free it to start handling the major functions of our UAV ASIC chip. Our Engine uses both stored memory and input related values to for a Highspeed Digital Design to sample the CMOS Imager and immediately start the compression and encryption of the Video Data.

Each Blitzcore CPU has a Combinatorial Engine in it right next to the RISC core processor. Our estimation of how fast the engine can go is accurate based on our truth tables and the amount of time delays between areas of logic and clocking speeds. The Combinatorial Engine will set us apart from any other company attempting to create this type of chip to handle the UAV market. It will take the industry some time to catch up. We have been working on high-speed digital design and video codec chips for more than 7 years and we are ready to release this technology into several different markets including the very lucrative UAV Market.

Project HD-UAV

Prototypes

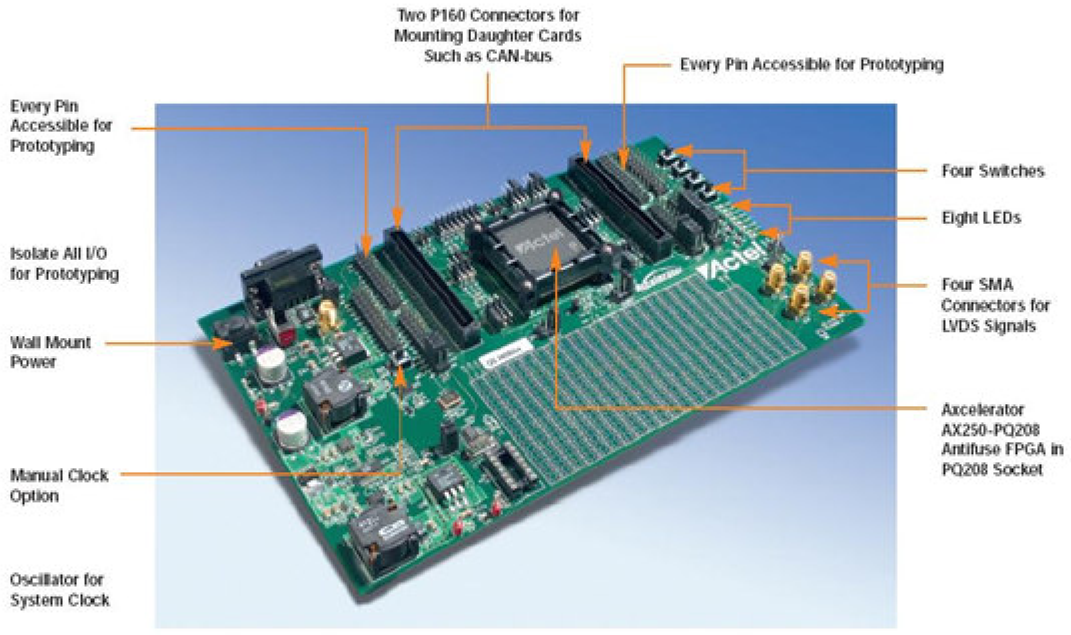
Prototype C

Purpose:

1. Validate simulation results (critical areas)
2. Test combinatorial Engine
3. Test and verify new instructions for Blitz Core (limited by FPGA)
4. Collect Data from FPGA to calculate speed for delays and clocking.
5. Initial CMOS Imager interface testing
6. Initial HDD firmware interface testing
7. Test areas of Distribution processor
8. Test areas of Flash Disk interface
9. Initial Video Driver Testing (Hi-Def)
10. Use all test results for Systems Analysis and refine the Gantt Chart for our development team.

This prototype will be purchased off-the-shelf to get the initial test results complete. The test results are critical to determine the type of design that "**Prototype B**" will have to be to have a fully functional design running on a single board. Below is a picture of the board that we will be using during this testing. We are currently working on our Phase 2 Development team that will have the talent to design and test in these strict areas of development. We will be purchasing about 4 of these boards including the support software for developing the Hardware Description Language. The Gantt Chart and Systems Analysis report will be adjusted according to the test results of Prototype C. The cost of this prototype and all the test tools is roughly $20,000 - $30,000 depending on the tools and licensable technology we use.

Off-the-shelf board for Prototype C


Two P160 Connectors for Mounting Daughter Cards Such as CAN-bus
Every Pin Accessible for Prototyping
Every Pin Accessible for Prototyping
Four Switches
Eight LEDs
Isolate All I/O for Prototyping
Four SMA Connectors for LVDS Signals
Wall Mount Power
Axcelerator AX250-PQ208 Antifuse FPGA in PQ208 Socket
Manual Clock Option
Oscillator for System Clock

Prototype B

Purpose:

1. Validate integration of all areas into one design.
2. Investor viewable design for raising more capital.
3. Verify speed and control of various functions on a single board with 4 FPGA's.
4. Validate and Verify all Library Files
5. Based on test results complete Risk Management report for Phase 3 Chip design.
6. Design Critical Testing Process for Phase 3 Chip Design.
7. Test results of Library and HDL files to build and design HD-UAV ASIC.
8. Test interface ports, (Ethernet, Firewire, USB etc…)

This prototype is a lot more difficult and costly to design. The Phase 2 development team will get injected with some excellent talent to help facilitate the building of this prototype. Our goal is to be completed with this prototype by February 15, 2007. After the testing of Prototype C, we will have a more refined price quote. The current price we are projecting is between $150,000 and $200,000 depending on the features the client and customer need and which the Senior Manager (Ryan) approve for designing into the prototype. This prototype will give us the test results we need for the final prototype which actually uses our HD-UAV ASIC chip for the first time. After the prototype B is complete the Gantt Chart and Systems Analysis will be close to 100% complete, so we can start development on the chip.

Prototype A

Purpose:

1. To test the first chip off the manufacturing line.
2. Test our new Printed Circuit Board techniques for reducing noise and handling high-speed digital design.
3. Validate and Verification process refined and created for testing and approving chips final design.
4. Show prototype to customers and clients to show and verify the design and function of the unit.
5. We will add more to this as we get more into Phase 2 Development teams test results.

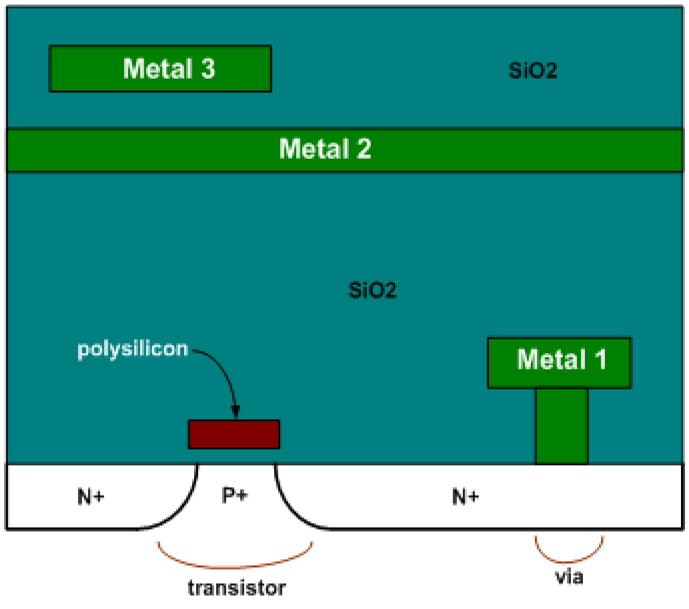
This prototype will be to validate the chip that a manufacturing company has built using our design files. We will have a lot more details about cost but the Gantt Charts and Systems Analysis reports are in their initial stages and not a lot of data can be derived from them in terms of cost.

Manufacturing Process

The HD-UAV ASIC will be outsourced to a leading chip manufacturer. Elite will scrutinize their process making sure that it meets the demands of our chip. We look for speed, heat dissipation, power drain, signal integrity, impurities and many other factors to determine who will ultimately be the company who manufactures our chip.

Integrated circuits are built on a silicon substrate provided by the wafer. Wafer sizes have steadily increase over the years. Larger wafers means more chips per wafer and higher productivity. The key figure of merit for a fabrication process is the size of the smallest transistor it can manfuacture. Transistor size helps determine both circuit speed and the amount of logic that can be put on a single chip. Fabrication technologies are usually identified by their minimum transistor length, so a process which can produce a transistor with a 0.13 um minimum channel length is called a 0.13 um process. (Below 0.10 um we switch to nanometer units, such as a 90 nm process.)

A pure silicon substrate contains equal numbers of two types of electric carriers: electrons and holes. The interplay between electrons and holes is what makes transistors work. The goal of doping is to create two types of regions in the substrate: an n-type region which contains primarily electrons and a p-type region which is dominated by holes. (Heavily doped regions are referred to as n+ and p+.) Transistors action occurs at properly formed boundaries between n-types and p-type regions.



Cross-section of an integrated circuit

Components are formed by a combination of processes:

- doping the substrate with impurities to create areas such as the n+ and p+ regions;
- adding or cutting away insulating glass (silicon dioxide, or SiO2) on top of the substrate;

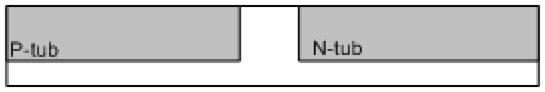
- adding wires made of polycrystalline silicon (polysilicon, also known as poly) or metal, insulated from the substrate by SiO2.

The n-type and p-type regions and the polysilicon can be used to make wires as well as transistors, but metal (either copper or aluminum) is the primary material for wiring together transistors because of its superior electrical properties. There may be several levels of metal wiring to ensure that enough wires can be made to create all the necessary connections. Glass insulations lets the wires be fabricated on top of the substrate using processes like those used to form transistors. The integration of wires with components, which eliminates the need to manually wire together components on the substrate, was one of the key inventions that made the intergated circuit feasible.

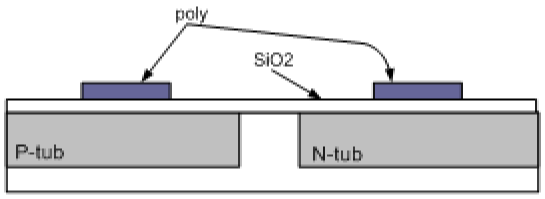
Features are patterned on the wafer by the photolithographic process; the wafer is covered with light-sensistive material called photoresist, which is then exposed to light with the proper pattern. The pattern left by the photoresist after development can be used to control where SiO2 is grown or where materials are placed on the surface of the wafer.

A layout contains summary information about the patterns to be made on the wafer. Photolithographic processing steps are performed using masks which are created from the layout information supplied by the layout, though in more complex processes some masks may be built from several layers while one layer in the layout may contribute to several masks.

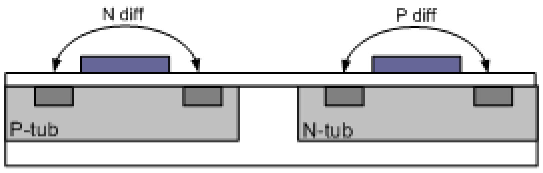
Transistors may be fabricated on the substrate by doping the substrate; transistors may also be fabricated within regions called tubs or wells. An n-type transistor is built in a p-doped, and a p-type transistor is built in an n-doped region. The wells prevent undesired conduction from the drain to the substrate. (Remember that the transistor type refers to the minority carrier which forms the inversion layer, so an n-type transistor pulles electrons out of a p-tub). The twin-tub process, which starts from an undoped wafer and creates both types of tubs, has become the most commonly used CMOS process because it produces tubs with better electrical characteristics.



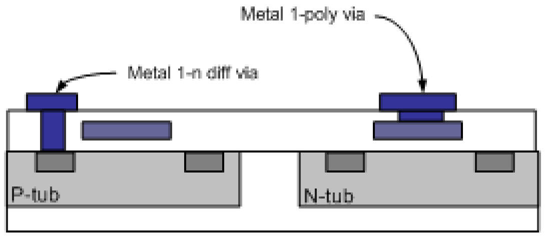
Forming tubs



Adding oxide and polysilicon



Diffusing source/drain regions



Adding metal

The twin-tub process

Details can vary from process to process, but these steps are representative. The first step is to put tubes into the wafer at the appropriate places for the n-type and p-type wafers. Regions on the wafer are selectively doped by the implanting ionized dopant atoms into the material, then heating the wafer to heal damage caused by ion implantation and further move the dopants by diffusion. The tub structure means that n-type and p-type wire cannot directly connect. Since the two diffusion wire types must exist in different type tubs, there is no way to build a via which can directly connect them. Connections must be made by a separate wire, usually meta, which runs over the tubs.

The next steps form an oxide covering of the wafer and the polysilicon wires. The oxide is formed in two steps: first, a thick field oxide is grown over the entire wafer. The field oxide is etched away in areas directly over transistors; a separate step grows a much thinner oxide which will form the insulator of the transistor gates. After the field and thin oxides have been grown, the polysilicon wires are formed by depositing polysilicon crystalline directly on the oxide.

Note that the polysilicon wires have been laid down before the diffusion wires were made-that order is critical to the success of MOS processing. Diffusion wires are laid down immediately after polysilicon deposition to create self-aligned transistors-the polysilicon masks the formation of diffiusion wires in the transistor channel. For the transistor to work properly, there must be no gap between the ends of the source and drain diffusion regions and the start of the transistor gate. If the diffusion were laid down first with a hole left for the polysilicion to cover, it would be very difficult to hit the gap with a polysilicon wire unless the transistor were made very large. Self-aligned processing allows much smaller transistors to be built.

After the diffusions are complete, another layer of oxide is deposited to insulate the polysilicon and metal wires. Aluminum has long been the dominate interconnect material, but copper has now moved into mass production. Copper is a much better conductor than aluminum, but even trace amounts of it will destroy the properties of semiconductors. Chips with copper interconnect include a special proection layer of copper. That layer prevents the copper from entering the substrate during processing.

Multiple layers of metal interconnect are separate by silicon dioxide. Each layer of SiO2 must be very smooth to allow the next layer of metal to be deposited without breaks. The deposition process may be somewhat uneven; in addition the existing layers of metal form the hills and valleys underneath the silicon dioxide. After an insulating layer is deposited, it is polished to a smooth surface using processes similar to those used to grind optical glass. This ensures that the next layer of interconnect will not have to form itself over an uneven surfcae that may cause breaks in the metal.

Holes are cut in the field oxide where vias to the susbtrate are desired. The metal 1 layer is then deposited where desired. The metal fills the cuts to make the connections between layers. The metal 2 layer requires an additional oxidation/cut/deposition sequence. Another layer of silicon dioxide is deposited and then polished to form the bae for the next layer of interconnect. Most modern processes offer at least four layers of metal.

After all the important circuit features have been formed, the chip is covered with a final passivation layer of SiO2 to protect the chip from chemical contamination.

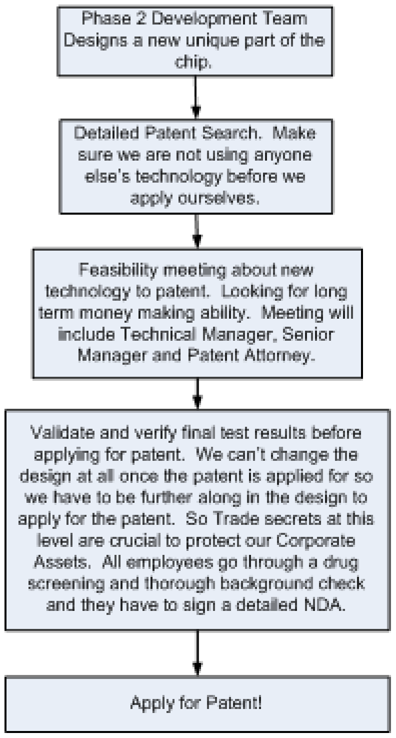
Modern VLSI fabrication processes in fact take hundreds of steps and dozens of masks. Generating very deep submicron structures with adequate electrical propertis is much more challenging than was the task of building early MOS Ics whose transistors were multiple microns across. The larger number of fabrication steps causes manufacturing times for 90 nm process to take up to six to eight months; those lead times are likely to increase to grow as the line widths shrinks.

Patents

HD-UAV Patents

1. HD-UAV ASIC – The entire chip
2. CMOS Imager – Proprietary design for our chip that speeds our imaging processing by 16 times
3. HD-UAV Blitzcore Processor – Proprietary RISC core
4. HD-UAV Combinatorial Engine – Proprietary logic design to speed up compression and encryption
5. HD-UAV Distribution Processor – One of the most crucial processors in the ASIC chip

There will be a lot more patents to apply for once the projects gets further into testing. Below is our process for applying for patents.


Phase 2 Development Team Designs a new unique part of the chip.
Detailed Patent Search. Make sure we are not using anyone else's technology before we apply ourselves.
Feasibility meeting about new technology to patent. Looking for long term money making ability. Meeting will include Technical Manager, Senior Manager and Patent Attorney.
Validate and verify final test results before applying for patent. We can't change the design at all once the patent is applied for so we have to be further along in the design to apply for the patent. So Trade secrets at this level are crucial to protect our Corporate Assets. All employees go through a drug screening and thorough background check and they have to sign a detailed NDA.
Apply for Patent!

Every processor in the chip has a proprietary instruction set that we designed. The firmware, which will be copyrighted, is the unique program that will run each processor. There will be an embedded operating system licensable to anyone in the aerospace industry wanting to use this chip for other applications besides the UAV.