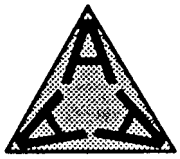
An Overview of the Advanced Amiga Architecture and Other Future Directions



Document Rev 1.0

1993 Developer's Conference Release

by Dave Haynie
Advanced Amiga System Group

Chapter 1
Introduction

Since the introduction of the original Amiga 1000 in 1985, there have been two major revisions of the Amiga chip set: ECS and AA. The ECS chip set introduced a minimally upgraded version of Agnus and Denise, while the AA chip set introduced more significant enhancements, including Lisa, a brand new Denise replacement. Despite the advantages wrought by AA, the AA chip set was still very much an evolution of the original Amiga chip set rather than anything revol *utionary. AA* maintained most of the original features of Agnus and the entire Paula chip, changing only those things related to video display.

The AAA chip set is the first Amiga chip set to break from this original Amiga architecture. It is composed of four completely new full custom VLSI integrated circuits. It improves every aspect of the Amiga chip set's performance, and its new architecture makes possible many things that could never be directly adapted to the original architecture in any practical sense.

1.1 Targets

This paper is intended to be an overview of the AAA chip set and the direction we're going in with respect to Systems architecture that will surround the AAA chips. This is not intended as a complete definition of either of these, we have hundreds of pages of internal documentation devoted to AAA and next generation systems architecture that does that job.

A good understanding of the original, ECS, and AA chip sets, while probably not vital, will be very helpful in understanding this document and, more importantly, the goals of AAA itself. While there have been many, many improvements in AAA over previous Amiga chip sets, AAA is still very much an Amiga chip set. It is an updated version of many of the same design philosophies that led to prior Amiga implementations.

1.2 Credits

Most of the material on the AAA chip set was gleaned, adapted, and sometimes outright copied from the internal document "Advanced AMIGA Architecture". Many thanks go out to Jim Redfield, Ed Hepler, and the test of the AAA design group for writing most of this paper for me.

Chapter 2
Goals of the AAA Project

The main point of the AAA chip set is to move the Amiga back toward the leading edge of personal computer technology. However, in doing so, one main requirement is basic compatibility with the existing Amiga chip sets, to keep as much software running as possible without compromising the other AAA goals.

2.1 Compatibility

AAA is designed to be largely register compatible with the ECS chip set. Most of the RGA registers from ECS are supported. The ECS "Ultra hires" registers have been eliminated, as they were never supported in actual practice. Some other display-generation details of ECS are no longer required or supported in AAA.

The AA registers are not supported in AAA. We believe that the 3.0 OS provides the necessary control over AA and that no one need program "to the metal" on AA systems. Additionally, some of the AA features were implemented in a less-than-ideal fashion, in order to fit in the same RGA address space originally implemented in ECS. All AA-equivalent function can be done much better by new AAA support than some kind of AA emulation. Some behaviors can't be perfectly emulated in AAA. Clearly, the AAA chip set's architecture will have an immediate impact on some elements of the ECS emulation apart from register-level compatibility. For example, on a VRAM system, there's no way to slow the system down to an equivalent cycle-stealing ECS mode. So a program will often find significantly more blitter, copper, and CPU access than on an ECS machine. This won't be a problem if you're using the OS correctly, but it could be a problem to take-over-the-system type programs. Such programs may also have some degree of problem with some AAA screen promotions and copper activity.

We envision AAA as a transitional system. It is highly desirable to eliminate the ECS-compatibility, or for that matter, reliance on any register-level compatibility, for the next generation of Amiga chips. So there's an excellent chance that you're seeing some of this stuff for the last time in AAA.

2.2 Flexibility

The pre-ECS, ECS, and to a lesser degree the AA chip sets were each designed with a single system architecture in mind. This defined a good deal of how the chips work together, what kind and how much memory they would support, which CPU interface they'd hook into, etc. Any deviation from this basis could result in a pricey system full of work-around logic, as we had to some extent in the A3000. They also forced this defined Amiga chip sub-system to be the same in all computers, from the low-end to the high-end, so we might have too much expense for an ideal low end, not enough power for an ideal high-end.

The AAA chips are designed to work in several configurations. They can use cheap DRAM, but will gain significant extra performance using VRAM (DRAM with a high speed serial port). They can easily hook up to a variety of 32-bit CPU buses via an easy-to-implement asynchronous slave interface. CPU access to Chip RAM can be gated though similarly to ECS or AA, or the CPU bus can master the entire chip bus (providing all RAM timing) for more efficient CPU to Chip bus access. Finally, the AAA chips can be assembled in "Single" or "Dual" configurations, depending on the display goals.

23 **Improvements**

Every aspect of the previous Amiga chip sets has been improved upon in AAA. While not every feature can be discussed here at length, some of the improvements follow.

23.1 **Memory Bandwidth**

The AAA chip bus is an improvement over the ECS and even AA chip bus in terms of memory speed. The AAA chips run a four cycle burst to Chip RAM, which in raw performance is 4.56 times faster than ECS memory access or 1.14 time faster than AA's two-cycle burst. However, the real key to AAA's memory architecture is its support for VRAM. With VRAM, display fetches have practically no effect on the normal parallel chip RAM bus, freeing it for use by Blitter, Copper, and CPU.

Page-Mode performance is actually a bit better than this implies, since under the right circumstances the AAA chips can run extended burst cycles. Bursts of up to 512 words can be run to keep up with high resolution displays. This improves overall system performance, but increases latency to chip RAM.

2.3.2 **CPU Bandwidth**

The CPU access to Chip RAM is improved over past systems. Since the AAA chips manage an asynchronous interface to the CPU, CPUs running at any speed take less of a synchronization penalty for chip RAM access than they do in the current A3000 and A4000 systems, where synch-up is managed externally by the Gary chip. Also, AAA's dynamic chip bus slot allocation allows the CPU to get in more often, it's not limited to one out of every two slots. Finally, as mentoned, an external device such as the CPU with some extra support logic can completely master the AAA chip RAM bus, allowing Chip RAM access as fast as today's 32-bit Fast RAM,

2.3.3 Chip Bus DMA

Unlike previous Amiga systems, the AAA chip bus DMA activity is dynamically managed. The different DMA channels (40 of them, including the standard and high-priority external channels for CPU) have fixed priorities with respect to one another, though the relation between the blitter and the CPU can be adjusted in various ways. Because of this dynamic allocation, its possible to run out of cycles before everything has all the time it wants. The

highest priority channels are the deterministic channels, the lowest the blitter or CPU. They are: graphics overlay, DRAM refresh, interrupt transfer, disk, high priority external access, audio, display, sprite, coprocessor, processor, and blitter (the latter two by default and changable, as mentioned).

When too many cycles are requested, something requiring deterministic acccess has to starve. That something will be the graphics fetch. When the system can't fetch enough graphics data for a line, it sends the graphics overrun interrupt to the CPU. This is to be taken as an indicator to software than something needs to be quieted down.

23.4 Blitter

The AAA blitter is significantly faster than the ECS/AA blitter when running in 32-bit mode, thanks to the faster, wider bus. Just doing basic scrolls, it can scroll a 640x200x2 screen about 6 times faster, or a 640x200x4 screen about 9 times faster than with the old blitter. But that's just raw data movement.

Logical improvments to the blitter streamline much of its use. In 32-bit mode, it's much easier to program. It now operates using pixel addressing rather than via masks, modulos, and shifts. It can operate on traditional Amiga bitplanes, or on chunky pixels of 2,4,8, or 16 bits width. The line-draw has also been improved, supporting a new "clip-rect" mode for better GUI performance under Intuition. Finally, several arithmetic operations have been added for "sort" and "tally" operations on planes of any pixel depth.

23.5 Copper

The copper has been improved in several areas. It can handle 32-bit operations for the new 32-bit registers, and supports a "move-multiple" function for more efficient loading of blocks of consecutive registers, such as color tables. The copper now has an interrupt capability, which lets it receive an interrupt from the blitter. This allows the copper to manage a series of blit operations, one after another, without additional processor intervention.

2.3.6 Graphics

Extensive improvements have been made to the graphics in AAA. A "single" *system with* Fast-Page DRAM can support displays up to 800x560x9 bitplanes. The same system using VRAM can support 800x560x13 bitplanes or 800x560x24 using "hybrid" pixels. A "dual" system can support up to 1280x1024x5 displays with Fast-Page DRAM, up to 1280x1024x8 bitplanes or 1024x768x24 "hybrid" using VRAM.

The AAA chip set supports 256 CLUT entries of 25 bits each, like AA does. It can handle sprites up to 128 bits wide. It supports up to 16 bitplanes, which makes dual 8-bit playfields possible. The AAA pixel clock is no longer ties to the AAA bus clock, so a variety of display resolutions, even standard ones, can be generated by an AAA system. Hardware-assisted screen promotion is also supported via scaled pixel clocks.

There are a large variety of pixel types supported in AAA. Along with the traditional bitplane-generated pixels (including HAM8 and HAM10), we have several kinds of chunky and compressed pixels.

Half-Chunky	Half-chunky pixels come in 2,4, or 8-bit depths. These indirect through the color lookup table like most planar modes do.
Chunky	Chunky pixels are 16-bits deep. They bypass the CLUT, providing 5 bits of red, S-bits of green, 5-bits of blue, and one genlock overlay directly.
Hybrid	Hybrid pixels are 24-bits deep, composed of separate chunky planes for Red, Green, and Blue. The genlock overlay is fixed at (R,G,B) = 0 for this mode.
PACKLUT	These compressed pixels are stored at 2 bits per pixel and decompress to 8-bit half-chunky pixels. This is done by dividing the screen into 4 x 4 pixel regions. Each region contains two colors (8-bit values indexed through the CLUT) and sixteen pixels.
PACKHY	These compressed pixels are stored at 4 bits per pixel and decompress to 24-bit direct pixels. This is done by dividing the screen into 4 x 4 pixel regions. Each region contains two colors (24-bit direct values) and sixteen pixels.

2.3.7 **Video Capture**

The AAA pixel bus direction can be reversed, allowing an optional low cost video capture device (framegrabber) to be implemented in AAA systems. Capture can be in any chunky display mode. This only works in systems that use VRAM.

2.3.8 Sound

The AAA chip set contains the first improvement in Amiga-based sound since the Amiga was introduced. The audio circuitry can handle sampling rates of better than 50kHz with 16-bit resolution. Eight channels are supported, and channels can be assigned to the left or right output. The 16-bit D/A converters are on-chip, and an external converter is also supported. Additionally, the chip set does 8-bit audio sampling.

2.3.9 **Floppy Disk**

In addition to supporting the original 1 megabyte disk used in previous Amiga systems, the AAA chip set supports 2 and 4 megabyte disk formats as well. This is, of course, direct support, no speed controls or other kludges are necessary.

As well as supporting the increased floppy densities, the AAA chipset has considerably more flexibility. It has built-in decoding hardware, which can decode MFM, RLL(2,7), and Biphase Mark (CD-ROM) formats. It can transfer by track, sector, a special "CD mode", and a special high speed *trackmode*. Its data I/O rate has increased from 0.5 Mbit/sec to approximately 11.4 Mbit/sec (though only encoded data can handle this, since the DMA rate available to the floppy logic peaks at 9.9Mbits/sec). Finally, the data separator is programmable, which is very useful at tweaking up to the optimal performance with any specific medium.

This flexible controller can handle practically any floppy format yet invented. With the proper software, it should have no problem with the original Mac format. It handles IBM formats at 360K, 720K, 1.2MB, 1.44MB. and 2.88MB, directly with sectoring (no track buffer necessary). In theory, an RLL(2,7) floppy format at 4 megabyte density could store somewhere between 4.0MB and 5.2MB on a single disk. It should also be able to support 21.6MB flopticals.

And it can support devices other than floppies, too. CD-ROM would be a direct connect, and similarly formatted DAT and digital radio should work too. It might even be possible to support an ST-506 hard drive (assuming such puppies still exist). Higher transfer rates need a cleaner signal, and may require external clocking of the data separator PLL, which is supported by the chip set.

23.10 UARTS

The AAA chip set contains two UARTs. Both are improvements over the Paula UART, each buffered with a four-byte FIFO. This significantly improves serial performance at high speeds, since fewer interrupts are taken, and increases reliability, since there's much more time available to respond to a serial interrupt.

Chapter 3
The AAA Chips

The AAA system consists of four completely new VLSI chips, implemented in high speed CMOS. The functions of the first three are partitioned similarly to those of the ECS/AA chip sets. Andrea is the chip bus controller, analogous to Agnus/Alice. Monica is the new display controller, replacing Denise. Mary, the Paula replacement, controls various types of I/O. Finally, a new chip, Linda, double-buffers full display lines.

For the most part, the four chips function as one. Between them, there are nearly 256 word-addressed registers (compatibility registers), 384 longword-addressed registers (new stuff), and 5 12 longword-addressed CLUT registers. It's often true that a single register address function is performed by two or three of the chips.

3.1 **The Andrea Chip**

Andrea is the chip bus controller, responsible for managing the chip bus. It is the core of the AAA system, much like a microprocessor is the core of a normal CPU system. The Andrea chip has a rather impressive list of features:

Chip RAM control. Normally, Andrea manages all chip RAM access, supporting both Fast-Page DRAM and VRAM. A simple configuration mechanism tells Andrea what's on the chip bus at startup time. Eight 256Kword banks of chip RAM are supported, and with a little extra logic banks of Fast-Page DRAM and VRAM can be intermixed on a bank-by-bank basis. A "high priority" bus request allows an external device to master the chip RAM bus rather than Andrea.

CPU bus gating. The Andrea chip controls access to the chip RAM bus and chip registers from the CPU port. CPU addresses go through Andrea, CPU data is externally latched under Andrea's control. This is a reasonably general purpose CPU interface, which manages synchronization to the chip bus, directly supporting CPUs of varying clock speeds (previously this was done with extra logic wrapped around the ECS and AA chip sets, like the A3000 and A4000).

Chip bus control. A variety of control signals for management of both the chip bus and various aspects of the other AAA chips are generated in Andrea. The multiplexed chip address/data bus is also mastered by Andrea. This bus contains a register address at the start of a cycle (like the RGA bus on ECS/AA), data in the later stages of a cycle. All of the AAA chip set's DMA channels are addressed by Andrea, which is also responsible for allocating the various DMA channels according to which units are requesting DMA and the priority of each requesting channel.

Clocks. The Andrea chip manages clocking of both the chip bus and the video display. Control lines from Andrea select one of eight possible pixel clock values available at ***any given time. These can be*** changed on a line-by-line basis.

Video timing control. AlI video timing and synchronization signals are created in Andrea. A number of different counters in Andrea generate video synchs, blanking, and the logical controls necessary to support **the AAA** display. Andrea also manages a light pen intput.

Blitter. The Andrea chip contains the Amiga blitter. This blitter can handle the AAA chip RAM bus at full speed and width, but it also has a compatibility mode that lets it handle 16bit data just like the pre-AAA blitter. In 32-bit bit mode, the blitter has a considerable number of new functions. The new blitter is pixel addressed, it automatically calculates first and last mask, proper shifts, and whether prereads are necessary. It supports pixel sizes of 1, 2,4, 8, or 16 bits, to cover all AAA display modes. It now has "sort" and "tally" functions designed mainly to assist chunky pixel processing. The sort function will run a bubble sort on a plane of pixels, governed by a sort key. The tally function records the number of instances of each byte value in a plane of pixel data. Finally, the new blitter can perform a variety of arithmetic operations ***on*** chunky pixels, including addition, averaging, subtraction, saturated subtraction, etc. The following tables illustrate example blitter speed (screens scrolled/second) for various system configuration:

Single, ***Fast-Puge DRAM***

Display	**640x200**	**640x400**	**800x560**	**1024x768**	**1280x1024**
2 Bitplane	489.06	233.42	124.54	NA	NA
4 Bitplane	233.37	105.58	51.49	NA	NA
8 Bitplane	105.53	41.65	14.96	NA	NA
16 Bitplane	41.61	NA	NA	NA	NA
Half Chunky	110.22	46.33	19.03	NA	NA
Chunky	46.52	14.59	NA	NA	NA
Hybrid	24.99	NA	NA	NA	NA
ECS 2 Bitplane	81.80	25.83	NA	NA	NA
ECS 4 Bitplane	25.87	NA	NA	NA	NA

Single, ***Video DRAM***

Display	**640x200**	**640x400**	**800x560**	**1024x768**	**1280x1024**
2 Bitplane	**504.23**	**233.42**	**124.54**	**NA**	**NA**
4 Bitplane	248.54	105.58	51.49	NA	NA
8 Bitplane	120.70	41.65	14.96	NA	NA
16 Bitplane	56.78	NA	NA	NA	NA
Half Chunky	127.39	46.33	19.03	NA	NA
Chunky	63.70	14.59	NA	NA	NA
Hybrid	42.17	NA	NA	NA	NA
ECS 2 Bit-plane	81.80	25.93	NA	NA	NA
ECS 4 Bitplane	25.87	NA	NA	NA	NA

Dual. Fast-Page DRAM

Display	640X200	640x400	800X560	1024 x 768	1280x1024
2 Bitplane	498.24	242.59	134.05	71.66	38.56
4 Bitplane	242.56	114.75	61.00	30.05	13.59
8 Bitplane	114.72	50.83	24.48	9.24	NA
16 Bitplane	50.80	18.87	NA	NA	NA
Half Chunky	118.77	54.86	27.72	11.93	3.44
Chunky	55.06	23.12	9.63	3.31	NA
Hybrid	33.54	NA	NA	NA	NA
ECS 2 Bitplane	81.80	25.83	NA	NA	NA
ECS 4 Bitplane	25.87	NA	NA	NA	NA

Dual, Video RAM

Display	640 x 200	640 x 400	800X560	1024 x 768	1280 x 1024
2 Bitplane	504.36	248.69	140.49	78.67	46.38
4 Bitplane	248.68	120.85	67.44	37.06	21.41
8 Bitplane	120.84	56.93	30.9 1	16.25	8.92
16 Bitplane	56.92	24.97	12.65	5.85	2.68
Half Chunky	126.85	62.93	35.71	20.12	11.92
Chunky	63.36	31.40	17.79	9.99	5.89
Hybrid	41.62	20.32	11.36	6.25	NA
ECS 2 Bitplane	81.80	25.83	NA	NA	NA
ECS 4 Bitplane	25.87	NA	NA	NA	NA

Copper. The AAA copper, as mentioned, supports both 16-bit and 32-bit register operations. A multiple move instruction has been implemented to greatly reduce the overhead of large sequential register movements. A new interrupt mechanism allows the copper to be interrupted. Interrupt sources, in order of priority, are vertical blanking, wait finished, and blitter finished. The blitter finished interrupt allows the copper to re-load the blitter with the next blit operation at the end of the current blitter operation. In this way, the CPU can usually just schedule blit operations in the copper's blitter interrupt routine and get on with other work.

3.2 **The Linda Chip**

The Linda chip is a display line buffer for the AAA system. Linda provides a great deal of the intelligence behind the AAA display system. While one complete line of display data is being fed to Monica from one of Linda's line buffers, the next line is being fetched into Linda's other line buffer. There are many advantages of this prefetch process:

Much more efficient Fast-Page display fetch cycles can be run than previously possible. Previous systems use zero or two-cycle Fast-Page fetches, AAA systems can run "burst" cycles hundreds of transfers in length.

Bitplane alignment can still be on word boundries (as allowed with ECS), Linda makes any necessary realignments in order to pass longword-aligned data on to Monica. There are, however, various alignment restrictions on some of the new display modes:

Mode	Alignment
Bitplane	16-bit
Half-Chunky	64-bit
Chunky	64-bit
PACKLUT	64-bit
PACKHY	64-bit
Overlay	128-bit
32-bit Sprites	16-bit
64-bit Sprites	64-bit
128-bit Sprites	128-bit

Rather than Fast-Page memory, Video RAM (VRAM) can be used without requiring the strict alignment requirements typical of most VRAM-based display systems. VRAM allows display fetch overhead to be essentially eliminated from normal chip bus activity.

The pixel clock is easily decoupled from the bus clock. The chip RAM side of Linda runs at chip RAM bus speed, the Monica side of Linda runs at pixel clock speed.

Packed pixel formats PACKLUT and PACKHY are decoded here, passing them on to Monica as normal 8-bit half-chunky or 24-bit hybrid pixels, respectively.

3.3 **The Monica Chip**

Monica is the AAA display controller chip. It takes in display timing data generated by Andrea and graphics data fetched by Linda and from that generates 25-bit digital and analog RBGoutput (24-bits of color, one of genlock overlay). Monica contains the 256 entry CLUT (color lookup table), HAM mode logic, priority control for playfields/overlay and sprite display, and 8-bit digital to analog converters, one each for Red, Green, and Blue.

HAM (hold and modify) mode is of course the special compact high-color display mode provided in ECS and AA chip sets. Using this mode, Monica can either supply a direct CLUT value or modify a previously displayed value. The five and six-bit modes **from ECS, with** 4096 color resolution, is supported, as well as the eight-bit mode from ECS and a new ten-bit HAM, which allows a full 24-bits worth of color resolution using only ten bitplanes.

An optional one-bit overlay plane is also supported in Monica for chunky or packed display modes. This requires a VRAM-based chip RAM buffer and the chunky or packed display must be in the odd playfield, while the overlay is fetched based on the even playfield pointer. There are a number of restrictions on the overlay plane when compared to **a** normal second playfield in the traditional bitplane display.

The actual display capability of *any* given system is ultimately determined by how fast Monica can be fed pixel data by Linda. Which is, of course, determined by how fast Linda can fetch data from the chip RAM bus. This depends on the type of memory, the size of the system (single or dual), and the display mode (chunky modes tend to allow a more efficient fetch from chip RAM for the same resolution). The following tables show non-interlaced resolutions with different system setups.

Single, Fast Page DRAM

Display Resolution	Bitplane	Half Chunky	Chunky	Hybrid	Pack LUT	Pack Hy
640x200	16	Yes	Yes	yes	Yes	Yes
704x200	16	Yes	Yes	Yes	Yes	Yes
640x400	12	Yes	Yes	no	Yes	Yes
800X 560	10	Yes	no	no	Yes	no
1024x 768	NA	NA	NA	NA	NA	NA
1280x 1024	NA	NA	NA	NA	NA	NA

Single, Video *DRAM*

Display Resolution	Bitplane	Half Chunky	Chunky	Hybrid	Pack LUT	Pack HY
640x200	16	Yes	Yes	Yes	Yes	Yes
704x 200	16	Yes	Yes	yes	Yes	Yes
640X400	16	Yes	Yes	Yes	Yes	Yes
800x 560	14	Yes	Yes	yes	Yes	Yes
1024x 768	NA	NA	NA	NA	NA	NA
1280x 1024	NA	NA	NA	NA	NA	NA

Dual, Fast-Page DRAM

Display Resolution	Bitplane	Half Chunky	Chunky	Hybrid	Pack LUT	Pack HY
640x200	16	yes	yes	yes	yes	Yes
704x 200	16	yes	Yes	yes	yes	Yes
640X400	16	yes	Yes	Yes	yes	Yes
800x 560	13	yes	yes	yes	yes	Yes
1024x 768	8	yes	yes	no	yes	Yes
1280x 1024	5	yes	no	no	yes	no

Dual, Video DRAM

Display Resolution	Bitplane	Half Chunky	Chunky		Hybrid	Pack LUT	Pack HY
640x200	16	yes	yes	Yes	yes	yes	
704x 200	16	yes	yes	Yes	yes	yes	
640X400	16	yes	yes	Yes	yes	yes	
800x 560	16	yes	yes	Yes	yes	yes	
1024x 768	11	yes	yes	Yes	yes	yes	
1280x 1024	8	yes	yes	no	yes	yes	

3.4 The Mary Chip

Mary is the **AAA** peripheral controller chip. It manages floppy disk, audio, and serial (UART) I/O. It is the first improvement in these areas for Amigas built into an Amiga chip set, and the improvements are considerable.

The floppy disk system was discussed in some detail in Chapter 2. Basically, the data separator now runs up to twenty times faster, with variable parameters and the option of a separate PLL clock. Mary supports internal decode of several formats, several new transfer options, hardware CRC calculation, and a higher DMA bandwidth to chip RAM. A comparison with Paula &tails these new features:

Function	Paula	Mary
raw bit width	2000,400O ns	88-9000 ns
max raw bit rate	0.5 Mbit/sec	11.4 Mbit/set
encoding methods:		
raw	Yes	Yes
GCR	Yes	Yes
MFM	no (done in software)	Yes
RLL(2,7)	no	Yes
Biphase-Mark	no	Yes
sync	l6-bit	32, 16,8-bit
transfer modes:		
track	Yes	Yes
sector	no	Yes
CD digital	no	Yes
"trackplus"	no	Yes
hardware CRC	no	yes (sector, trackplus)
async PLL clock	no	Yes
input types	pulse	pulse, NRZ

Mary's audio system is also a substantial improvement over Paula's. The sampling rate,

number of channels, sample resolution, volume resolution, sampling accuracy, and flexibility have all been improved. On-chip DACs are now at 16-bit resolution, and a digital audio output is available at 20-bits per channel (the supported samples are 16-bits, but sample volume yields a 17-bit result, and eight channels summed into a single output without scaling yields a 20-bit

Function	Paula	Mary
sample rate	29kHz	64kHz
channels	4	8
volume bits	6	12 (signed)
volume aliasing	Yes	no
sample size	8 bits	8 or 16bits
digital out	no	yes
dynamic range	15 bits	16 bits
channels on left	2 (0,3)	0-8 (any or all)
channels on right	2 (1,2)	O-8 (any or all)
global mono bit	no	yes
ouput time resolution	280 ns	280 ns
period resolution	280 ns	280/64 ns

value). A comparison of audio in Paula versus Mary yields:

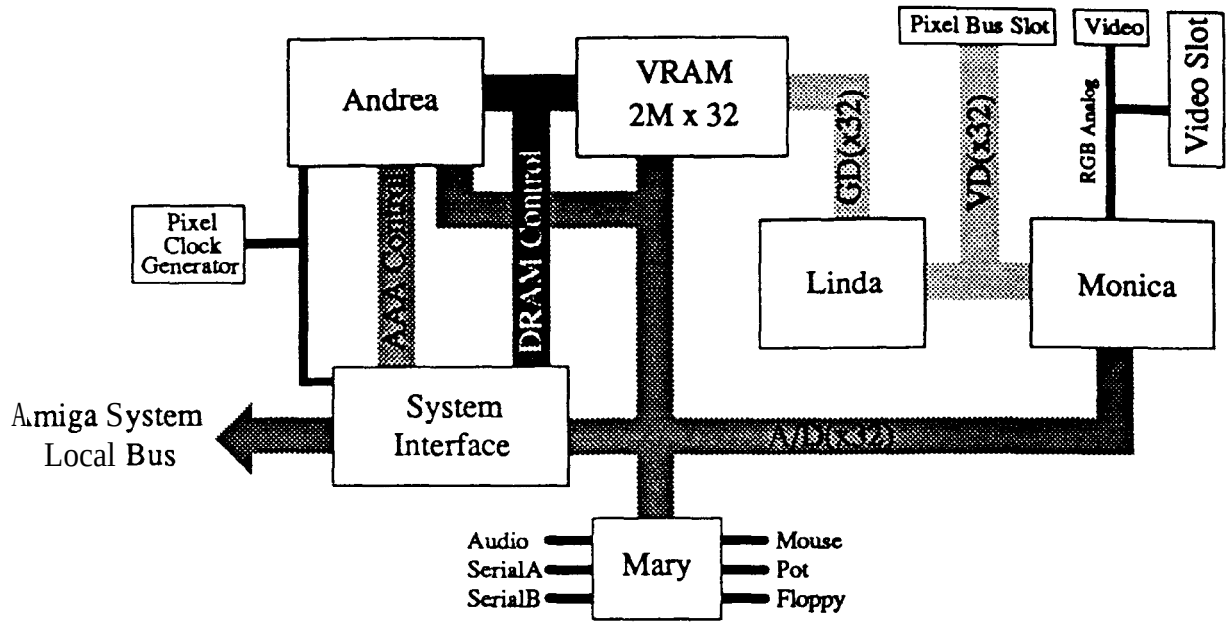
The pot inputs in Mary have been enhanced over Paula's in separate ways. First of all, in order to make them consistent between the widely varying display frequencies supported in AAA, Mary maintains its own sampling counter, which is basically an NTSC compatible horizontal line counter. There is also a new audio sampling mode, which turns the pot lines around to support 8-bit stereo input (with appropriate analog interface circuitry). Higher resolutions are available via an external ADC.

As mentioned before, the Mary UART is an enhancement over the Paula UART. A four deep FIFO has been added, which will reduce interrupt overhead (and overruns caused by missed interrupts) and make faster serial reads possible. Mary contains two UARTS, one that's in the same RGA address as the original Paula UART.

35 The Single System

The "single'* AAA subsystem consists of one each of the four AAA chips. This example system uses video RAM, which is certainly the preferred implementation. Andrea controls this VRAM and the chip bus. The 32-bit parallel port of the VRAM connects to Andrea and the A/D bus (register address and all data go across this bus). The serial ports of the VRAM connect to Linda, which handles all the video fetching, under Andrea's direction. Mary connects to the A/D bus and to various I/O ports, such as audio, floppy, and serial. Finally, Monica connects to the other side of Linda, the A/D bus, and to the video output (digital or analog).

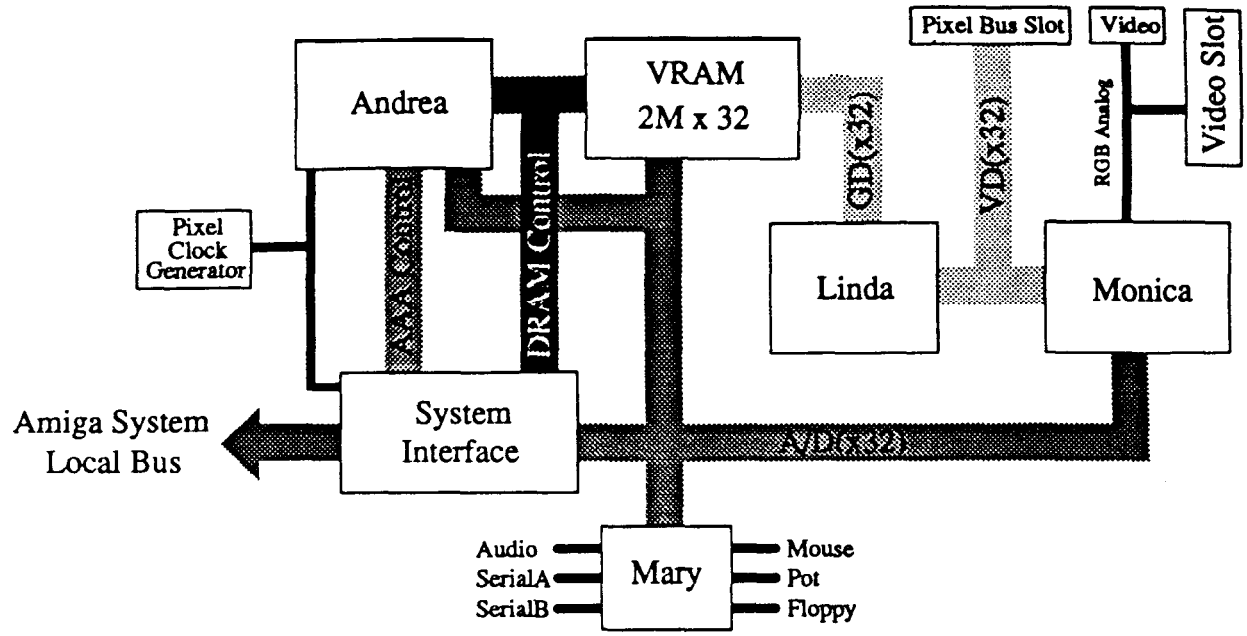
The interface to any kind of system will be implemented in some kind of "glue" chip, most likely a new gate array, though this function certainly could be implemented in a PAL and



TTL based circuit, as on the AAA prototype system. Ideally, uch a glue chip will do a variety of jobs. It should buffer data between the CPU/Local bus and the chip bus. A four longword deep buffer here will allow a CPU to write to chip RAM without an immediate delay, and it can also be used to match data rates between the chip and local buses, which are most likely running at different speeds. Andrea requires some kind of external logic to properly drive RAS* lines to each DRAM bank -- this also might be implemented in such a glue device. This will be especially true if its desirable to build an AAA system that handles VRAM in some kind of SIMM, since a SIMM wouldn't naturally drop into AAA's addressing and AUTOCONFIG scheme. The glue chip may also have something to do with pixel clock generation, though the main pixel clock generator will more than likely be a custom clock synthesis device.

The "pixel bus slot" shown is an expansion slot that makes the pixel bus (video data bus) available for expansion. This isn't the pixel output, but the raw data sent from memory to Monica. The main reason for providing some kind of expansion here is to support the framegrabber mode of AAA. In this mode, Linda reverses direction. Data from the pixel bus (supplied by a video capture bus of some kind, in an appropriate format) goes into Linda and is read out and stored by Andrea. This mode also requires the system to be slaved to an external pixel clock, like in genlock mode.

Single AAA systems can also support a traditional AA-compatible video slot. This can support devices that use either analog video or 25-bit digital video. Not every AA-compatible video &vice will necessarily work here, of course, since the pixel clock and display rates can be significantly higher than those possible in AA systems. It's not obvious, however, that a new type of video slot would be a better solution, though perhaps some of the reserved pins on the AA slot specification will be used for extra AAA signals in actual AAA implementations.



3.6 The Dual System

An alternate AAA system configuration is the so-called "dual" system. This system uses an Andrea and Mary as before, but contains two Lindas and two Monicas. Andrea, Mary, and Monica still sit on a 32-bit random-access bus, but the display path is now 64-bits wide. Lindas and Monicas are paired in "even" and "odd" groups to process, in turn, even and odd pixels. The pixel rate generated at output is twice the pixel clock rate, so pixel rates as high as 110 MHz can be reached with this kind of system.

In order to support HAM mode, which is of course dependent on past pixels, each Monica indicates its last CLUT access and HAM operation on a special HAM bus, and in turn snoops the HAM bus of the other Monica. In order to generate the full-speed video output and still use the Monica video DACs, each Monica chip has a direction control for its pixel bus. In this setup, the even Monica puts pixel information out, while the odd Monica takes that same information in and feeds it, along with its own pixel data, to the DACs. Both 24-bit values go to the DACs in the same pixel clock

This system supports a 64-bit pixel bus for video capture and other similar operations. It's possible to support an AA style 25-bit digital video slot as well, but with even more limitations. Since the pixel bus of the odd Monica acts as input, only every other pixel is actually available in digital form on this dual system. In some resolutions, that won't be a problem, in others it will **A** full featured digital video port would require an external DAC and either a multiplexer or 48-bit pixel path. A better solution for this kind of thing would be to adopt some kind of digital video transmission protocol as soon as one becomes available.

Chapter 4
Future System Concerns

The AAA chips obviously function as just part of a whole Amiga system. It's certainly possible to build a machine, like the Amiga 500, where the Amiga chips essentially are the whole system. Such a system would be composed of the AAA chips, a microprocessor, memory, some CIAs, analog stuff for audio, and one gate array for "glue". Of course, such a system is rather boring to talk about. It's also somewhat unlikely that early AAA systems will be of this flavor.

Taking the high-end route, there's considerably more to an Amiga system than the custom chips. It would, however, be a mistake to base things directly on past high-end systems like the A3000/A4000. The system architecture of those systems, while fine for the time, is lacking in a number of important areas. The most obvious factor is that the A3000 architecture isn't very modular or flexible. It was designed to support our ideas for the A3000, nothing more. Even in the A4000, extra logic was necessary to push this A3000 architecture is a slightly different direction. On the AAA prototype motherboard, there is extensive high-speed PAL logic necessary to implement a servicable but unsophisticated AAA system.

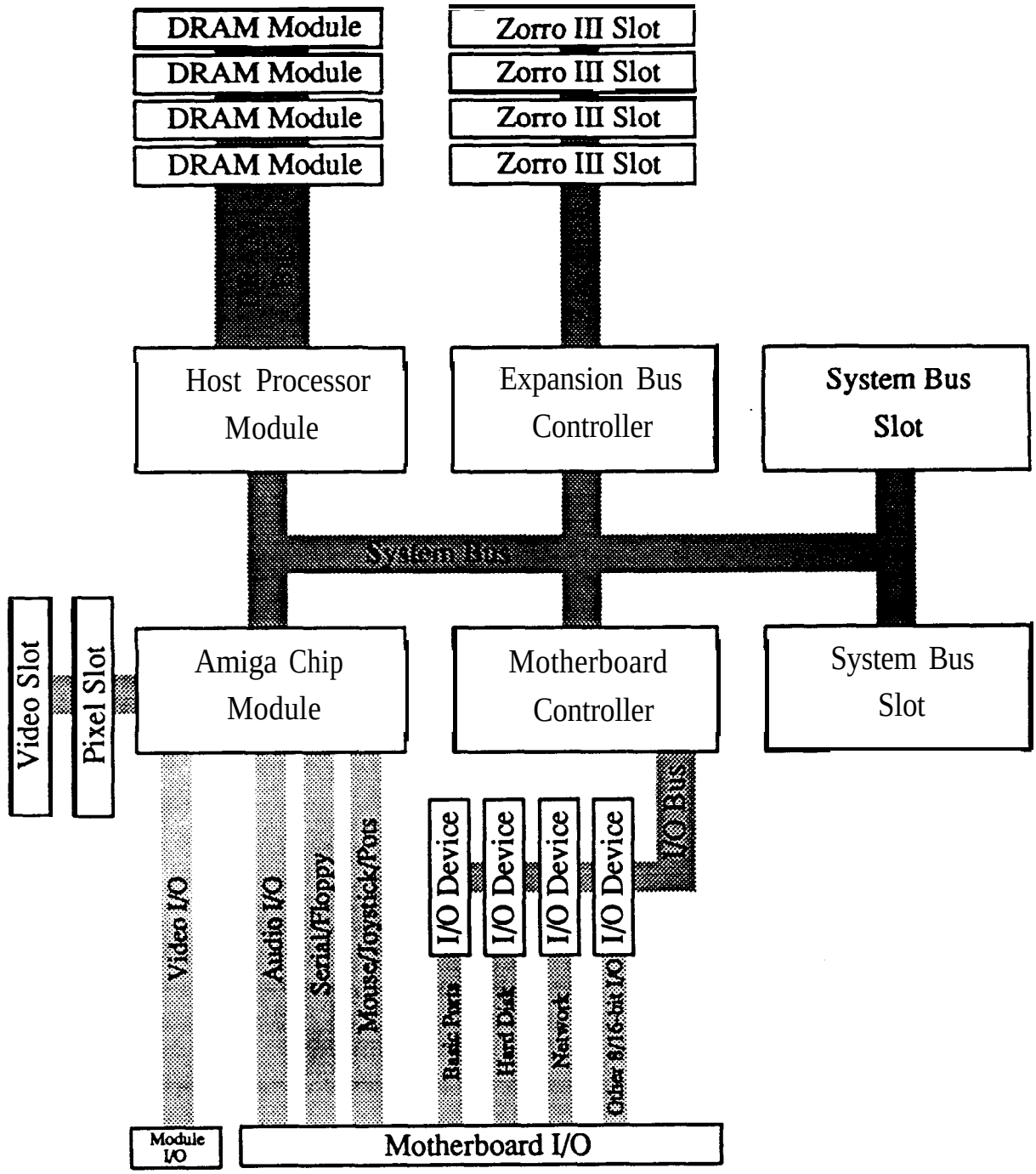
The primary goal of an advanced Amiga system can be summed up in one word: ***modularity.*** Such a new system, both logically and physically, is composed of several interchangable subsystems. No one piece has any unnatural dependence on any other; interconnections between the system components have to be based on intentional system standards, not chance implementation details.

4.1 The **System Bus**

The next generation system should have a processor-independent system bus optimised for chip to chip interconnect. For the most part this replaces the traditional CPU-specific local bus found on previous Amiga systems. This establishes a standard to which several generations of new system and, eventually, Amiga chips can be designed. Since each major system chip hooks into the system bus independently of any other, this finally breaks the interdependence of chips in a chip set, allowing upgrades as necessary to any piece of the system.

4.2 **The Motherboard**

The motherboard for such a system contains just the basics that will be needed by every system. This will certainly include a number of basic I/O chips for the standard ports on that machine. The CPU, Amiga chips, and various other elements of the system are located on separate modules. These don't necessarily have to be physically located on different cards, but ideally they will be. Not only does this make motherboard upgrade much easier, but it allows several different motherboards to be designed using the same plug-in modules, and it allows Commodore to easily support more options in system and processor makeup.



'The heart of the motherboard is the motherboard controller. This manages motherboard based I/O, such as CIA chips, hard disk interface (IDE, SCSI, whatever), network, etc. This also acts as a support for the system bus, handling arbitration of bus master and interrupts. Most I/O chips sit on a flexible 8/16 bit I/O bus defined by the system controller. Many programmable chip selects and interrupt inputs allow new I/O devices to be added as time goes by without extra glue or the need to redesign the motherboard controller.

43 The Amiga Chip Module

For the first time in an Amiga system, the Amiga chips are located on a plug-in module rather than fixed into the motherboard. This certainly allows easy upgrade to new Amiga chip sets, but its also a very reasonably thing to consider just in light of the AAA chip set, since a veriety of configurations can be made just with AAA, even if no other Amiga chips were considered. The Amiga module connects to the system bus and to a special motherboard-support connector. The system interface device glues Andrea to the system bus, providing the proper bus translation, data FIFOing, and chip selects. The support connector routes general Amiga resources, such as audio, floppy and UART lines, to the appropriate I/O connectors on the motherboard. Digital and analog video also go off this way to mate with video expansion connectors. The main video connector and other connectors specific to the Amiga module will be provided directly on the Amiga module.

4.4 The Host Processor Module

The main system processor is supported on the host processor module. This card has one connector to the system bus, and it gets the first system bus access on boot up by default. It has a second connector to a wide DRAM bus located on the motherboard. While the motherboard houses this DRAM, it's totally up to the host module to drive the DRAM bus. This of course means that the host module must contain a DRAM controller. In most cases, the DRAM controller will be intregrated with the system bus controller to provide a low cost interface between host CPU and the motherboard. Locating the DRAM controller on the host module allows DRAM, the most speed-critical element of the system, to be optimized for any host processor chosen.

4.5 The Expansion Controller

The expansion controller is another system bus chip that does conversions to and from the Zorro bus protocols. This can be located on a motherboard (maybe for towers) or on a system bus module (maybe for desktops). It is completely optional -- if a Zorro bus isn't desired, or should be an option on some systems, fine.

4.6 Open System Bus Slots

There will likely be a free system bus slot or two on most expandable machines. These support various kinds of high-speed expansion: fast peripherals, processor farms, DSPs, etc. The number of open system bus slots will of course depend on the final system bus specification, the space available on a given motherboard, etc. Due to the anticipated speed and purpose of such slots, it's not expected that there will be more than two or three open slots in any system, and the module cards aren't likely to be very large. These aren't designed to replace Zorro III as a general purpose expansion bus. Instead, the system bus should be thought of as a more flexible local bus/coprocessor slot.