Author: Laurent Lefebvre

Issue To: | **Copy No:**

R400 Sequencer Specification

SEQ

Version 0.1

Overview: This is an architectural specification for the R400 Sequencer block (SEQ). It provides an overview of the required capabilities and expected uses of the block. It also describes the block interfaces, internal sub-blocks, and provides internal state diagrams.

AUTOMATICALLY UPDATED FIELDS:
Document Location: D:\Perforce\r400\arch\doc\gfx\MC\R400 MemCtl.doc
Current Intranet Search Title: R400 Memory Controller Architectural Specification

APPROVALS

Name/Dept	Signature/Date

Remarks:

Table Of Contents

Revision Changes:

Rev 0.1 (Laurent Lefebvre) Date: May 7, 2001	First draft.

1. Overview

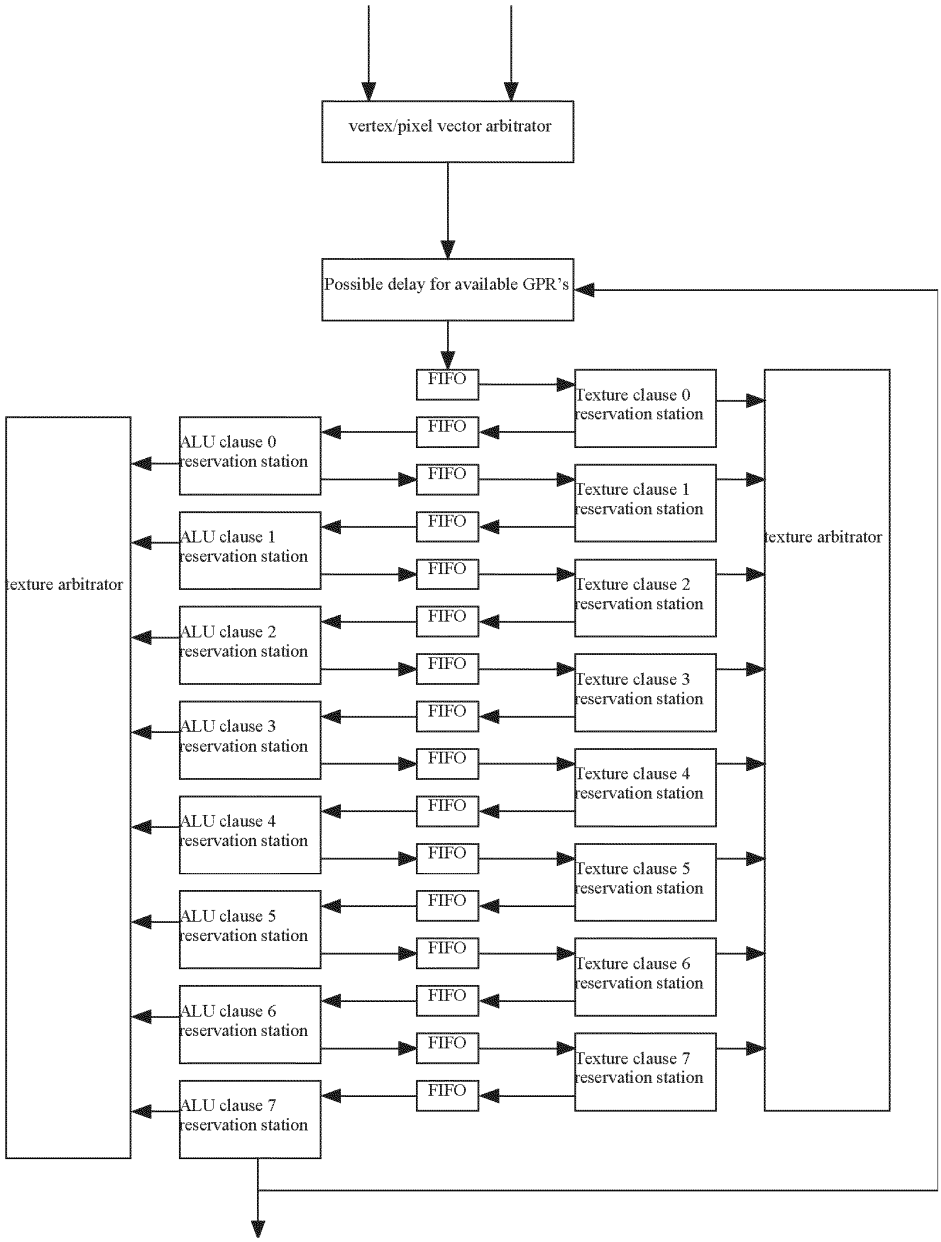
The sequencer first arbitrates between vectors of 16 vertices that arrive directly from primitive assembly and vectors of 8 quads (32 pixels) that are generated in the raster engine.

The vertex or pixel program specifies how many GPR's it needs to execute. The sequencer will not start the next vector until the needed space is available.

The sequencer is based on the R300 design. It chooses an ALU clause and a texture clause to execute, and execute all of the instructions in a clause before looking for a new clause of the same type. Each vector will have eight texture and eight alu clauses, but clauses do not need to contain instructions. A vector of pixels or vertices ping-pongs along the sequencer FIFO, bouncing from texture reservation station to alu reservation station. A FIFO exists between each reservation stage, holding up vectors until the vector currently occupying a reservation station has left. A vector at a reservation station can be chosen to execute. The sequencer looks at all eight alu reservation stations to choose an alu clause to execute and all eight texture stations to choose a texture clause to execute. The arbitrator will give priority to clauses/reservation stations closer to the top of the pipeline. It will not execute an alu clause until the texture fetches initiated by the previous texture clause have completed.

To support the shader pipe the raster engine also contains the shader instruction cache and constant store.

1.1 Top Level Block Diagram



The rasterizer always checks the vertices FIFO first and if allowed by the sequencer sends the data to the shader. If the vertex FIFO is empty then, the rasterizer takes the first entry of the pixel FIFO (a vector of 32 pixels) and sends it to the interpolators. Then the sequencer takes control of the packet.

On receipt of a packet, the input state machine (not pictured but just before the first FIFO) allocated enough space in the registers to store the interpolatoted values and temporaries. Following this, the input state machine stacks the packet in the first FIFO.

On receipt of a command, the level 0 texture machine issues a texure request and corresponding register address for the texture address (ta). A small command (tcmd) is passed to the texture system identifying the current level number

(0) as well as the register set being used. One texture request is sent every 4 clocks causing the texturing of four 2x2s worth of data.

Uppon recept of the return data (identified by the tcmd containing the level number 0), the level 0 texture machine issues a register address for the return value (td). Then, it puts the finished packet in FIFO 1.

On receipt of a command, the level 0 ALU machine issues a complete set of level 0 shader instructions. For each instruction, the state machine generates 3 source addresses, one destination address (2 cycles later) and an instruction id wich is used to index into the instruction store. Once the last instruction as been issued, the packet is put into FIFO 2. Note that in the case of a pixel packet, the two vectors of 16 pixels are interleaved in order to hide the latency of the ALUs (8 cycles).

All other level process in the same way until the packet finally reaches the last ALU machine (8). On completion of the level 8 ALU clause, a valid bit is sent to the Render Backend wich picks up the color data. This requires that the last instruction writes to the output register – a condition that is almost always true. If the packet was a vertex packet, instead of sending the valid bit to the RB, it is sent to the PA, which picks up the data a puts it into the vertex store.

Only one ALU state machine may have access to the SRAM address bus or the instruction decode bus at one time. Similarly, only one texture state machine may have access to the SRAM address bus at one time. Arbitration is performed by two arbitrer blocks (one for the ALU state machines and one for the texture state machines). The arbitrers always favor the higher number state machines, preventing a bunch of half finished jobs from clogging up the SRAMS.

Each state machine maintains an address pointer specifying where the 16 (or 32) entries vector is located in the SRAM (the texture machine has two pointers one for the read address and one for the write). Upon completion of its job, the address pointer is incremented by a predefined amount equal to the total number of registers required by the shading code. A comparison of the address pointer for the first state machine in the chain (the input state machine), and the last machine in the chain (the level 8 ALU machine), gives an indication of how much unallocated SRAM memory is available. When this number falls below a preset watermark, the input state machine will stall the rasterizer preventing new data from entering the chain.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.