



UNIVERSITÀ DEGLI STUDI DI ANCONA

FACOLTÀ DI INGEGNERIA

Corso di Laurea in Elettronica

TESINA DI ELETTRONICA APPLICATA II

PROGETTAZIONE IN VHDL
DEI PROCESSORI MATEMATICI
VERTEX SHADER E PIXEL SHADER

Prof. Massimo CONTI

Studenti: Alessandro GIORGETTI
Daniele PETRACCINI
Francesco GARELLI
Gian Maria RICCI

Anno Accademico 2001-2002

Alla mia mamma, mi manchi tanto.

(Alessandro)

A Marta, colei che ha allietato i miei ultimi giorni di servizio civile, portando all'interno della casa di riposo un raggio di sole.

(Daniele)

Ringrazio Alessandro, Daniele e Gian Maria di essere stati pazienti con me e di avermi fatto capire come si dovrebbe lavorare per ottenere un buon risultato.

Un ringraziamento anche alla mia Maria che mi aiuta nei momenti di bisogno.

(Francesco)

Alla mia famiglia e a Michela che sopportano con pazienza i miei lunghi momenti di isolamento davanti al compilatore e ai rimanenti membri del team Nablasoft, per il lavoro svolto insieme.

(Gian Maria)

SOMMARIO

PARTE PRIMA	1
Introduzione alle Pipeline Grafiche 3D	1
CAPITOLO 1. INTRODUZIONE ALLE PIPELINE GRAFICHE 3D	2
1.1. INTRODUZIONE	2
1.1.1 <i>Significato di Pipeline</i>	2
1.1.2 <i>Matrici</i>	3
1.1.3 <i>Coordinate Omogenee</i>	4
1.2. PIPELINE 3D	4
1.3. DATABASE TRAVERSAL	5
1.4. GEOMETRY SUBSYSTEM	7
1.4.1 <i>Model And View Transform</i>	7
1.5. LIGHTING	11
1.5.1 <i>Introduzione</i>	11
1.5.2 <i>Light</i>	11
1.5.3 <i>Material</i>	13
1.5.4 <i>Shading</i>	13
1.5.5 <i>Modellazione della luce</i>	14
1.5.5.a <i>Diffuse Component</i>	14
1.5.5.b <i>Specular Component</i>	16
1.5.5.c <i>Ambient Component</i>	17
1.5.6 <i>Light Equation</i>	17
1.6. VIEWING TRANSFORMATION	19
1.6.1 <i>Parallel Projection</i>	20
1.6.1.a <i>Proiezioni ortografiche</i>	21
1.6.1.b <i>Proiezioni oblique</i>	22
1.6.2 <i>Perspective Projection</i>	22
1.6.3 <i>Viewing Volume</i>	23
1.6.4 <i>Canonical View Volume</i>	24
1.6.5 <i>Clipping & Screen Mapping</i>	26
1.7. PIXEL STAGE	29
1.7.1 <i>Introduzione</i>	29
1.7.2 <i>Scan-conversion</i>	29
1.7.3 <i>Texturing</i>	31
1.7.4 <i>Shading</i>	31
1.7.5 <i>Visibile Surface Determination mediante Z-buffer</i>	32
1.8. PERFORMANCES	33
1.8.1 <i>Geometry Calculations</i>	34
1.8.2 <i>Rasterizzazione ed accessi al Frame Buffer</i>	35
1.9. VERTEX SHADER AND PIXEL SHADER ARCHITECTURE	37
1.9.1 <i>Introduzione</i>	37
1.9.2 <i>Vertex Shader</i>	40
1.9.3 <i>Pixel Shader</i>	42
1.9.3.a <i>Color Registers</i>	43
1.9.3.b <i>Constant Registers</i>	43
1.9.3.c <i>Temporary Registers</i>	43
1.9.3.d <i>Texture Registers</i>	43
1.9.3.e <i>Pixel Shader ALU</i>	44
1.9.4 <i>Impieghi comuni di Vertex Shader e Pixel Shader</i>	44

PARTE SECONDA	47
Introduzione al linguaggio VHDL.....	47
CAPITOLO 2. INTRODUZIONE AL LINGUAGGIO VHDL	48
2.1. COME NASCE	48
2.2. A COSA SERVE: METODOLOGIA DI PROGETTO DI UN SISTEMA DIGITALE	48
2.3. DIFFERENTI TIPI DI SIMULAZIONI	49
2.4. SISTEMI DI HARDWARE SYNTHESIS	50
2.5. PREREQUISITI E CARATTERISTICHE DEL LINGUAGGIO	51
2.6. PACKAGES & LIBRARIES	52
2.7. PROGETTAZIONE TOP-DOWN, REALIZZAZIONE BOTTOM-UP	52
2.8. ELEMENTI DEL LINGUAGGIO	53
2.9. I TIPI DI DATO NEL DETTAGLIO	54
2.9.1 <i>Integer & Real</i>	55
2.9.2 <i>Tipi enumerati</i>	55
2.9.3 <i>Tipi che rappresentano entità fisiche</i>	55
2.9.4 <i>Tipo aggregato – ARRAY</i>	55
2.9.5 <i>Tipo aggregato – RECORD</i>	56
2.9.6 <i>Subtype</i>	56
2.9.7 <i>Alias</i>	56
2.9.8 <i>Puntatori – ACCESS TYPE</i>	56
2.9.9 <i>File types</i>	57
2.10. DESCRIZIONE DEI COMPONENTI	57
2.10.1 <i>ENTITY</i>	57
2.10.2 <i>ARCHITECTURE</i>	58
2.10.3 <i>CONFIGURATION</i>	59
2.11. TIPI DI DATO E FUNZIONI DEFINITI DALL'UTENTE	60
2.11.1 <i>PACKAGE & PACKAGE BODY</i>	60
2.12. BLOCK & COMPONENT	61
2.13. SUBPROGRAMS	62
2.13.1 <i>Function</i>	63
2.13.2 <i>Procedures</i>	63
2.14. TENERE SOTTO CONTROLLO L'ESECUZIONE – ASSERT	64
2.15. L'IMPORTANZA DEL TEMPO	64
2.16. OPERAZIONI DI ASSEGNAZIONE	65
2.17. ASSEGNAZIONE DI VARIABILI E COSTANTI	65
2.17.1 <i>Assegnazione di Segnali</i>	65
2.17.2 <i>INERTIAL delay – Risposta in frequenza limitata</i>	67
2.17.3 <i>TRANSPORT delay – Risposta in frequenza illimitata</i>	68
2.17.4 <i>Particolarità degli assegnamenti</i>	68
2.18. EVENTI E TRANSAZIONI	68
2.18.1 <i>Transazioni multiple sullo stesso driver</i>	69
2.19. DELTA DELAY (δ-DELAY)	70
2.19.1 <i>Pericoli del δ-delay</i>	71
2.20. CONCURRENT PROCESSING	72
2.20.1 <i>Descrivere un blocco sequenziale – PROCESS</i>	72
2.20.2 <i>Un esempio di elaborazione concorrente</i>	72
2.20.3 <i>Processes & WAIT statement</i>	74

PARTE TERZA	75
Progettazione in VHDL del Vertex Shader	75
CAPITOLO 3. REALIZZARE UNA ALU FLOATING POINT	76
3.1. FUNZIONI DI CONVERSIONE	76
3.1.1 Conversioni di quantità integer	76
3.1.2 Conversioni di quantità floating point e specifica IEEE	78
3.1.3 Breve descrizione del formato single precision floating point	79
3.1.4 Funzioni di conversione: da rappresentazione binaria a reale.....	79
3.1.5 Conversione da real a rappresentazione binaria.	80
3.2. TEST DELLE FUNZIONI DI CONVERSIONE E PRIMI PASSI CON IL MODELSIM	83
3.2.1 Lavorare con il ModelSim.....	83
3.2.2 Strutturazione di un file di test in VHDL	84
3.2.3 Simulazione	86
3.2.4 Debug.....	87
3.2.5 La struttura ALU_FP.....	88
3.3. UTILIZZARE FILE CON IL VHDL	90
3.3.1 Simulazione mediante testbench automatici	90
3.3.2 Scrivere float in un file.....	91
3.3.3 File di testo e libreria standard	91
3.3.4 Differenze di versione	93
CAPITOLO 4. IMPLEMENTAZIONE DEL VERTEX SHADER	95
4.1. DESCRIZIONE DELLA STRUTTURA “VERTEX SHADER”	95
4.1.1 Da dove nasce il concetto di Vertex Shader (VS)	95
4.1.2 La struttura di un Vertex Shader.....	96
4.1.3 Vertex Shader in dettaglio.....	97
4.1.4 Decodificare gli opcode ed il formato binario.....	100
4.2. SIMULAZIONE DEL PROCESSO DI PROGRAMMAZIONE	101
4.2.1 Prima fase: la decisione di come strutturare l’entità	101
4.2.2 Due processi distinti (programmazione, esecuzione)	101
4.2.3 Processo di programmazione.....	102
4.2.4 Processo di dissassemblaggio.....	104
4.2.5 La simulazione	105
4.3. UNA ALU PER IL VERTEX SHADER	106
4.3.1 Requisiti di base.....	106
4.3.2 La struttura interna dell’ALU_VS.....	107
4.3.3 Analisi del processo di funzionamento.....	109
4.3.4 Ancora sull’I/O in VHDL.....	113
4.3.5 Simulazione	116
4.4. COMPONENTI BASE DEL VERTEX SHADER	118
4.4.1 Disegnare lo schema.....	118
4.4.2 Latch n-bit, latch_r n-bit.....	118
4.4.3 Multiplexer e demultiplexer	120
4.4.4 I banchi di registri.....	122
4.4.5 Program counter e circuito per il suo incremento.....	125
4.4.6 Swizzler ed inverter.....	127
4.4.7 Dec_Src e Dec_Dest	129
Nella pagina seguente viene mostrato il diagramma di flusso del DEC_DEST.....	130
4.4.8 Ritardi	131
4.5. STRUTTURA E SIMULAZIONI DEL VERTEX SHADER	132

4.5.1 Progettazione strutturale	132
4.5.2 Bus principali.....	132
4.5.3 Program counter e ciclo istruzione (sezione decodifica).....	134
4.5.4 La sezione di decodifica.....	136
4.5.5 L'unità di controllo	137
4.5.6 Opcode particolari.....	142
4.5.7 L'unità di test e la simulazione	143
PARTE QUARTA.....	148
Progettazione in VHDL del Pixel Shader v.1.4	148
CAPITOLO 5. SPECIFICHE DI PROGETTO DEL PIXEL SHADER V.1.4.....	149
5.1. LINEE GENERALI DEL PROGETTO	149
5.2. TERMINOLOGIA	150
5.3. SCHEMA DI PRINCIPIO DEL PIXEL SHADER V.1.4.....	150
5.3.1 Banco di registri <i>c</i>	151
5.3.2 Banco di registri <i>r</i>	151
5.3.3 Banco di registri <i>t</i>	151
5.3.4 Banco di registri <i>v</i>	152
5.3.5 Instruction Pool.....	152
5.3.6 Pixel Shader ALU.....	152
5.3.7 Funzionamento del Pixel Shader	153
5.4. ARCHITETTURA INTERNA DELL'ALU	153
5.5. ISTRUZIONI DEL PIXEL SHADER V.1.4.....	155
5.5.1 Tipi di istruzioni.....	155
5.5.2 Struttura di uno shader	155
5.5.3 Istruzioni PS, DEF, PHASE.....	156
5.5.4 Istruzioni aritmetiche	156
5.5.5 Istruzioni di texture addressing.....	159
5.6. SOURCE REGISTER SELECTOR (SWIZZLER)	160
5.7. SOURCE REGISTER MODIFIER	161
5.8. INSTRUCTION MODIFIER	162
5.9. DESTINATION REGISTER WRITE MASK	163
5.10. MODIFICATORI E MASCHERE PER LE ISTRUZIONI TEXLD E TEXCRD	164
5.10.1 Source register selectors.....	164
5.10.2 Source register modifiers.....	164
5.10.3 Destination register write mask.....	165
5.11. INSTRUCTION PAIRING	165
5.12. PIXEL SHADER ASSEMBLER	166
5.12.1 Struttura di un file .pso	166
5.12.2 Codifica delle istruzioni.....	167
5.12.3 OpCode	168
5.12.4 Codifica dei registri	168
5.13. STRUMENTI SOFTWARE UTILIZZATI NEL PROGETTO.....	170
5.14. ORGANIZZAZIONE DEL PROGETTO	171
CAPITOLO 6. TEST BENCH PER LA SIMULAZIONE IN VHDL DEL PROCESSORE.172	
6.1. INTRODUZIONE.....	172
6.2. CLOCK GENERATOR	174
6.3. TEXTURE SAMPLER	174
6.4. BEM MODULE	176

6.5. PIXEL SHADER	177
6.6. TEST UNIT	179
6.6.1 <i>HDL Designer e generazione di una macchina a stati finiti</i>	181
6.6.2 <i>IF + 3 PROCESSES</i>	183
6.6.3 <i>IF + 2 PROCESSES</i>	184
6.6.4 <i>CASE + 3 PROCESSES</i>	185
6.6.5 <i>CASE + 2 PROCESSES</i>	186
6.6.6 <i>Test Unit – Finite State Machine (FSM)</i>	186
CAPITOLO 7. ARCHITETTURA INTERNA DEL PIXEL SHADER V.1.4	196
7.1. SCHEMA A MACROBLOCCHI DEL PIXEL SHADER	196
7.2. ALU	199
7.2.1 <i>L'entità ALUX</i>	199
7.3. REGISTRI	200
7.3.1 <i>L'entità RegX</i>	201
7.3.2 <i>L'entità RegR</i>	202
7.3.3 <i>Le entità AddrSel1 e AddrSel2</i>	202
7.4. INSTRUCTION POOL	204
7.5. IL MACROBLOCCO CONTROLLO REGISTRI E DECODIFICA (1° PARTE)	206
7.5.1 <i>L'entità Write Register Selector (WRS)</i>	206
7.5.2 <i>Le entità Program Counter (PC) e Program Counter Increment (PCIncr)</i>	208
7.5.3 <i>Gli Instruction Latch</i>	209
7.5.4 <i>L'entità Source Information Processing (SIP)</i>	210
7.5.5 <i>SIP Driver</i>	212
7.5.6 <i>Source Decoder</i>	214
7.6. IL MACROBLOCCO SWIZZLING & MASKING	215
7.6.1 <i>L'entità Data Combiner (DC)</i>	215
7.6.2 <i>L'entità Source Swizzler (SS)</i>	216
7.6.3 <i>Source Masking</i>	217
7.6.4 <i>Swizzling & Masking: schema completo</i>	220
7.7. IL MACROBLOCCO MODIFICATORI SORGENTI	221
7.7.1 <i>L'entità Modifier</i>	221
7.8. RETE DI MULTIPLEXER ASSOCIATA ALLE ALU	224
7.9. IL MACROBLOCCO MODIFICATORI ISTRUZIONE	232
7.9.1 <i>L'entità Instruction Modifier (IM)</i>	232
7.10. IL MACROBLOCCO CONTROLLO REGISTRI E DECODIFICA (2° PARTE)	233
7.10.1 <i>L'entità Source Modifier Deliverer (SMD)</i>	234
7.10.2 <i>L'entità Source Modifier Decoder (SMDec)</i>	235
7.10.3 <i>L'entità Instruction Modifier Deliverer (IMD)</i>	237
7.10.4 <i>L'entità Destination Address Deliverer (DAD)</i>	237
7.10.5 <i>L'entità Destination Mask Generator (DMG)</i>	238
7.11. I BUS	240
7.11.1 <i>Il tipo BIT_BUS</i>	240
7.11.2 <i>Le entità BusConnector</i>	241
7.11.3 <i>Il componente Switch</i>	242
7.12. ULTERIORI COMPONENTI DEL PIXEL SHADER	243
7.12.1 <i>TexldDemux</i>	243
7.12.2 <i>Multiplexer per la selezione dell'Unità di Controllo</i>	244
7.12.3 <i>ExitLatch</i>	245
7.12.4 <i>Ripper</i>	245
7.13. UNITÀ DI CONTROLLO	246

7.13.1 <i>MainCU</i>	246
7.13.2 <i>La VectCU</i>	259
7.13.3 <i>La ScalCU</i>	262
7.14. UNA POSSIBILE REALIZZAZIONE CIRCUITALE DEL BANCO DI REGISTRI	265
7.15. SCHEMA COMPLETO DEL PIXEL SHADER V.1.4	268
CAPITOLO 8. SIMULAZIONI	269
8.1. INTRODUZIONE	269
8.2. CONFIGURAZIONE ED UTILIZZO DI VHDL SIMILI	269
8.3. INPUT ED OUTPUT DEL TEST BENCH	271
8.4. FORME D'ONDA E RISULTATO DELLA SIMULAZIONE	273
8.5. CONCLUSIONI E POSSIBILI MIGLIORAMENTI	276
Bibliografia	278

Bibliografia

PARTE PRIMA

Foley, van Dam, Feiner, Hughes, "Computer Graphics Principles and Practice"

Tomas Moller, Eric Haines, "Real-Time Rendering"

Alan H. Watt, "Advanced Animation and Rendering Techniques"

Nvidia, Developers Resources, sito web: <http://developer.nvidia.com/>

ATI, Developers resources, sito web: <http://www.ati.com/developer/index.html>

Extreme Tech Tutorials, sito web: <http://www.extremetech.com>

Lectures dalla Princeton university, sito web:

<http://www.cs.princeton.edu/courses/archive/fall99/cs426/lectures/>

Microsoft DirectX Developer Site, sito web: <http://www.microsoft.com/directx>

PARTE SECONDA

Zainalabedin Navabi, "VHDL: analysis and modeling of digital systems", McGraw-Hill

Peter J. Ashenden, "The VHDL Cookbook", disponibile all'indirizzo web:

<http://www.doc.eng.cmu.ac.th/course/cpe321/materials/vhdl/VHDL-Cookbook.pdf>

"The Hamburg VHDL Archive", sito web: <http://tech-www.informatik.uni-hamburg.de/vhdl/>

PARTE TERZA

Peter J. Ashenden, "The VHDL Cookbook", disponibile all'indirizzo web:

<http://www.doc.eng.cmu.ac.th/course/cpe321/materials/vhdl/VHDL-Cookbook.pdf>

Peter J. Ashenden, "The Designer's Guide To VHDL Second Edition",

Morgan Kaufmann Publishers 1996

Microsoft MSDN, sito web: <http://www.msdn.microsoft.com>

Microsoft direct 8.1 SDK, sito web:

<http://msdn.microsoft.com/library/default.asp?url=/nhp/Default.asp?contentid=28000410>

Nvidia, Developers Resources, sito web: <http://developer.nvidia.com/>

PARTE QUARTA

Microsoft MSDN, sito web: <http://www.msdn.microsoft.com>

Microsoft direct 8.1 SDK, sito web:

<http://msdn.microsoft.com/library/default.asp?url=/nhp/Default.asp?contentid=28000410>

ATI, Developers resources, sito web: <http://www.ati.com/developer/index.html>

Nvidia, Developers Resources, sito web: <http://developer.nvidia.com/>

Direct3D/DirectX Graphics Game Programming, sito web:

<http://www.shaderx.com/direct3d.net/index.html>

SHADER STUDIO, sito web: <http://www.shaderstudio.com/>

Autori



Alessandro Giorgetti è nato a Loreto (AN) il 23 gennaio 1974.
Ha conseguito la maturità scientifica nel 1993 al Liceo Scientifico “Ettore Majorana” di Osimo (AN).

E’ attualmente iscritto presso la Facoltà di Ingegneria di Ancona, corso di laurea in Ing. Elettronica (indirizzo microelettronica).

Membro del Team NablaSoft; sviluppatore C/C++, VB/VB.NET, C#, VHDL.

e-mail: guardian@nablasoft.com



Daniele Petraccini è nato a Cesena (FO) il 3 aprile 1974.

Ha conseguito il diploma di ragioniere e perito commerciale nel 1993 all’Istituto Tecnico Commerciale “Filippo Corridoni” di Osimo (AN).

E’ attualmente iscritto presso la Facoltà di Ingegneria di Ancona, corso di laurea in Ing. Elettronica (indirizzo microelettronica).

Membro del Team NablaSoft; ricercatore matematico.

e-mail: petra@nablasoft.com



Gian Maria Ricci è nato a Sassoferrato (AN) il 18 agosto 1974.

Ha conseguito la maturità scientifica nel 1993 al Liceo Scientifico “Vito Volterra” di Sassoferrato (AN).

E’ attualmente iscritto presso la Facoltà di Ingegneria di Ancona, corso di laurea in Ing. Elettronica (indirizzo microelettronica).

Membro del Team NablaSoft; sviluppatore C/C++, VB/VB.NET, Assembly, VHDL.

e-mail: alkampfer@nablasoft.com



Francesco Garelli è nato ad Ancona il 21 gennaio 1973.

Ha conseguito la maturità scientifica nel 1994.

E’ attualmente iscritto presso la Facoltà di Ingegneria di Ancona, corso di laurea in Ing. Elettronica (indirizzo microelettronica).

Membro del Team NablaSoft; sviluppatore C/C++, Assembly e VHDL.

e-mail: vliw@nablasoft.com