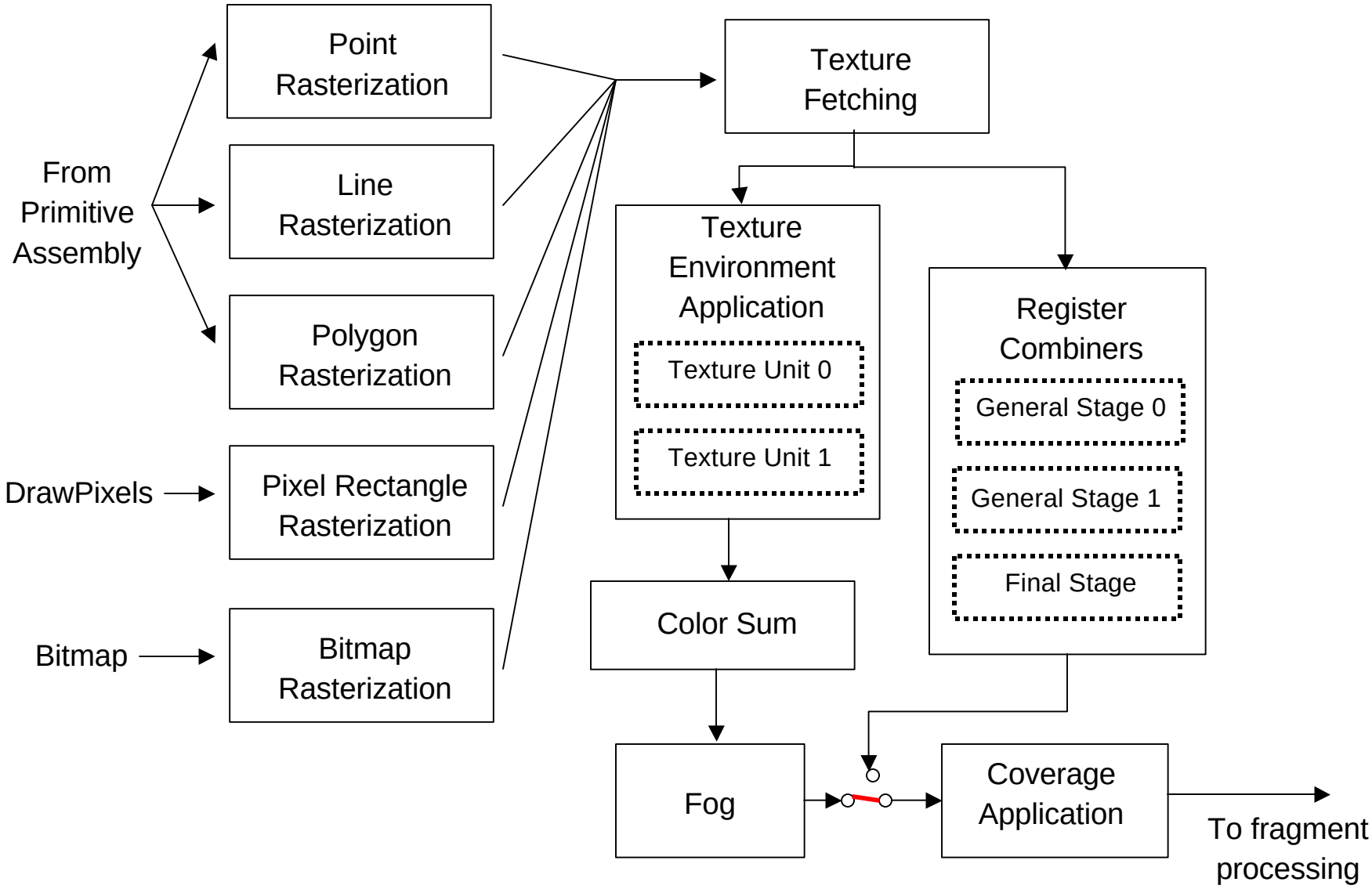
GeForce 256 and RIVA TNT Combiners

How to best utilize the per-pixel operations using OpenGL on NVIDIA Graphics Processors

NVIDIA OpenGL Combiners

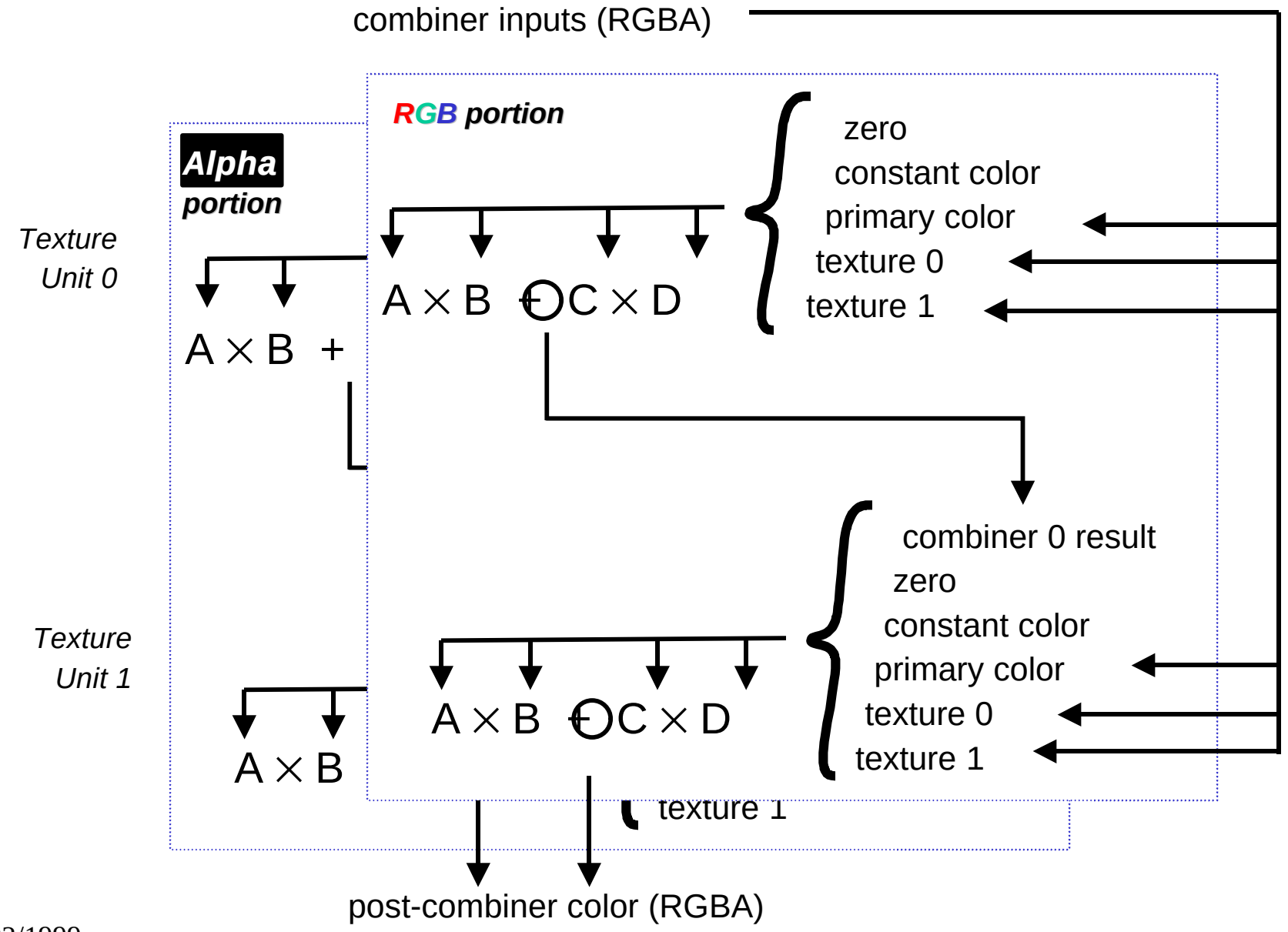
(see Figure 3.1 of OpenGL 1.2 spec)



Which Texture Environment or Combiner Extension to use?

- **Base OpenGL texture environment**
  - supports modulate, replace, blend, decal
  - all OpenGL implementation support this
- **EXT_texture_env_add**
  - supports add
  - widely supported extension, but not guaranteed
- **EXT_texture_env_combine**
  - supports AB+C, AB+(1-A)C
  - additional scale & bias capability
  - user-defined constant
  - multi-vendor extension (NVIDIA & ATI)
- **NV_texture_env_combine4**
  - supports AB+CD, generalizes EXT_texture_env_combine
  - TNT & later NVIDIA graphics processors
- **NV_register_combiners**
  - register model, non-linear data flow
  - signed math, input mappings, multiple dot products
  - subsumes texture environment, color sum, & fog stages
  - additional inputs: fog color & factor, & secondary color
  - number of stages independent of active textures
  - GeForce & future NVIDIA graphics processors (not TNT)

NV_texture_env_combine4



NV_register_combiners
Functional Overview

- **overrides texture stages/environment, color sum, and fog in current APIs**

- **signed math (negative one to positive range)**
  - extended range through scaling

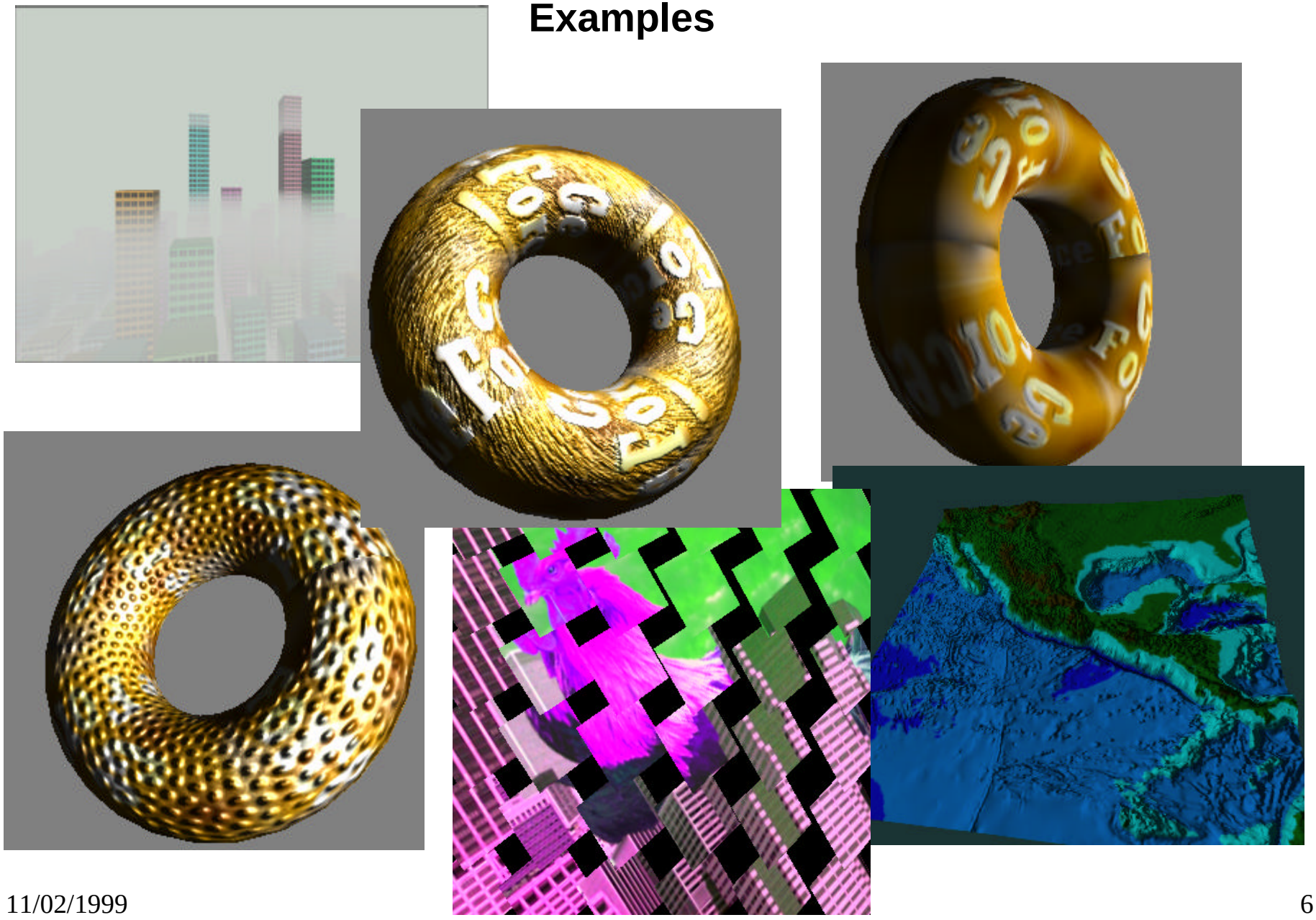
- **dot products for lighting and image processing applications**
  - specially designed for specular, diffuse, and ambient per-pixel lighting
  - object space bump map lighting
  - tangent space bump map lighting
  - non-photorealistic lighting models
  - post-filtering 3x3 color matrix for color space conversions

- **register model supports non-linear data flows**
  - superior to linear chain in current APIs

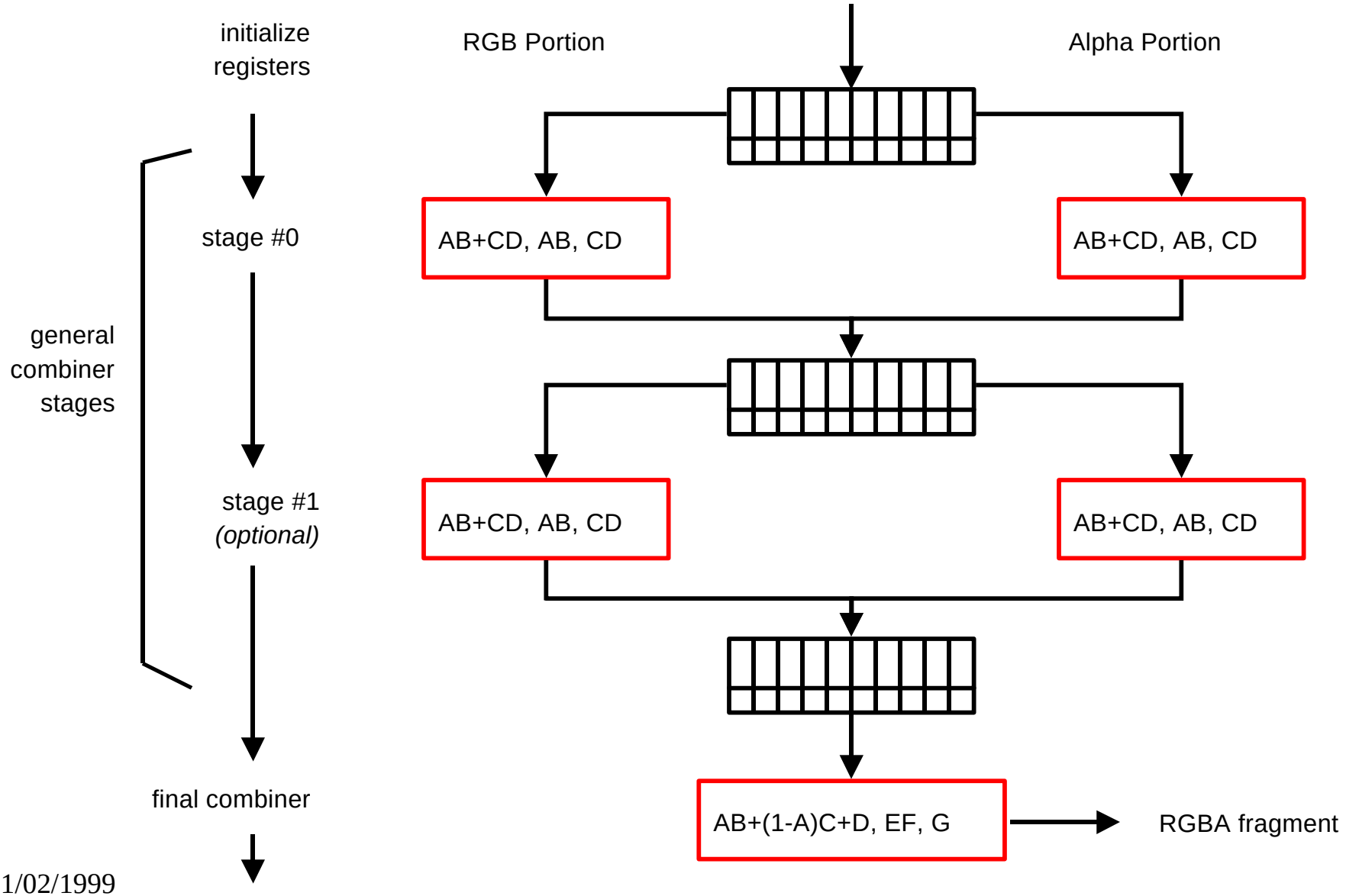
- **effectively, a VLIW instruction set for fragment coloring**

- **very efficient hardware implementation**

NV_register_combiners Examples



NV_register_combiners
Register Combiner Operational Overview



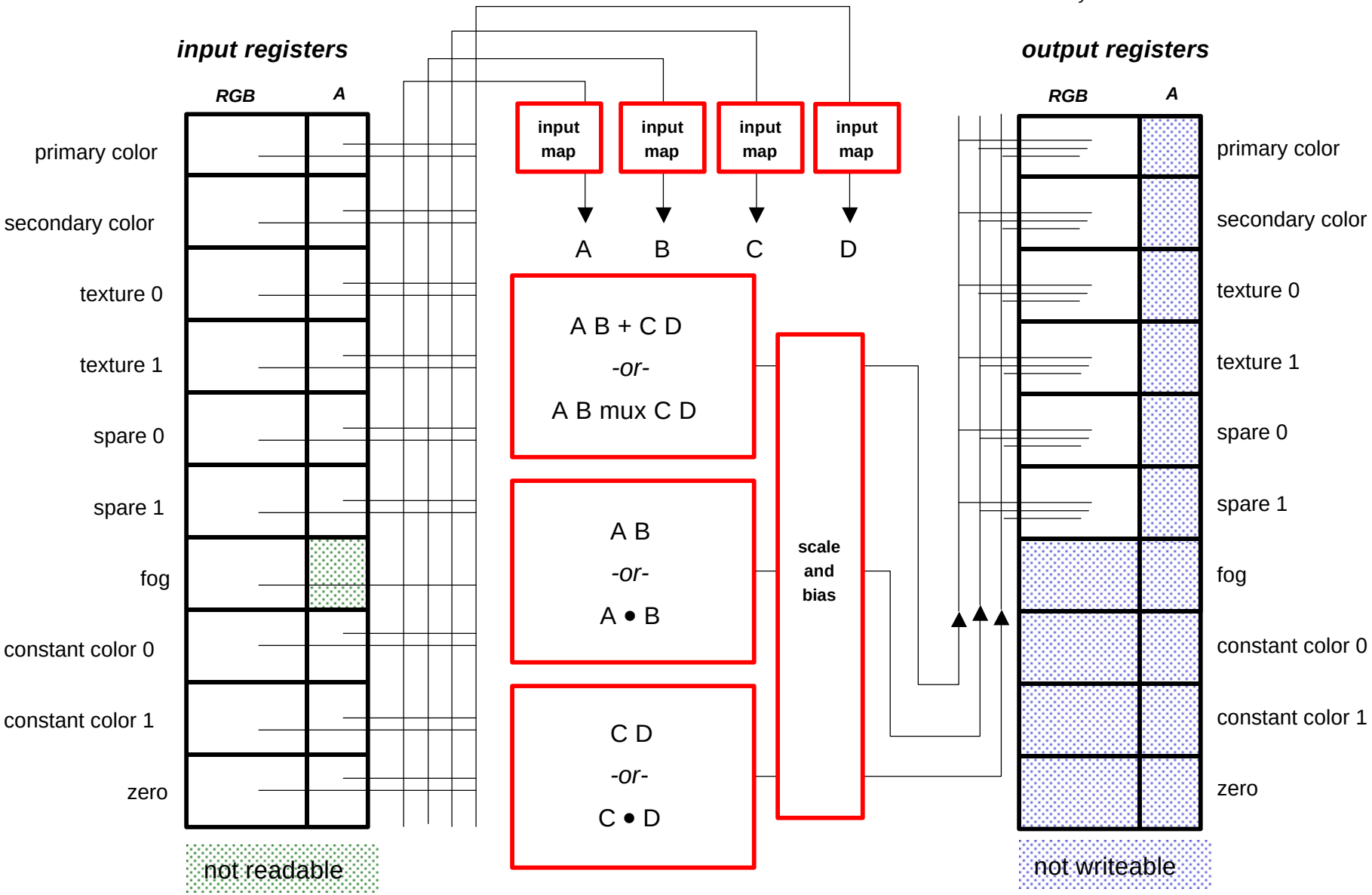
NV_register_combiners Register Set

- **primary (diffuse) color**
  - initialized to RGBA of fragment's primary color
- **secondary (specular) color**
  - initialized to RGB of fragment's secondary color
  - alpha initially undefined
- **texture 0 & texture 1**
  - initialized to fragment's filtered RGBA texel from numbered texture unit
  - undefined if numbered texture unit is disabled
- **spare 0 & spare 1**
  - initially undefined
- **fog**
  - RGB is current fog color
  - Alpha is fragment's fog factor (only available in final combiner)
  - read-only
- **constant color 0 & constant color 1**
  - initialized to user-defined RGBA value
  - read-only
- **zero**
  - constant, read-only value of zero

NV_register_combiners

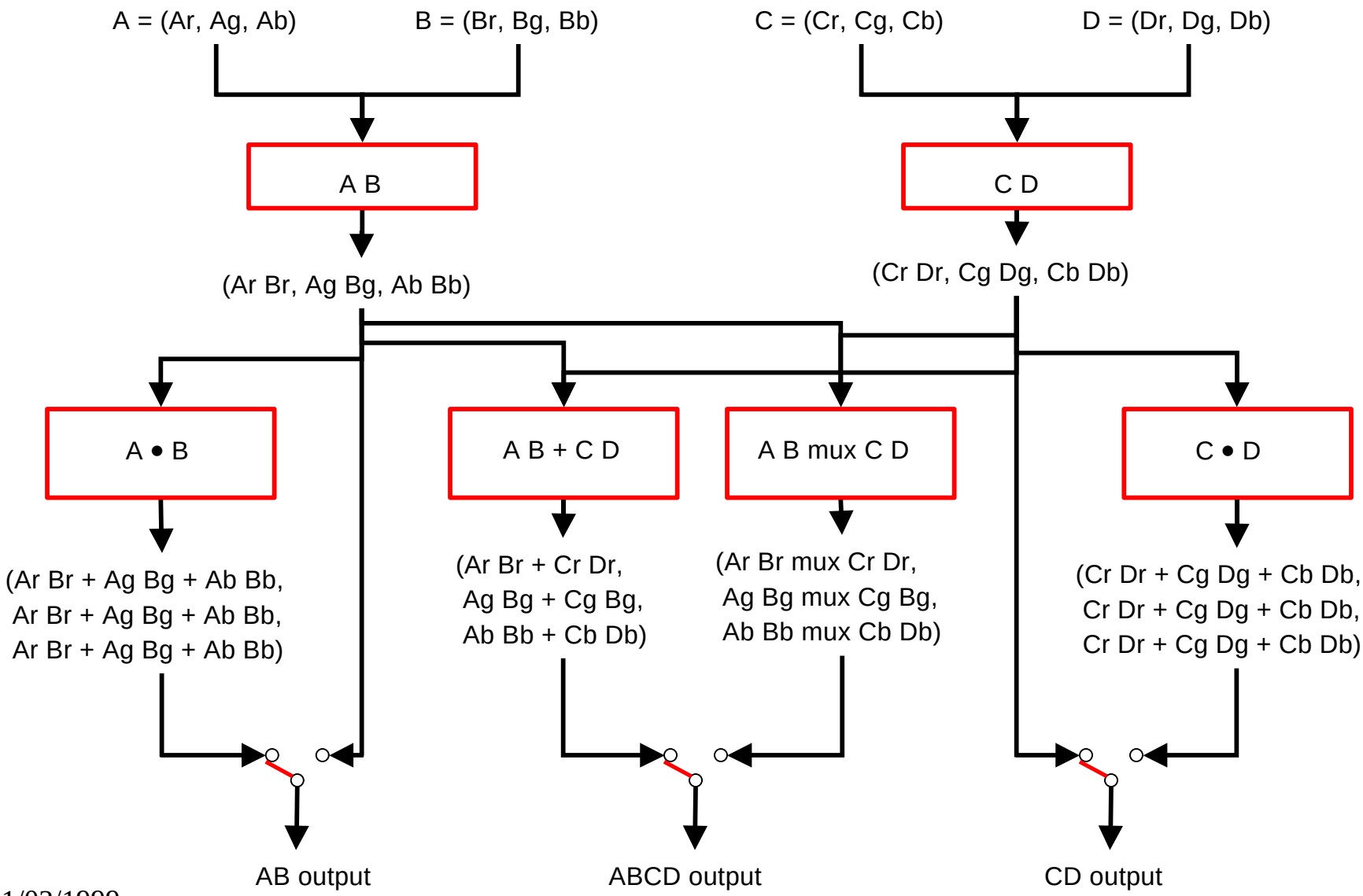
General Combiner Operation, RGB portion

written output registers must be mutually exclusive



NV_register_combiners

General Combiner Stage RGB Computations



NV_register_combiners
General Combiner Mux Computation

the AB mux CD computation is

```
if (texture 0 alpha > 0.5)
   output AB
else
   output CD
```

note that texture 0 alpha is initially undefined if texture unit zero is not enabled

NV_register_combiners
General Combiner Input Mapping Modes

- unsigned identity
  - [0, 1] -> [0, 1]
  - max(0.0, x)
- unsigned invert
  - [0, 1] -> [1, 0]
  - 1.0 - min(max(x, 0.0, 1.0))
- expand normal
  - [0, 1] -> [-1, 1]
  - 2.0 * max(0, x) - 1.0
- expand negate
  - [0, 1] -> [1, -1]
  - -2.0 * max(0.0, x) + 1.0
- half bias normal
  - [0, 1] -> [-0.5, 0.5]
  - max(0.0, x) - 0.5
- half bias negate
  - [0, 1] -> [0.5, -0.5]
  - -max(0.0, x) + 0.5
- signed identity
  - [-1, 1] -> [-1 1]
  - x
- signed negate
  - [-1, 1] -> [1, -1]
  - -x

NV_register_combiners
General Combiner Scale & Bias Modes

- Scale by 1.0, no bias
  x
- Scale by 1.0, bias by -0.5
  x - 0.5
- Scale by 0.5, no bias
  0.5 * x
- Scale by 2.0, no bias
  2.0 * x
- Scale by 2.0, bias by -0.5
  2.0 * x - 0.5
- Scale by 4.0, no bias
  4.0 * x

NV_register_combiners
General Combiner Stage Register Outputs

six outputs per general combiner stage:

three RGB outputs (AB, CD, ABCD)
written to RGB portion of writable registers

three Alpha outputs (AB, CD, ABCD)
written to Alpha portion of writable registers

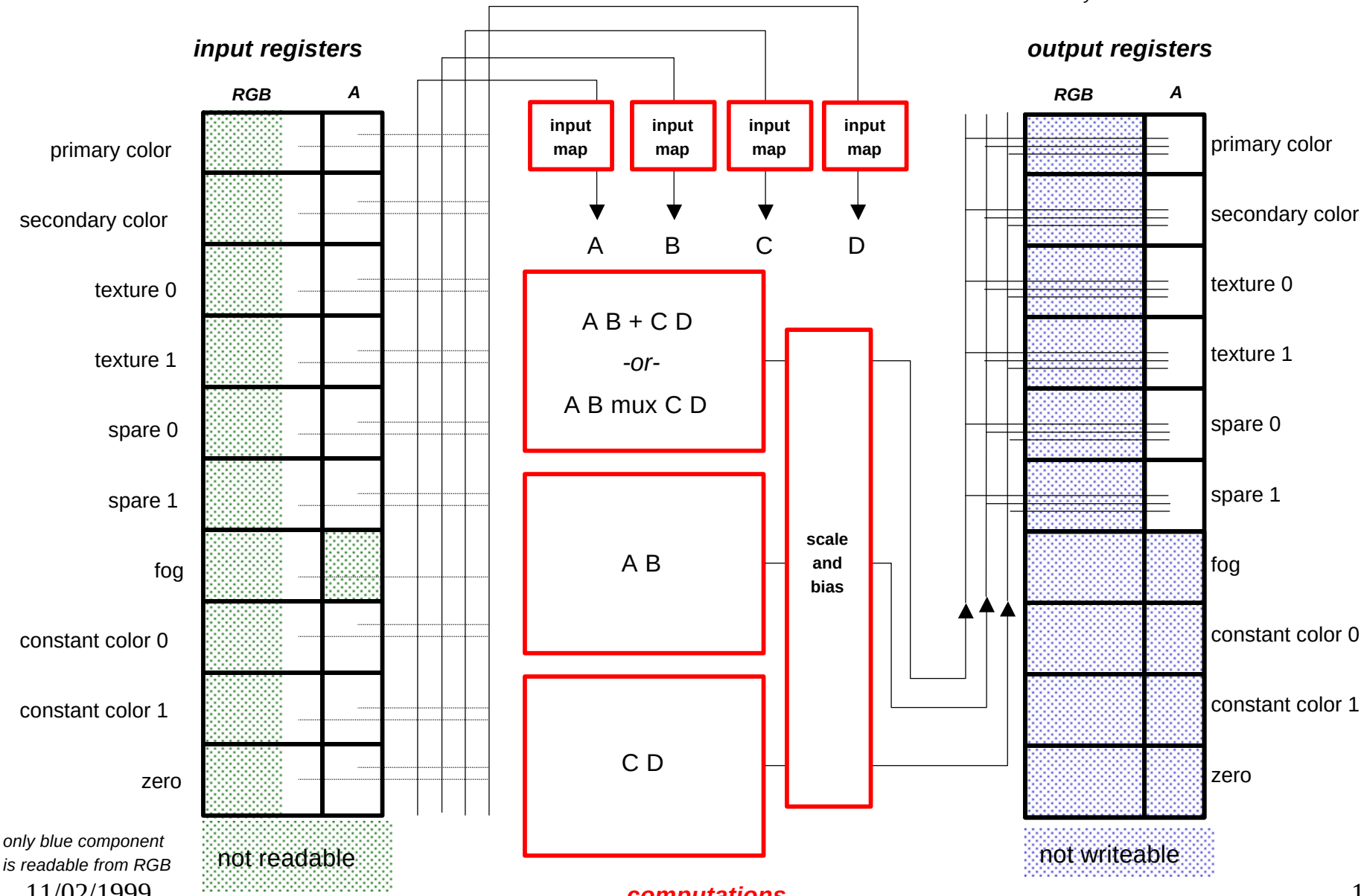
RGB outputs must be written to distinct registers
Alpha outputs must be written to distinct registers
I.e., outputs can not be written to same register portion

if either the RGB AB or CD output of a stage computes a dot product,
the RGB ABCD output for the stage *must* be discarded

any output can be discarded

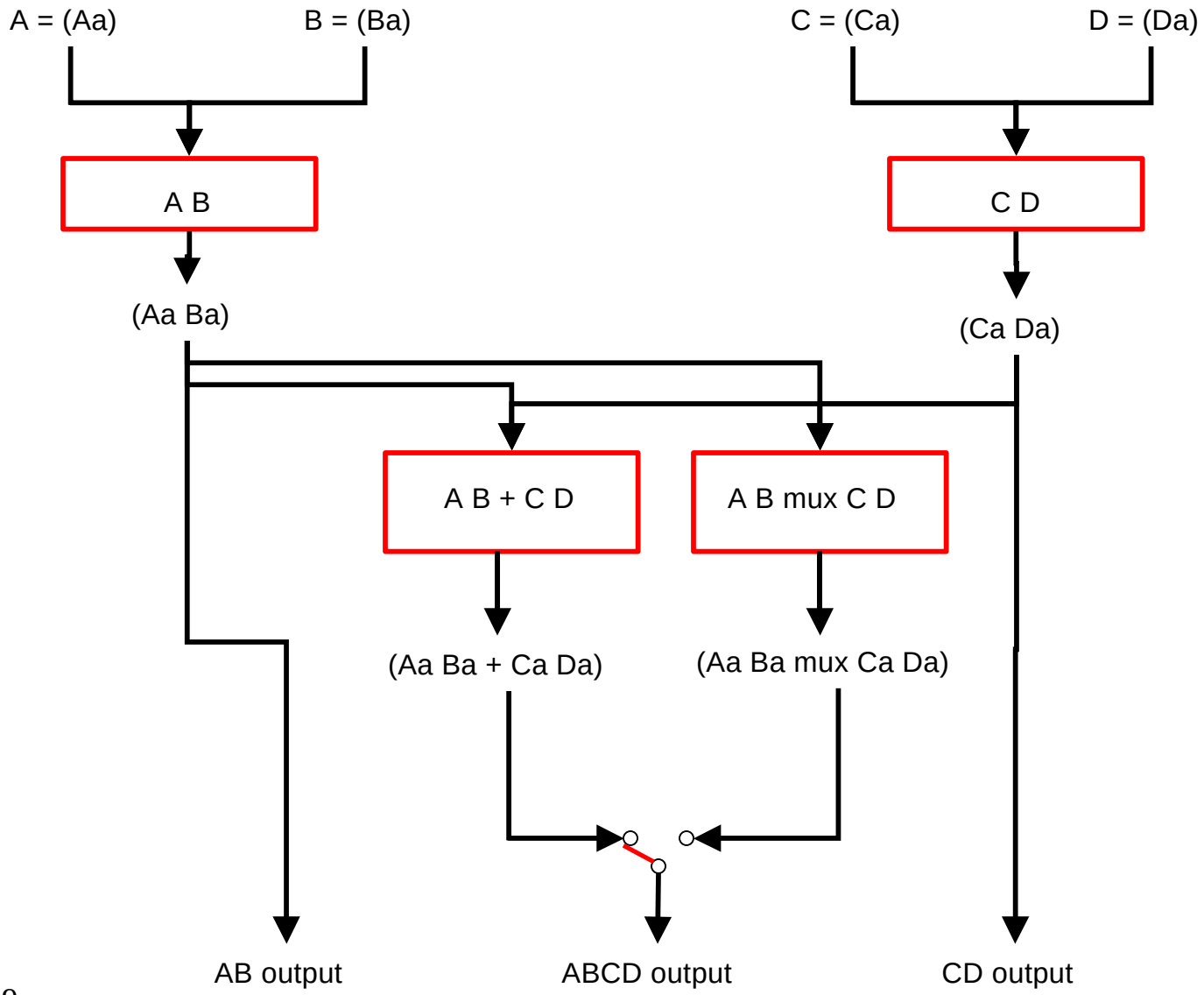
NV_register_combiners

General Combiner Operation, Alpha portion



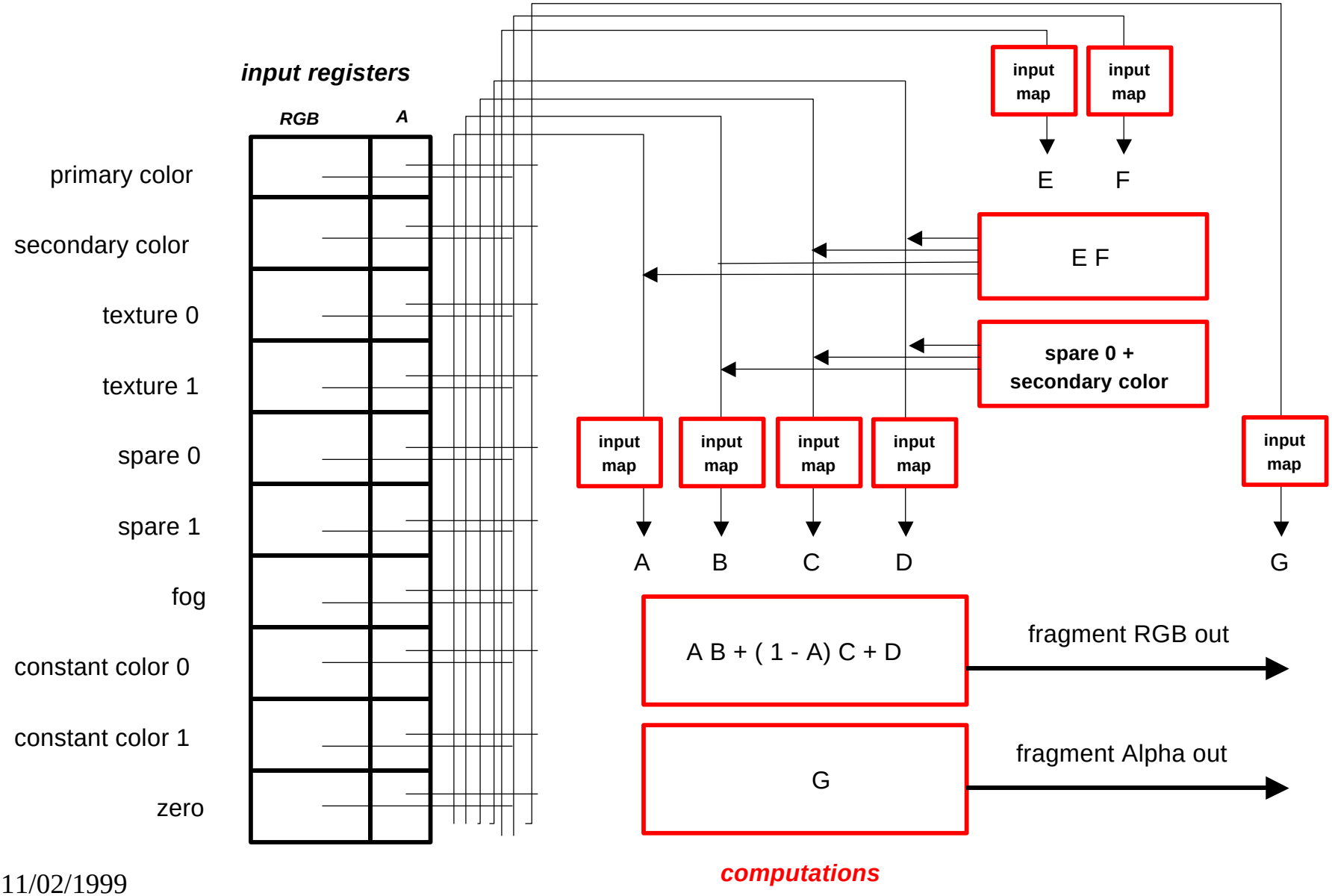
NV_register_combiners

General Combiner Stage Alpha Computations



NV_register_combiners

Final Combiner Operation



NV_register_combiners
Final Combiner Input Mapping Modes

- unsigned identity
  - [0, 1] -> [0, 1]
  - max(0.0, x)

- unsigned invert
  - [0, 1] -> [1, 0]
  - 1.0 - min(max(x, 0.0, 1.0))

NV_register_combiners
Final Combiner Color Sum Clamping

- spare 0 + secondary color clamping options
  - clamp to [0,1] range
  - or sum has [0, 2] range
- A B + (1-A) C + D operates in [0, 4] range

But bug in GeForce and Quadro hardware means useful range is [0, 2].

This bug is fixed in subsequent hardware!

NV_register_combiners
OpenGL API
Register Combiners Enable/Disable

Enable register combiners

```
glEnable(GL_REGISTER_COMBINERS_NV);
```

Disable register combiners

```
glDisable(GL_REGISTER_COMBINERS_NV);
```

NV_register_combiners
OpenGL API
General Combiner Input/Output Control

glCombinerInputNV(**GLenum stage,**
GLenum portion,
GLenum variable,
GLenum input,
GLenum mapping,
GLenum componentUsage);

glCombinerOutputNV(**GLenum stage,**
GLenum portion,
GLenum abOutput,
GLenum cdOutput,
GLenum sumOutput,
GLenum scale,
GLenum bias,
GLboolean abDotProduct,
GLboolean cdDotProduct,
GLboolean muxSum);

Red indicates specifiers
Blue indicates state values

NV_register_combiners
OpenGL API
General Combiner Input/Output Control Tokens

```
stage =                 { GL_COMBINER0_NV, GL_COMBINER1_NV }
portion =               { GL_RGB, GL_ALPHA }
variable =              { GL_VARIABLE_A_NV, GL_VARIABLE_B_NV,
                        GL_VARIABLE_C_NV, GL_VARIABLE_D_NV }
input =                 { GL_ZERO, GL_PRIMARY_COLOR_NV, GL_SECONDARY_COLOR_NV,
                          GL_CONSTANT_COLOR0_NV, GL_CONSTANT_COLOR1_NV,
                          GL_TEXTURE0_ARB, GL_TEXTURE1_ARB, GL_FOG,
                          GL_SPARE0_NV, GL_SPARE1_NV }
mapping =   { GL_UNSIGNED_IDENTITY_NV, GL_UNSIGNED_INVERT_NV,
                          GL_EXPAND_NORMAL_NV, GL_EXPAND_NEGATE_NV,
                          GL_HALF_BIAS_NORMAL_NV, GL_HALF_BIAS_NEGATE_NV,
                          GL_SIGNED_IDENTITY_NV, GL_SIGNED_NEGATE_NV }
componentUsage =        { GL_RGB, GL_BLUE, GL_ALPHA }
abOutput, cdOutput,
sumOutput =             { GL_DISCARD_NV,
                          GL_PRIMARY_COLOR_NV, GL_SECONDARY_COLOR_NV,
                          GL_TEXTURE0_ARB, GL_TEXTURE1_ARB,
                          GL_SPARE0_NV, GL_SPARE1_NV }
scale =                 { GL_NONE, GL_SCALE_BY_TWO_NV, GL_SCALE_BY_FOUR_NV,
                          GL_SCALE_BY_ONE_HALF_NV }
bias =                  { GL_NONE, GL_BIAS_BY_NEGATIVE_ONE_HALF_NV }
```

NV_register_combiners
OpenGL API
Final Combiner Input Control

glFinalCombinerInputNV(**GLenum variable,**
GLenum input,
GLenum mapping,
GLenum componentUsage);

Red indicates specifiers
Blue indicates state values

NV_register_combiners
OpenGL API
Final Combiner Input Control Tokens

```
variable =              { GL_VARIABLE_A_NV, GL_VARIABLE_B_NV,
                        GL_VARIABLE_C_NV, GL_VARIABLE_D_NV,
                          GL_VARIABLE_E_NV, GL_VARIABLE_F_NV,
                          GL_VARIABLE_G_NV  }
input =                 { GL_ZERO, GL_PRIMARY_COLOR_NV, GL_SECONDARY_COLOR_NV,
                            GL_CONSTANT_COLOR0_NV, GL_CONSTANT_COLOR1_NV,
                            GL_TEXTURE0_ARB, GL_TEXTURE1_ARB, GL_FOG,
                            GL_SPARE0_NV, GL_SPARE1_NV,
                            GL_E_TIMES_F_NV, GL_SPARE0_PLUS_SECONDARY_COLOR_NV  }
mapping =   { GL_UNSIGNED_IDENTITY_NV, GL_UNSIGNED_INVERT_NV }
componentUsage =        { GL_RGB, GL_BLUE, GL_ALPHA }
```

NV_register_combiners
OpenGL API
Combiner Parameter Control

```
glCombinerParameterfvNV(GLenum pnamev,
                        const GLfloat *params);

glCombinerParameterivNV(GLenum pname,
                        const GLint *params);

glCombinerParameterfNV(GLenum pname,
                       GLfloat param);

glCombinerParameteriNV(GLenum pnamev,
                       GLint param);
```

NV_register_combiners
OpenGL API
Combiner Parameter Control Tokens

pname = **{ GL_NUM_COMBINERS_NV, GL_COLOR_SUM_CLAMP_NV }**

pnamev = **{ GL_NUM_COMBINERS_NV, GL_COLOR_SUM_CLAMP_NV,**
GL_CONSTANT_COLOR0_NV, GL_CONSTANT_COLOR1_NV }

NV_register_combiners OpenGL API Combiner State Queries

```
glGetCombinerInputParameterfvNV(GLenum stage,
                                GLenum portion,
                                GLenum variable,
                                GLenum pname_i,
                                GLfloat *params);

glGetCombinerInputParameterivNV(GLenum stage,
                                GLenum portion,
                                GLenum variable,
                                GLenum pname_i,
                                GLint *params);

glGetCombinerOutputParameterfvNV(GLenum stage,
                                 GLenum portion,
                                 GLenum pname_o,
                                 GLfloat *params);

glGetCombinerOutputParameterivNV(GLenum stage,
                                 GLenum portion,
                                 GLenum pname_o,
                                 GLint *params);

glGetFinalCombinerInputParameterfvNV(GLenum fvariable,
                                     GLenum pname_i,
                                     GLfloat *params);

glGetFinalCombinerInputParameterivNV(GLenum fvariable,
                                     GLenum pname_i,
                                     GLfloat *params);
```

Red indicates specifiers

NV_register_combiners
OpenGL API
Combiner State Queries Tokens

stage = { GL_COMBINER0_NV, GL_COMBINER1_NV }
portion = { GL_RGB, GL_ALPHA }
variable = { GL_VARIABLE_A_NV, GL_VARIABLE_B_NV,
GL_VARIABLE_C_NV, GL_VARIABLE_D_NV }
fvariable = { GL_VARIABLE_A_NV, GL_VARIABLE_B_NV,
GL_VARIABLE_C_NV, GL_VARIABLE_D_NV,
GL_VARIABLE_E_NV, GL_VARIABLE_F_NV,
GL_VARIABLE_G_NV }

pname_i = { GL_COMBINER_INPUT_NV, GL_COMBINER_MAPPING_NV,
GL_COMPONENT_USAGE_NV }

pname_o = { GL_COMBINER_AB_DOT_PRODUCT_NV,
GL_COMBINER_CD_DOT_PRODUCT_NV,
GL_COMBINER_MUX_SUM_NV,
GL_COMBINER_SCALE_NV,
GL_COMBINER_BIAS_NV,
GL_COMBINER_AB_OUTPUT_NV,
GL_COMBINER_CD_OUTPUT_NV,
GL_COMBINER_SUM_OUTPUT_NV }