

```
// Glue code for next generation Gryphon SVBC
//
// Address map
// Device Base Address Size HEX
// -----
// ROM FFF8-0000 52428 80000
// SRAM FFF0-0000 52428 80000
// SERPORT FFEF-0000 256 100
// PARPORT FFEF-0100 256 100
// FDC FFEF-0200 256 100
// RTC FFEF-0300 256 100
// SND FFEF-0400 256 100
// KEYBD FFEF-0500 256 100
// VIDREG FFEF-0600 256 100
// IDE FFEF-0700 256 100
// ETHNET FFEF-1000 65536 10000
// VIDRAM FFEF-2000 2097152 200000
// DRAM 0000-0000
```

```
module xagdin (
    output wire csrom,
    output wire csram,
    output wire csenet,
    output wire csdram,
    output wire csser,
    output wire cspar,
    output wire csfdc,
    output wire csrtc,
    output wire csvidr,
    output wire cside,
    output wire cskeyb,
    output wire cscopro,
    output wire ds0,
    output wire ds1,

    output wire ioen,

    output wire cs_video,
    output wire int_keyb,
    output wire bcl,
    output wire bdir,

    output wire avec,

    input wire reset,
```

```
input wire [31:8] addrbus_h,
input wire a0,
input wire rw,
input wire as_n,
input wire fc0,
input wire fc1,
input wire fc2,

input wire delay160,
input wire delay240,
input wire delay640,
input wire dtack_net,
input wire dtack_ser,
input wire dtack_par,
input wire dsack0_cop,

input wire dsack1_vid,
input wire dsack1_cop,

input wire p24,
input wire p25,
input wire ipl0,
input wire ipl1,
input wire ipl2
);
```

```
localparam ROM      = 16'hFFF8;
localparam SRAM      = 16'hFFF0;
localparam SERPORT   = 24'hFFEF00;
localparam PARPORT   = 24'hFFEF01;
localparam FDC       = 24'hFFEF02;
localparam RTC       = 24'hFFEF03;
localparam SND       = 24'hFFEF04;
localparam KEYBD     = 24'hFFEF05;
localparam VIDREG    = 24'hFFEF06;
localparam IDE       = 24'hFFEF07;
localparam ETHNET    = 16'hFFEE;
localparam VIDRAM    = 12'hFFD;
localparam DRAM      = 8'h00;
localparam COPROC_ID = 7'b0010001;
```

```
wire cpuspc;
```

```
wire da;
wire db;
wire cssnd;
wire csvram;
```

```
assign cpushpc = fc2 & fc1 & fc0;
```

```
reg boot;
```

```
/* Address decocoding */
```

```
assign csrom = boot ? as_n : ~(~as_n & ~cpuspc & (addrbus_h[31:16] == ROM));
assign csram = boot ? 1'h1 : ~(~as_n & ~cpuspc & (addrbus_h[31:16] == SRAM));
assign csetnet = ~(~as_n & ~cpuspc & (addrbus_h[31:16] == ETHNET));
assign csvram = ~(~as_n & ~cpuspc & (addrbus_h[31:20] == VIDRAM));
assign csdram = ~(~as_n & ~cpuspc & (addrbus_h[31:24] == DRAM));
assign csser = ~(~as_n & ~cpuspc & (addrbus_h[31:8] == SERPORT));
assign cspar = ~(~as_n & ~cpuspc & (addrbus_h[31:8] == PARPORT));
assign csfdc = ~(~as_n & ~cpuspc & (addrbus_h[31:8] == FDC));
assign csrtc = ~(~as_n & ~cpuspc & (addrbus_h[31:8] == RTC));
assign csvidr = ~(~as_n & ~cpuspc & (addrbus_h[31:8] == VIDRAM));
assign cssnd = ~(~as_n & ~cpuspc & (addrbus_h[31:8] == SND));
assign csidr = ~(~as_n & ~cpuspc & (addrbus_h[31:8] == IDE));
assign cskeyb = ~(~as_n & ~cpuspc & (addrbus_h[31:8] == KEYBD));
```

```
/* DS0 generation */
```

```
assign da = (csrom & delay160) | (csram & delay160) | (csfdc & delay240) | (csrtc & delay640) | (cssnd & delay640) | (cskeyb & delay640) | ~dtack_net;
assign db = ~dtack_ser | ~dtack_par | ~dsack0_cop;
```

```
assign ds0 = da | db;
```

```
/* DS1 generation */
```

```
assign ds1 = ~dsack1_vid | ~dsack1_cop | ~(csidr & delay160);
```

```
/* Byte Decode logic */
```

```
assign ioen = ~(csrom & csram & csetnet & csvram & csdram & csvidr);
```

```
/* Misc decode from U19 */
```

```
assign cs_video = ~csvram | ~csvidr;
assign int_keyb = ~(p24 | p25);
assign bdir = ~(cssnd & ~rw);
assign bcl = ~(cssnd & a0);
```

```
/*  Coprocessor decode  */
```

```
assign cscopro = ~(fc2 & fc1 & fc0 & (addrbus_h[19:13] == COPROC_ID));
```

```
/* Autovector Interrupts */
```

```
assign avec = ~(cpuspc * ~as_n & ~ipl0 | cpuspc & ~as_n & ~ipl1 | cpuspc & ~as_n & ~ipl2);
```

```
/* Handle the boot trampoline */
```

```
reg [1:0] boot_cnt;
```

```
always @(posedge as_n or negedge reset) begin
```

```
    if (!reset) begin
```

```
        boot <= 1;
```

```
        boot_cnt <= 2'h0;
```

```
    end else begin
```

```
        if (boot_cnt == 2'h3) begin
```

```
            boot <= 0;
```

```
        end else begin
```

```
            boot_cnt <= boot_cnt + 2'h1;
```

```
        end
```

```
    end
```

```
end
```

```
endmodule
```