Bridge of Life Education

Caravel SOC Introduction

Interrupt SRAM Wishbone UserProject Testbench Firmware

Josh

Topics

- System Block Diagram (modules)
- Processor VexRISCV functions & its interface
- Reset / POR
- Management Protect Are – power control
- Clocking / DLL, configuration SPI register
- Housekeeping SPI registers
- GPIO
- SPI
- Interrupt (IRQ)
- Memory-mapped IO address
- SRAM
- Bus - Wishbone
- Peripherals – Counter/Timer/UART
- User project design (counter) Interface
- Testbench
- Firmware code

Bridge of Life Education

Caravel SOC – Interrupt (IRQ)

Josh

Outline

- Introduction
- EventManagmet-Base Interrupt System
- IRQ from User Project
  - Interrupt Enabling
  - Interrupt Service Routine

Introduction

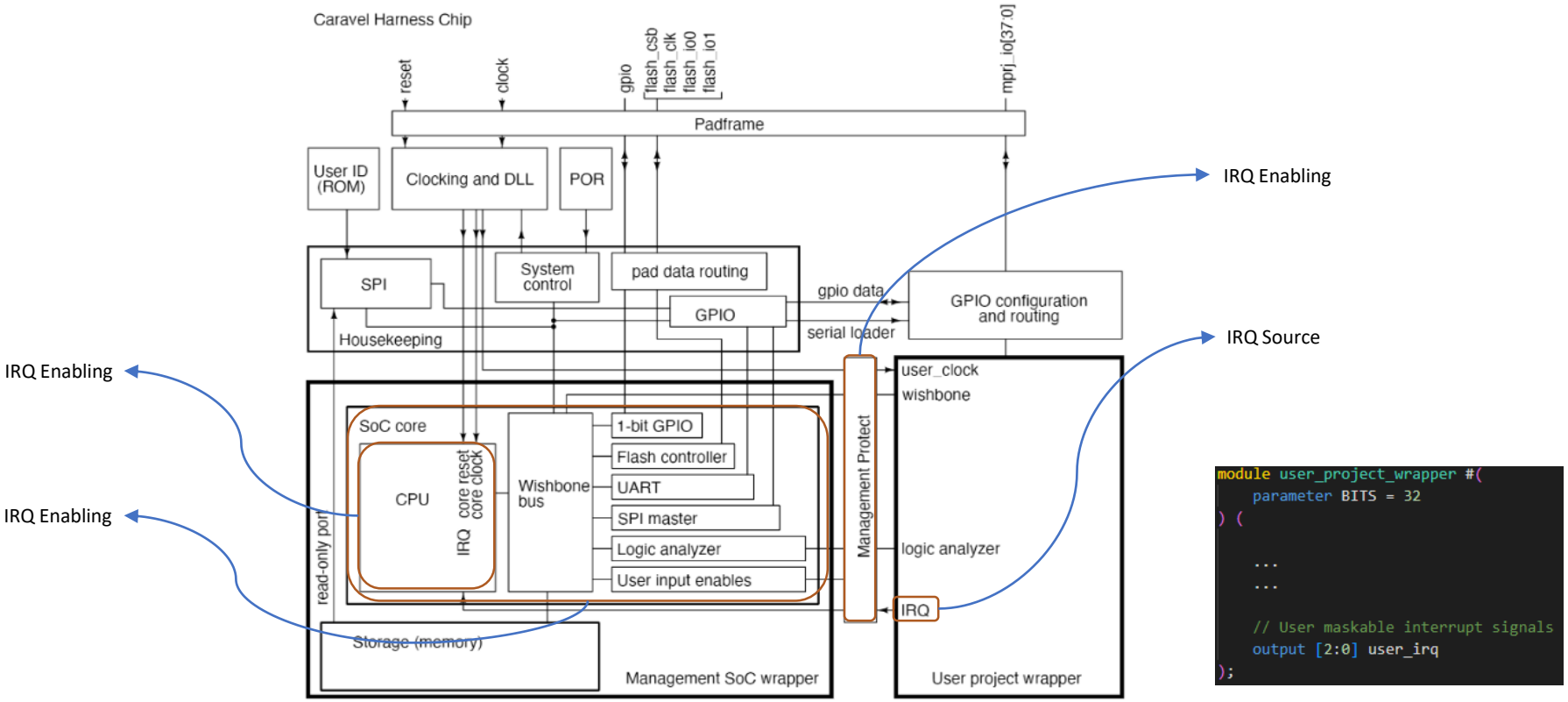
- Caravel Management SoC supports multiple types of interrupt
  - Timer
  - UART
  - User Project
- Generic usage model
  - Setup the interrupt service routine (ISR)
    - Checking corresponding interrupt status register
    - Clearing status register in the routine
    - May need to mask interrupt and unmask interrupt if nested interrupt not allowed
  - Enabling corresponding interrupt enable control registers
    - The interrupt register enabling must included the whole interrupt delivery path

EventManagmet-Base Interrupt System

- Interrupt Event is represented and controlled by registers
  - Interrupt Status, Mode, and Enabling

Interrupt	Module
0	TIMER0
1	UART
2	USER_IRQ_0
3	USER_IRQ_1
4	USER_IRQ_2
5	USER_IRQ_3
6	USER_IRQ_4
7	USER_IRQ_5

IRQ from User Project




```
module user_project_wrapper #(
    parameter BITS = 32
) (

    ...
    ...

    // User maskable interrupt signals
    output [2:0] user_irq
);
```

https://github.com/bol-edu/caravel-soc

IRQ from User Project

- Enabling User Project USER_IRQ_0
  - EventManger-based interrupt Control/Status Register

Register	Address	Description
USER_IRQ_0_IN	0xF0006800	IRQ Input(s) Status.
USER_IRQ_0_MODE	0xF0006804	IRQ Mode: 0: Edge, 1: Change
USER_IRQ_0_EDGE	0xF0006808	IRQ Edge (when in Edge mode): 0: Rising Edge, 1: Falling Edge
USER_IRQ_0_EV_STATUS	0xF000680C	This register contains the current raw level of the i0 event trigger. Writes to this register have no effect [0]: Level of i0 event
USER_IRQ_0_EV_PENDING	0xF0006810	When an i0 event occurs, the corresponding bit will be set in this register. To clear the Event, set the corresponding bit in this register [0]: 1 if i0 event occurred
USER_IRQ_0_EV_ENABLE	0xF0006814	This register enables the corresponding i0 events. Write a 0 to this register to disable individual events [0]: Write a 1 to enable i0 Event
USER_IRQ_ENA_OUT	0xf0009800	User IRQ Enables. Which is the bitmap to user_irq[2:0]. Configuring this register can be skipped in case the Management Protect isn't exist.

https://github.com/JoshSu0/caravel-soc_fpga

IRQ from User Project

- Enabling User Project USER_IRQ_0
  - CSR IRQ Mask Register
    - Interrupt Bitmaps
      - CSR_IRQ_MASK
        - 1’b1: unmask
      - CSR_IRQ_PENDING
        - 1’b1: IRQ Pending
  - Application interface definition

```
#define CSR_IRQ_MASK 0xBC0
#define CSR_IRQ_PENDING 0xFC0
```

```
#define TIMER0_INTERRUPT 0
#define UART_INTERRUPT 1
#define USER_IRQ_0_INTERRUPT 2
#define USER_IRQ_1_INTERRUPT 3
#define USER_IRQ_2_INTERRUPT 4
#define USER_IRQ_3_INTERRUPT 5
#define USER_IRQ_4_INTERRUPT 6
#define USER_IRQ_5_INTERRUPT 7
```

```
static inline unsigned int irq_getmask(void)
static inline void irq_setmask(unsigned int mask)
static inline unsigned int irq_pending(void)
```

https://github.com/JoshSu0/caravel-soc_fpga

IRQ from User Project

- Enabling User Project USER_IRQ_0
  - RISC-V Machine Interrupt and Status Register
    - Machine Interrupt-Enable register, mie
      - Bits 11, MEIE (Machine External Interrupt Enable)
        - USER_IRQ_x interrupts described previously are mapped to the external interrupt type

15 … 12	11	10	9	8	7	6	5	4	3	2	1	0
0	MEIE	0	SEIE	0	MTIE	0	STIE	0	MSIE	0	SSIE	0
4	1	1	1	1	1	1	1	1	1	1	1	1

    - Machine-mode Status register, mstatus
      - Bits 3, MIE (Machine Interrupt Enable)
        - A global interrupt enable bit of RISC-V

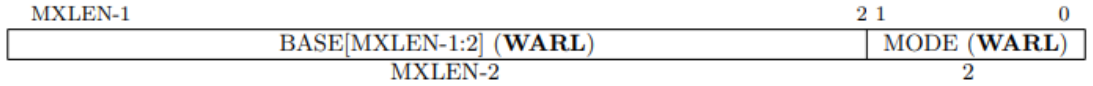
31	30 … 23	22	21	20	19	18	17
SD	WPRI	TSR	TW	TVM	MXR	SUM	MPRV
1	8	1	1	1	1	1	1

16 15	14 13	12 11	10 9	8	7	6	5	4 … 4	3	2 … 2	1	0 … 0
XS[1:0]	FS[1:0]	MPP[1:0]	VS[1:0]	SPP	MPIE	UBE	SPIE	WPRI	MIE	WPRI	SIE	WPRI
2	2	2	2	1	1	1	1	1	1	1	1	1

https://github.com/riscv/riscv-isa-manual/releases/download/Priv-v1.12/riscv-privileged-20211203.pdf

IRQ from User Project

- ISR Setup User Project USER_IRQ_0
  - RSIC-V Machine Trap-Vector Base-Address Register, mtvec
    - Hooked the ISR



Value	Name	Description
0	Direct	All exceptions set pc to BASE.
1	Vectored	Asynchronous interrupts set pc to BASE+4×cause
>= 2	NA	Reserved

https://github.com/JoshSu0/caravel-soc_fpga

IRQ from User Project

- ISR Setup User Project USER_IRQ_0
  - Hook is setup at C Runtime Initialization file
    - crt0_vex.s
  - ISR is defined in isr.c

```
void isr(void)
{
    uint32_t irqs = irq_pending() & irq_getmask();

    if ( irqs & (1 << USER_IRQ_0_INTERRUPT)) {
        user_irq_0_ev_pending_write(1); //Clear Interrupt Pending Event
    }

    return;
}
```

```
.global isr
.global _start

_start:
  j crt_init
  nop
  nop
  nop
  nop
  nop
  nop
  nop

.global  trap_entry
trap_entry:
  sw x1,  - 1*4(sp)
  ...
  ...
  sw x31, -16*4(sp)
  addi sp,sp,-16*4
  call isr
  lw x1 , 15*4(sp)
  ...
  ...
  lw x31,  0*4(sp)
  addi sp,sp,16*4
  mret
```

```
crt_init:
  la sp, _fstack
  la a0, trap_entry
  csrw mtvec, a0
```

https://github.com/JoshSu0/caravel-soc_fpga

Bridge of Life Education

Caravel SOC – SRAM

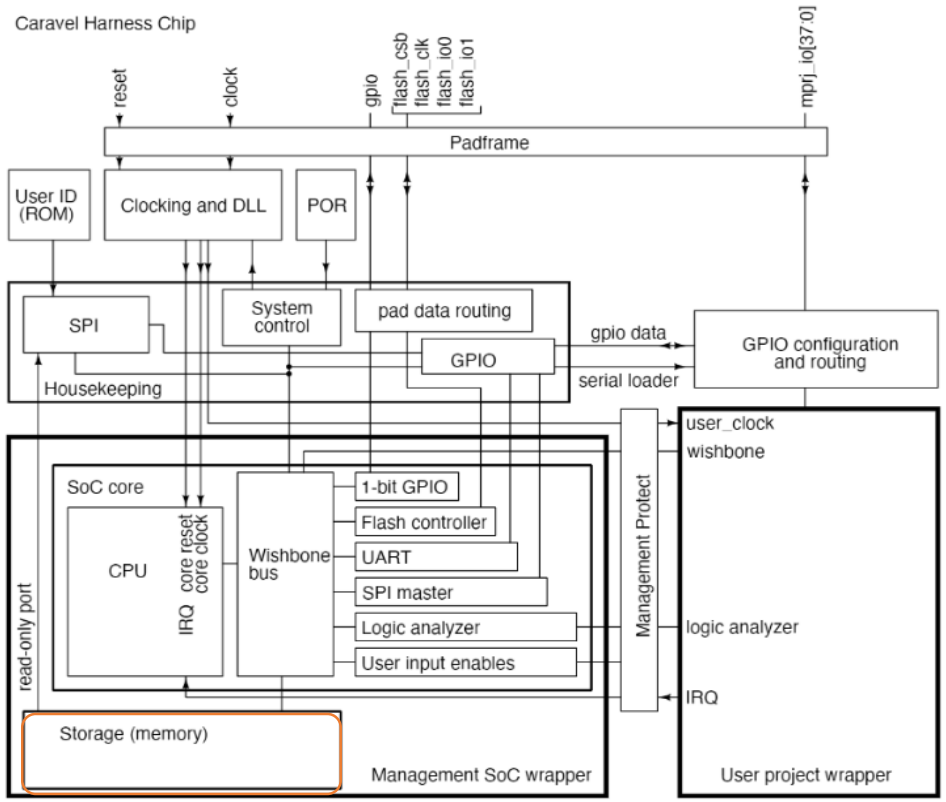
Josh

Outline

- Introduction
- Memory description
  - Executable Data Initialization
- SRAM Implementation

SRAM

- Storage (memory)
  - Connected through the Wishbone Interface



https://github.com/bol-edu/caravel-soc

Memory description

- Link Descriptor
  - sections.lds
- SRAM Usage
  - dff, Executable Data Section
    - .data
      - Initialized global variables
    - .bss
      - Uninitialized global variables
  - dff2, Stack for C Runtime

```
MEMORY {
    vexriscv_debug : ORIGIN = 0xf00f0000, LENGTH = 0x00000100
    dff : ORIGIN = 0x00000000, LENGTH = 0x00000400
    dff2 : ORIGIN = 0x00000400, LENGTH = 0x00000200
    flash : ORIGIN = 0x10000000, LENGTH = 0x01000000
    mprj : ORIGIN = 0x30000000, LENGTH = 0x00100000
    hk : ORIGIN = 0x26000000, LENGTH = 0x00100000
    csr : ORIGIN = 0xf0000000, LENGTH = 0x00010000
}
```

```
.data :
{
    . = ALIGN(8);
    _fdata = .;
    *(.data .data.* .gnu.linkonce.d.*)
    *(.data1)
    _gp = ALIGN(16);
    *(.sdata .sdata.* .gnu.linkonce.s.*)
    . = ALIGN(8);
    _edata = .;
} > dff AT > flash
```

```
.bss :
{
    . = ALIGN(8);
    _fbss = .;
    *(.dynsbss)
    *(.sbss .sbss.* .gnu.linkonce.sb.*)
    *(.scommon)
    *(.dynbss)
    *(.bss .bss.* .gnu.linkonce.b.*)
    *(COMMON)
    . = ALIGN(8);
    _ebss = .;
    _end = .;
} > dff
```

Executable Data Initialization

- C Runtime Initialization
  - Copy Data section from ROM to RAM

```
data_init:
  la a0, _fdata
  la a1, _edata
  la a2, _fdata_rom
data_loop:
  beq a0,a1,data_done
  lw a3,0(a2)
  sw a3,0(a0)
  add a0,a0,4
  add a2,a2,4
  j data_loop
data_done:

bss_init:
  la a0, _fbss
  la a1, _ebss
bss_loop:
  beq a0,a1,bss_done
  sw zero,0(a0)
  add a0,a0,4
#ifndef SIM
  j bss_loop
#endif
bss_done:
```

SRAM Implementation

- Instances at mgmt._core.v
  - dff
    - 1024 bytes
    - RAM256.v
  - dff2
    - 512 bytes
    - RAM128.v

```
RAM256 RAM256(
    .A0(dff_bus_adr[7:0]),
    .CLK(sys_clk),
    .Di0(dff_di),
    .EN0(dff_en),
    .WE0(dff_we),
    .Do0(dff_do)
);

RAM128 RAM128(
    .A0(dff2_bus_adr[6:0]),
    .CLK(sys_clk),
    .Di0(dff2_di),
    .EN0(dff2_en),
    .WE0(dff2_we),
    .Do0(dff2_do)
);
```

```
module RAM128 #( parameter COLS=1)
(
`ifdef USE_POWER_PINS
    VPWR,
    VGND,
`endif
    CLK,
    WE0,
    EN0,
    Di0,
    Do0,
    A0
);
    localparam A_WIDTH = 7+$clog2(COLS);

    input   wire            VPWR;
    input   wire            VGND;
    input   wire            CLK;
    input   wire    [3:0]   WE0;
    input   wire            EN0;
    input   wire    [31:0]  Di0;
    output  reg     [31:0]  Do0;
    input   wire    [(A_WIDTH - 1): 0]   A0;

    reg [31:0] RAM[(256*COLS)-1 : 0];

    always @(posedge CLK)
        if(EN0) begin
            Do0 <= RAM[A0];
            if(WE0[0]) RAM[A0][ 7: 0] <= Di0[7:0];
            if(WE0[1]) RAM[A0][15:8] <= Di0[15:8];
            if(WE0[2]) RAM[A0][23:16] <= Di0[23:16];
            if(WE0[3]) RAM[A0][31:24] <= Di0[31:24];
        end
        else
            Do0 <= 32'b0;

endmodule
```

https://github.com/bol-edu/caravel-soc

Bridge of Life Education

Caravel SOC – Wishbone

Josh

Outline

- WISHBONE Introduction
- WISHBONE in Caravel
  - Path to User Project Slave
    - Request Selection
    - Slave Selection

Introduction

- System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores
- Purpose - Fostering design reuse by alleviating SoC integration problems

Introduction

- Variable core interconnection methods
  - Point-to-point
  - Data flow
  - Shared bus
    - Caravel SoC applied
  - Crossbar switch

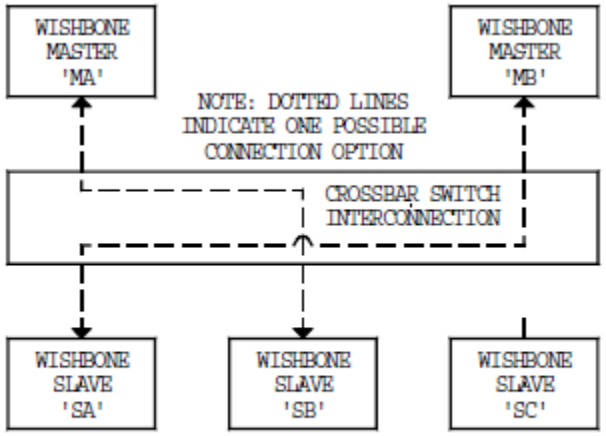


Figure A-5. Crossbar switch interconnection.

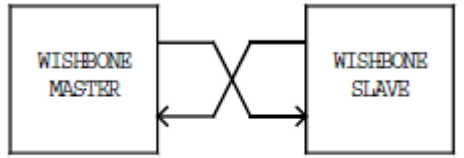


Figure A-2. The point-to-point interconnection.

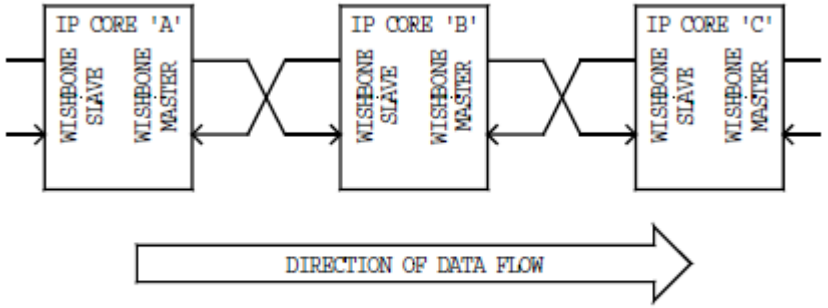


Figure A-3. The data flow interconnection.

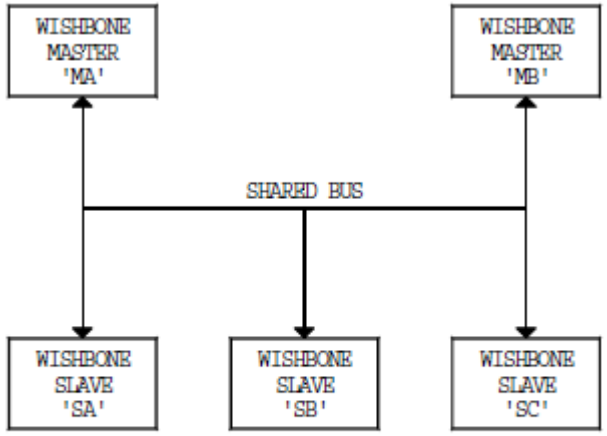


Figure A-4. Shared bus interconnection.

Introduction

- WISHBONE Signals
  - SYSCON
    - RST_O
    - CLK_O
  - Common for both MASTER/SLAVE
    - RST_I
    - CLK_I
    - DAT_I(), DAT_O()
    - TGD_I(), TGD_O()
  - MASTER
    - ADR_O(), WE_O, SEL_O(), STB_O, ACK_I, CYC_O
    - ERR_I, RTY_I, LOCK_O, TGA_O(), TGC_O(), CTI_O(), BTE_O()
  - SLAVE
    - ADR_I(), WE_I, SEL_I(), STB_I, ACK_O, CYC_I
    - ERR_O, RTY_O, LOCK_I, TGA_I(), TGC_I(), CTI_I(), BTE_I()

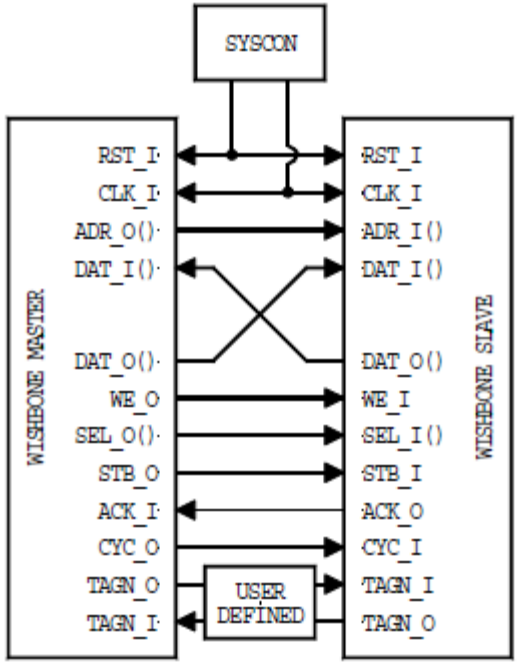
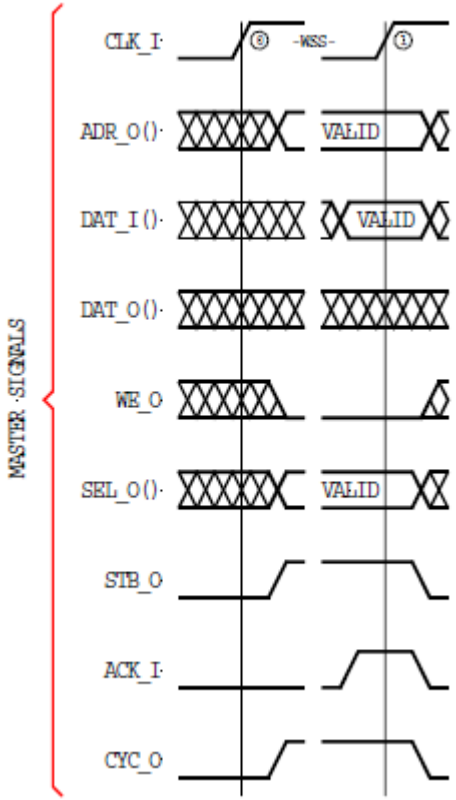
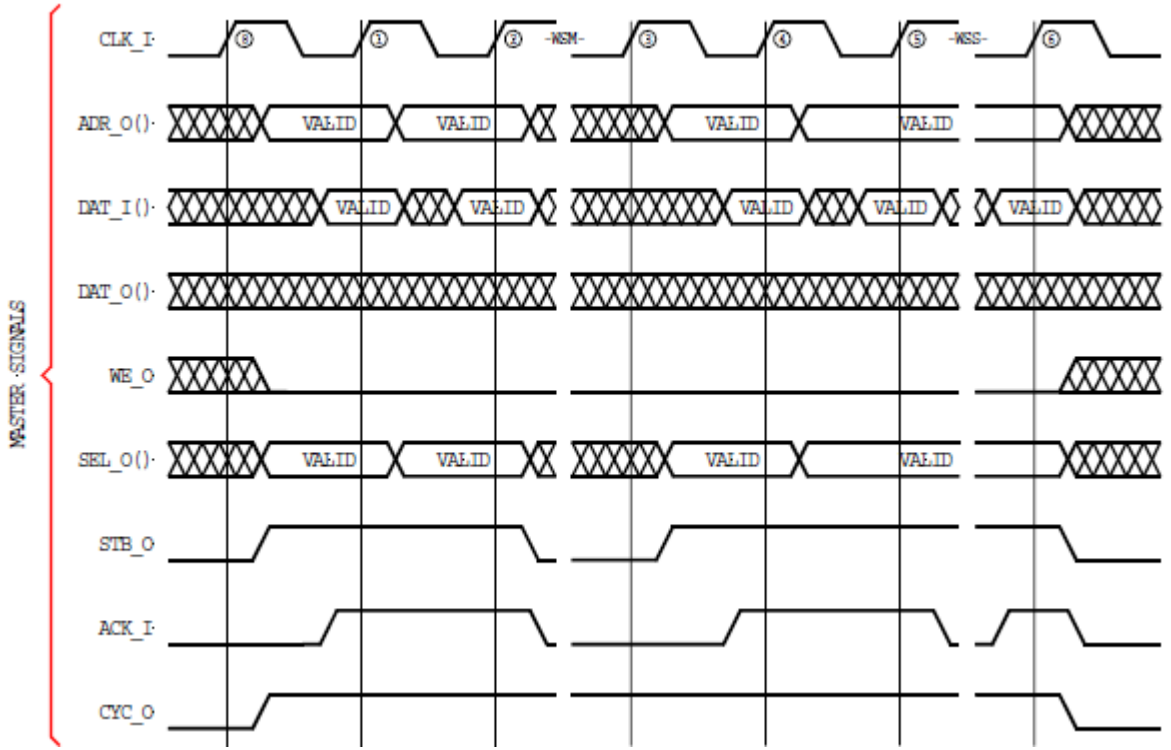


Figure 1-2. Standard connection for timing diagrams.

Introduction

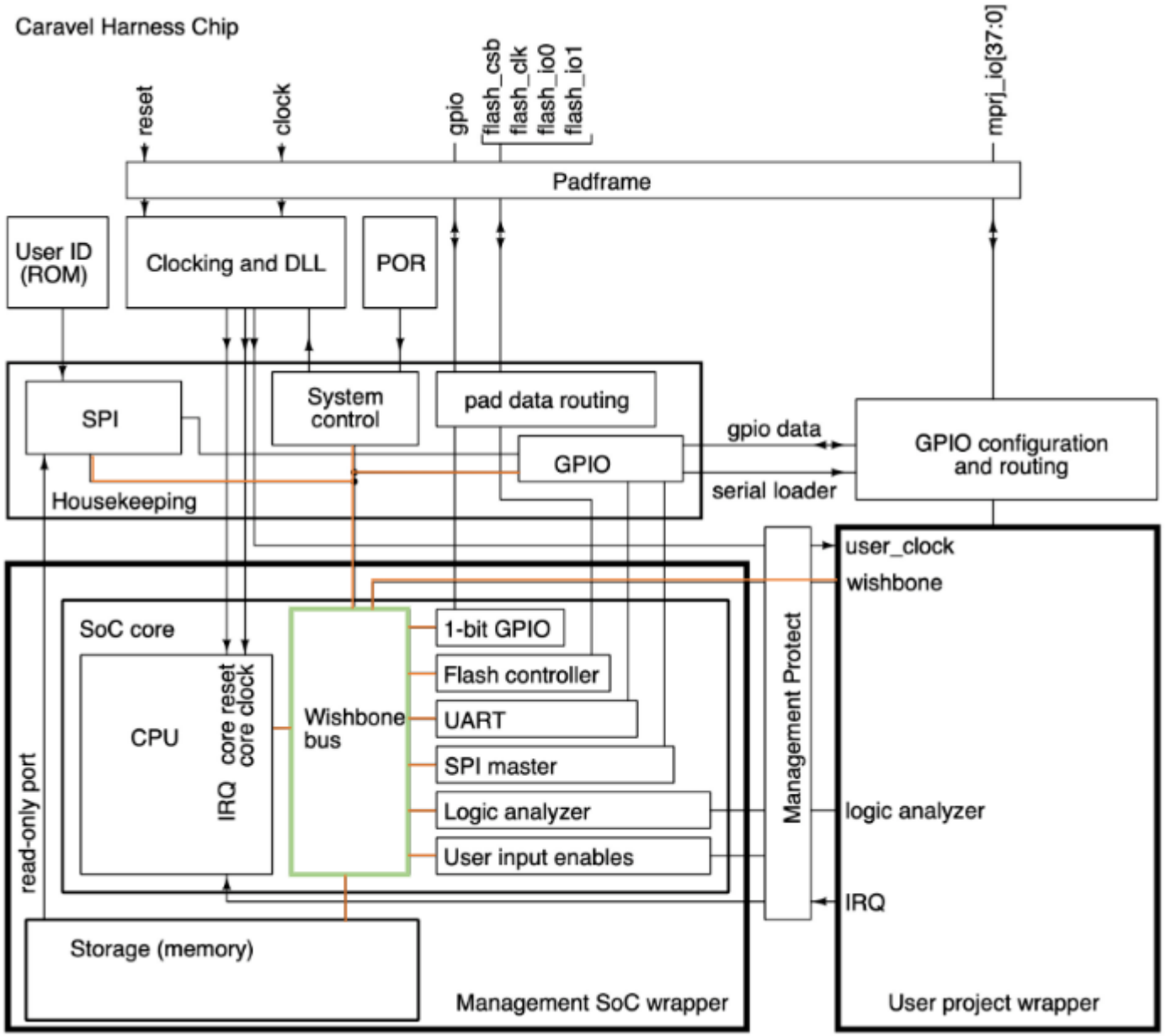
- WISHBONE Bus Cycles
  - SINGLE READ/WRITE
  - BLOCK READ/WRITE





Introduction

- Components on WISHBONE Bus
  - Master
    - SoC Core
      - CPU
      - UART
  - Slave
    - SoC core
      - 1-bit GPIO
      - Flash Controller
      - UART
      - SPI master
      - Logic analyzer
      - User input enables
    - User Project wrapper
    - Management SoC
      - Storage
    - Housekeeping
      - GPIO
      - SPI
      - System Control



Signals to WISHBONE Bus @VexRiscv

```
module VexRiscv (
  ...
  output              iBusWishbone_CYC,
  output              iBusWishbone_STB,
  input               iBusWishbone_ACK,
  output              iBusWishbone_WE,
  output     [29:0]   iBusWishbone_ADR,
  input      [31:0]   iBusWishbone_DAT_MISO,
  output     [31:0]   iBusWishbone_DAT_MOSI,
  output     [3:0]    iBusWishbone_SEL,
  input               iBusWishbone_ERR,
  output     [2:0]    iBusWishbone_CTI,
  output     [1:0]    iBusWishbone_BTE,
  output              dBusWishbone_CYC,
  output              dBusWishbone_STB,
  input               dBusWishbone_ACK,
  output              dBusWishbone_WE,
  output     [29:0]   dBusWishbone_ADR,
  input      [31:0]   dBusWishbone_DAT_MISO,
  output     [31:0]   dBusWishbone_DAT_MOSI,
  output reg [3:0]    dBusWishbone_SEL,
  input               dBusWishbone_ERR,
  output     [2:0]    dBusWishbone_CTI,
  output     [1:0]    dBusWishbone_BTE,
  ...
);
```

- VexRiscv WISHBONE Address output
  - iBusWishbone_ADR
    - assign iBusWishbone_ADR = (iBus_cmd_m2sPipe_payload_pc >>> 2);
  - dBusWishbone_ADR
    - assign dBusWishbone_ADR = (dBus_cmd_halfPipe_payload_address >>> 2);
- https://github.com/SpinalHDL/VexRiscv/blob/master/src/main/scala/vexriscv/plugin/

```
56    def getWishboneConfig() = WishboneConfig(
57      addressWidth = 30,
58      dataWidth = 32,
59      selWidth = 4,
60      useSTALL = false,
61      useLOCK = false,
62      useERR = true,
63      useRTY = false,
64      tgaWidth = 0,
65      tgcWidth = 0,
66      tgdWidth = 0,
67      useBTE = true,
68      useCTI = true
69    )
```

@DBusSimplePlugin.scala

```
180    def toWishbone(): Wishbone = {
181      val wishboneConfig = DBusSimpleBus.getWishboneConfig()
182      val bus = Wishbone(wishboneConfig)
183      val cmdStage = cmd.halfPipe()
184
185      bus.ADR := cmdStage.address >> 2
186      bus.CTI := B"000"
187      bus.BTE := "00"
188      bus.SEL := genMask(cmdStage).resized
189      when(!cmdStage.wr) {
190        bus.SEL := "1111"
191      }
```

@IBusSimplePlugin.scala

```
144    def toWishbone(): Wishbone = {
145      val wishboneConfig = IBusSimpleBus.getWishboneConfig()
146      val bus = Wishbone(wishboneConfig)
147      val cmdPipe = cmd.stage()
148
149      bus.ADR := (cmdPipe.pc >>  2)
150      bus.CTI := B"000"
151      bus.BTE := "00"
152      bus.SEL := "1111"
```

Round-robin for WB requests @Management Core

- wire [2:0] request;
- reg [1:0] grant = 2’d0;
- assign request = {dbg_uart_wishbone_cyc, mgmtsoc_dbus_dbus_cyc, mgmtsoc_ibus_ibus_cyc};
  - always @(posedge sys_clk) begin
    - Round-robin for requests

```
reg [29:0] comb_array_muxed0 = 30'd0;
reg [31:0] comb_array_muxed1 = 32'd0;
reg [3:0] comb_array_muxed2 = 4'd0;
reg comb_array_muxed3 = 1'd0;
reg comb_array_muxed4 = 1'd0;
reg comb_array_muxed5 = 1'd0;
reg [2:0] comb_array_muxed6 = 3'd0;
reg [1:0] comb_array_muxed7 = 2'd0;
```

```
always @(*) begin
    comb_array_muxed0 = 30'd0;
    case (grant)
        1'd0: begin
            comb_array_muxed0 = mgmtsoc_ibus_ibus_adr;
        end
        1'd1: begin
            comb_array_muxed0 = mgmtsoc_dbus_dbus_adr;
        end
        default: begin
            comb_array_muxed0 = dbg_uart_wishbone_adr;
        end
    endcase
end
```

```
case (grant)
    1'd0: begin
        if ((~request[0])) begin
            if (request[1]) begin
                grant <= 1'd1;
            end else begin
                if (request[2]) begin
                    grant <= 2'd2;
                end
            end
        end
    end
    1'd1: begin
        if ((~request[1])) begin
            if (request[2]) begin
                grant <= 2'd2;
            end else begin
                if (request[0]) begin
                    grant <= 1'd0;
                end
            end
        end
    end
    2'd2: begin
        if ((~request[2])) begin
            if (request[0]) begin
                grant <= 1'd0;
            end else begin
                if (request[1]) begin
                    grant <= 1'd1;
                end
            end
        end
    end
endcase
```

WISHBONE Share Signals @Management Core

- WISHBONE Shared Signals (shared_xxx)
  - Output
    - shared_adr, shared_dat_w, shared_sel, shared_cyc, shared_stb, shared_we, shared_cti, shared_bte
      - From selected requesters, comb_array_muxed0~comb_array_muxed7
  - Input
    - shared_ack, shared_dat_r
      - To selected requesters, mgmtsoc_dbus_dbus_xxx, mgmtsoc_ibus_ibus_xxx, dbg_uart_wishbone_xxx

```
wire [29:0] shared_adr;
wire [31:0] shared_dat_w;
reg [31:0] shared_dat_r = 32'd0;
wire [3:0] shared_sel;
wire shared_cyc;
wire shared_stb;
reg shared_ack = 1'd0;
wire shared_we;
wire [2:0] shared_cti;
wire [1:0] shared_bte;
```

```
assign shared_adr = comb_array_muxed0;
assign shared_dat_w = comb_array_muxed1;
assign shared_sel = comb_array_muxed2;
assign shared_cyc = comb_array_muxed3;
assign shared_stb = comb_array_muxed4;
assign shared_we = comb_array_muxed5;
assign shared_cti = comb_array_muxed6;
assign shared_bte = comb_array_muxed7;
assign mgmtsoc_ibus_ibus_dat_r = shared_dat_r;
assign mgmtsoc_dbus_dbus_dat_r = shared_dat_r;
assign dbg_uart_wishbone_dat_r = shared_dat_r;
assign mgmtsoc_ibus_ibus_ack = (shared_ack & (grant == 1'd0));
assign mgmtsoc_dbus_dbus_ack = (shared_ack & (grant == 1'd1));
assign dbg_uart_wishbone_ack = (shared_ack & (grant == 2'd2));
```

WISHBONE Signals for Slaves @Management Core

- Signals to each WISHBONE Slave
  - dff_bus
  - dff2_bus
  - mgmtsoc_litespimmap_bus
  - hk
  - mprj
  - mgmtsoc_wishbone
  - mgmptsoc_vexriscv_debug_bus
- WISHBONE slave selection
  - reg [6:0] slave_sel
    - Partial Decoding
  - Slave's CYC
    - shared_cyc & slave_sel[n]

Memory Regions

Region	Address	Size
dff	0x00000000	0x00000400
dff2	0x00000400	0x00000200
flash	0x10000000	0x01000000
hk	0x26000000	0x00100000
user project	0x30000000	0x10000000
csr	0xf0000000	0x00010000
vexriscv_debug	0xf00f0000	0x00000100

WISHBONE Signals for Slaves @Management Core

- WISHBONE Slave Signals
  - Output
    - To each WISHBONE Slave, mgmtsoc_vexriscv_debug_bus, dff_bus, dff2_bus, mgmtsoc_litespimmap_bus, mprj, hk, mgmtsoc_wishbone
      - E.g. mprj_adr, mprj_dat_w, mprj_sel, mprj_stb, mprj_we, mprj_cit, mprj_bte
      - From shared signals
  - Input
    - ((((((mgmtsoc_vexriscv_debug_bus_ack | dff_bus_ack) | dff2_bus_ack) | mgmtsoc_litespimmap_bus_ack) | ***mprj_ack***) | hk_ack) | mgmtsoc_wishbone_ack);
      - To shared_ack
    - (((((((({32{slave_sel_r[0]}} & mgmtsoc_vexriscv_debug_bus_dat_r) | ({32{slave_sel_r[1]}} & dff_bus_dat_r)) | ({32{slave_sel_r[2]}} & dff2_bus_dat_r)) | ({32{slave_sel_r[3]}} & mgmtsoc_litespimmap_bus_dat_r)) | ***({32{slave_sel_r[4]}} & mprj_dat_r)***) | ({32{slave_sel_r[5]}} & hk_dat_r)) | ({32{slave_sel_r[6]}} & mgmtsoc_wishbone_dat_r));
      - To shared_dat_r

WB Slaves Cycle Selection

- assign *mgmtsoc_vexriscv_debug_bus_cyc* = (shared_cyc & slave_sel[0]);
  - slave_sel[0] = shared_adr[29:6] == 24'd15732480, 0xF00F00 *(to RiscV Debug, 32h'F00F0000)*
- assign *dff_bus_cyc* = (shared_cyc & slave_sel[1]);
  - slave_sel[1] = shared_adr[29:8] == 1'd0, 0x0 *(to SRAM, 32'h0~32'h3FF)*
- assign *dff2_bus_cyc* = (shared_cyc & slave_sel[2]);
  - slave_sel[2] = shared_adr[29:7] == 2'd2, 0x2 *(to SRAM, 32h'400~32'h5FF)*
- assign *mgmtsoc_litespimmap_bus_cyc* = (shared_cyc & slave_sel[3]);
  - slave_sel[3] = shared_adr[29:22] == 5'd16, 0x10 *(to SPI Flash, 32h'10000000)*
- assign ***mprj_cyc*** = ***(shared_cyc & slave_sel[4])***;
  - **slave_sel[4] = shared_adr[29:26] == 2'd3, 0x3** ***(to User Project, 32'h30000000)***
- assign *hk_cyc* = (shared_cyc & slave_sel[5]);
  - slave_sel[5] = shared_adr[29:20] == 8'd152, 0x98 *(to House Keeping, 32h'26000000)*
- assign *mgmtsoc_wishbone_cyc* = (shared_cyc & slave_sel[6]);
- sSlave_sel[6] = shared_adr[29:14] == 16'd61440, 0xF000 *(to CSR, 32h'F0000000)*

```
always @(*) begin
    slave_sel = 7'd0;
    slave_sel[0] = (shared_adr[29:6] == 24'd15732480);
    slave_sel[1] = (shared_adr[29:8] == 1'd0);
    slave_sel[2] = (shared_adr[29:7] == 2'd2);
    slave_sel[3] = (shared_adr[29:22] == 5'd16);
    slave_sel[4] = (shared_adr[29:26] == 2'd3);
    slave_sel[5] = (shared_adr[29:20] == 8'd152);
    slave_sel[6] = (shared_adr[29:14] == 16'd61440);
end
```

WISHBONE Signals for Slaves @Management Core

- Signals of each WISHBONE Slaves

```
assign mgmtsoc_vexriscv_debug_bus_adr = shared_adr;
assign mgmtsoc_vexriscv_debug_bus_dat_w = shared_dat_w;
assign mgmtsoc_vexriscv_debug_bus_sel = shared_sel;
assign mgmtsoc_vexriscv_debug_bus_stb = shared_stb;
assign mgmtsoc_vexriscv_debug_bus_we = shared_we;
assign mgmtsoc_vexriscv_debug_bus_cti = shared_cti;
assign mgmtsoc_vexriscv_debug_bus_bte = shared_bte;
assign dff_bus_adr = shared_adr;
assign dff_bus_dat_w = shared_dat_w;
assign dff_bus_sel = shared_sel;
assign dff_bus_stb = shared_stb;
assign dff_bus_we = shared_we;
assign dff_bus_cti = shared_cti;
assign dff_bus_bte = shared_bte;
assign dff2_bus_adr = shared_adr;
assign dff2_bus_dat_w = shared_dat_w;
assign dff2_bus_sel = shared_sel;
assign dff2_bus_stb = shared_stb;
assign dff2_bus_we = shared_we;
assign dff2_bus_cti = shared_cti;
assign dff2_bus_bte = shared_bte;
assign mgmtsoc_litespimmap_bus_adr = shared_adr;
assign mgmtsoc_litespimmap_bus_dat_w = shared_dat_w;
assign mgmtsoc_litespimmap_bus_sel = shared_sel;
assign mgmtsoc_litespimmap_bus_stb = shared_stb;
assign mgmtsoc_litespimmap_bus_we = shared_we;
assign mgmtsoc_litespimmap_bus_cti = shared_cti;
assign mgmtsoc_litespimmap_bus_bte = shared_bte;
```

```
assign mprj_adr = shared_adr;
assign mprj_dat_w = shared_dat_w;
assign mprj_sel = shared_sel;
assign mprj_stb = shared_stb;
assign mprj_we = shared_we;
assign mprj_cti = shared_cti;
assign mprj_bte = shared_bte;
assign hk_adr = shared_adr;
assign hk_dat_w = shared_dat_w;
assign hk_sel = shared_sel;
assign hk_stb = shared_stb;
assign hk_we = shared_we;
assign hk_cti = shared_cti;
assign hk_bte = shared_bte;
assign mgmtsoc_wishbone_adr = shared_adr;
assign mgmtsoc_wishbone_dat_w = shared_dat_w;
assign mgmtsoc_wishbone_sel = shared_sel;
assign mgmtsoc_wishbone_stb = shared_stb;
assign mgmtsoc_wishbone_we = shared_we;
assign mgmtsoc_wishbone_cti = shared_cti;
assign mgmtsoc_wishbone_bte = shared_bte;
```

```
assign mgmtsoc_vexriscv_debug_bus_cyc = (shared_cyc & slave_sel[0]);
assign dff_bus_cyc = (shared_cyc & slave_sel[1]);
assign dff2_bus_cyc = (shared_cyc & slave_sel[2]);
assign mgmtsoc_litespimmap_bus_cyc = (shared_cyc & slave_sel[3]);
assign mprj_cyc = (shared_cyc & slave_sel[4]);
assign hk_cyc = (shared_cyc & slave_sel[5]);
assign mgmtsoc_wishbone_cyc = (shared_cyc & slave_sel[6]);
```

WISHBONE Signals for User Project Slave @Management Core

- Signals to User Project WISHBONE
  - Output
    - mprj_cyc_o , mprj_stb_o, mprj_we_o, mprj_sel_o, mprj_adr_o, mprj_dat_o
      - 30bits address to 32bits here
      - From mprj_cyc, mprj_stb, mprj_we, mprj_sel, mprj_adr, mprj_dat_w (from shared_xxx)
  - Input
    - mprj_ack_i, mprj_dat_i
      - To mprj_ack, mprj_dat_r (to shared_xxx)

```
output wire mprj_cyc_o,
output wire mprj_stb_o,
output wire mprj_we_o,
output wire [3:0] mprj_sel_o,
output reg [31:0] mprj_adr_o,
output wire [31:0] mprj_dat_o,
input wire [31:0] mprj_dat_i,
input wire mprj_ack_i,
```

```
assign mprj_cyc_o = mprj_cyc;
assign mprj_stb_o = mprj_stb;
assign mprj_we_o = mprj_we;
assign mprj_sel_o = mprj_sel;
always @(*) begin
    mprj_adr_o = 32'd0;
    mprj_adr_o[31:2] = mprj_adr;
    mprj_adr_o[1:0] = 1'd0;
end
assign mprj_dat_r = mprj_dat_i;
assign mprj_dat_o = mprj_dat_w;
assign mprj_ack = mprj_ack_i;
```

WISHBONE Signals for User Project Slave @Management Core Wrapper

- Signals to User Project WISHBONE
  - Output
    - mprj_cyc_o , mprj_stb_o, mprj_we_o, mprj_sel_o, mprj_adr_o, mprj_dat_o
      - From mprj_cyc_o , mprj_stb_o, mprj_we_o, mprj_sel_o, mprj_adr_o, mprj_dat_o (from Management Core)
  - Input
    - mprj_ack_i, mprj_dat_i
      - To mprj_ack_i, mprj_dat_i (to Management Core)

```
output    mprj_cyc_o,
output    mprj_stb_o,
output    mprj_we_o,
output [3:0]  mprj_sel_o,
output [31:0] mprj_adr_o,
output [31:0] mprj_dat_o,
input     mprj_ack_i,
input  [31:0] mprj_dat_i,
```

```
.mprj_ack_i(mprj_ack_i),
.mprj_dat_i(mprj_dat_i),
.mprj_cyc_o(mprj_cyc_o),
.mprj_stb_o(mprj_stb_o),
.mprj_we_o(mprj_we_o),
.mprj_sel_o(mprj_sel_o),
.mprj_adr_o(mprj_adr_o),
.mprj_dat_o(mprj_dat_o),
```

WISHBONE Signals for User Project Slave
@Carval

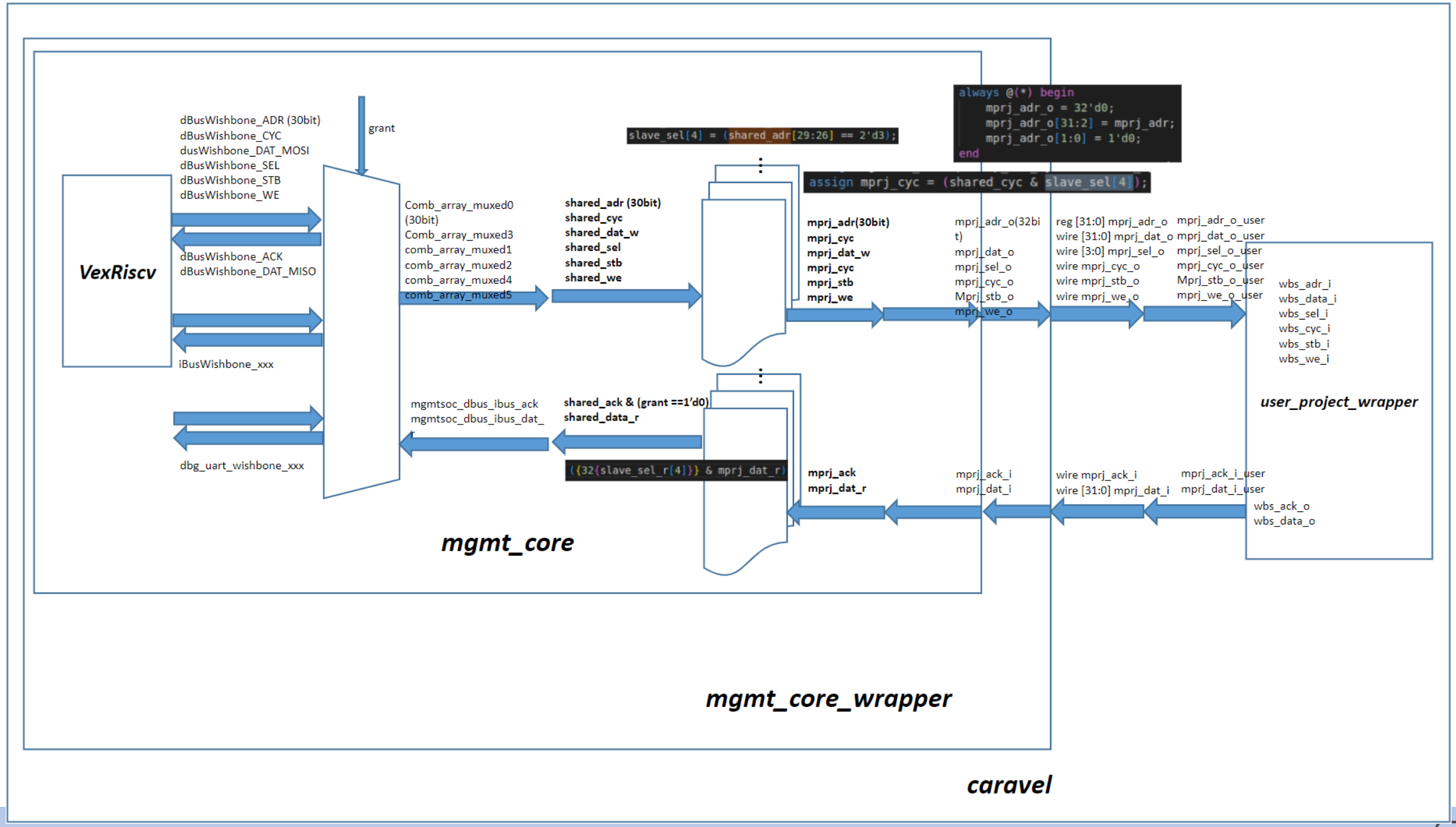
- Signals to User Project WISHBONE
  - Output
    - mprj_cyc_o_core (user), mprj_stb_o_core (user), mprj_we_o_core (user), mprj_sel_o_core (user), mprj_adr_o_core (user), mprj_dat_o_core (user)
      - From mprj_cyc_o , mprj_stb_o, mprj_we_o, mprj_sel_o, mprj_adr_o, mprj_dat_o (from Management Core Wrapper)
  - Input
    - mprj_ack_i_core (user), mprj_dat_i_core (user)
      - To mprj_ack_i, mprj_dat_i (to Management Core Wrapper)

```
wire mprj_cyc_o_core;
wire mprj_stb_o_core;
wire mprj_we_o_core;
wire [3:0] mprj_sel_o_core;
wire [31:0] mprj_adr_o_core;
wire [31:0] mprj_dat_o_core;
wire mprj_ack_i_core;
wire [31:0] mprj_dat_i_core;
```

```
.mprj_cyc_o(mprj_cyc_o_core),
.mprj_stb_o(mprj_stb_o_core),
.mprj_we_o(mprj_we_o_core),
.mprj_sel_o(mprj_sel_o_core),
.mprj_adr_o(mprj_adr_o_core),
.mprj_dat_o(mprj_dat_o_core),
.mprj_ack_i(mprj_ack_i_core),
.mprj_dat_i(mprj_dat_i_core),
```

```
assign mprj_cyc_o_user = mprj_cyc_o_core;
assign mprj_stb_o_user = mprj_stb_o_core;
assign mprj_we_o_user = mprj_we_o_core;
assign mprj_sel_o_user = mprj_sel_o_core;
assign mprj_adr_o_user = mprj_adr_o_core;
assign mprj_dat_o_user = mprj_dat_o_core;
assign mprj_dat_i_core = mprj_dat_i_user;
assign mprj_ack_i_core = mprj_ack_i_user;
```

```
.wbs_cyc_i(mprj_cyc_o_user),
.wbs_stb_i(mprj_stb_o_user),
.wbs_we_i(mprj_we_o_user),
.wbs_sel_i(mprj_sel_o_user),
.wbs_adr_i(mprj_adr_o_user),
.wbs_dat_i(mprj_dat_o_user),
.wbs_ack_o(mprj_ack_i_user),
.wbs_dat_o(mprj_dat_i_user),
```


dBusWishbone_ADR (30bit)
dBusWishbone_CYC
dusWishbone_DAT_MOSI
dBusWishbone_SEL
dBusWishbone_STB
dBusWishbone_WE
grant
VexRiscv
dBusWishbone_ACK
dBusWishbone_DAT_MISO
iBusWishbone_xxx
dbg_uart_wishbone_xxx
Comb_array_muxed0 (30bit)
Comb_array_muxed3
comb_array_muxed1
comb_array_muxed2
comb_array_muxed4
comb_array_muxed5
shared_adr (30bit)
shared_cyc
shared_dat_w
shared_sel
shared_stb
shared_we
slave_sel[4] = (shared_adr[29:26] == 2'd3);
always @(*) begin
mprj_adr_o = 32'd0;
mprj_adr_o[31:2] = mprj_adr;
mprj_adr_o[1:0] = 1'd0;
end
assign mprj_cyc = (shared_cyc & slave_sel[4]);
mprj_adr(30bit)
mprj_cyc
mprj_dat_w
mprj_cyc
mprj_stb
mprj_we
mprj_adr_o(32bit)
mprj_dat_o
mprj_sel_o
mprj_cyc_o
Mprj_stb_o
mprj_we_o
reg [31:0] mprj_adr_o
wire [31:0] mprj_dat_o
wire [3:0] mprj_sel_o
wire mprj_cyc_o
wire mprj_stb_o
wire mprj_we_o
mprj_adr_o_user
mprj_dat_o_user
mprj_sel_o_user
mprj_cyc_o_user
Mprj_stb_o_user
mprj_we_o_user
wbs_adr_i
wbs_data_i
wbs_sel_i
wbs_cyc_i
wbs_stb_i
wbs_we_i
user_project_wrapper
mgmtsoc_dbus_ibus_ack
mgmtsoc_dbus_ibus_dat_r
shared_ack & (grant ==1'd0)
shared_data_r
({32{slave_sel_r[4]}} & mprj_dat_r)
mprj_ack
mprj_dat_r
mprj_ack_i
mprj_dat_i
wire mprj_ack_i
wire [31:0] mprj_dat_i
mprj_ack_i_user
mprj_dat_i_user
wbs_ack_o
wbs_data_o
mgmt_core
mgmt_core_wrapper
caravel

Bridge of Life Education

Caravel SOC – User Project Design Interface

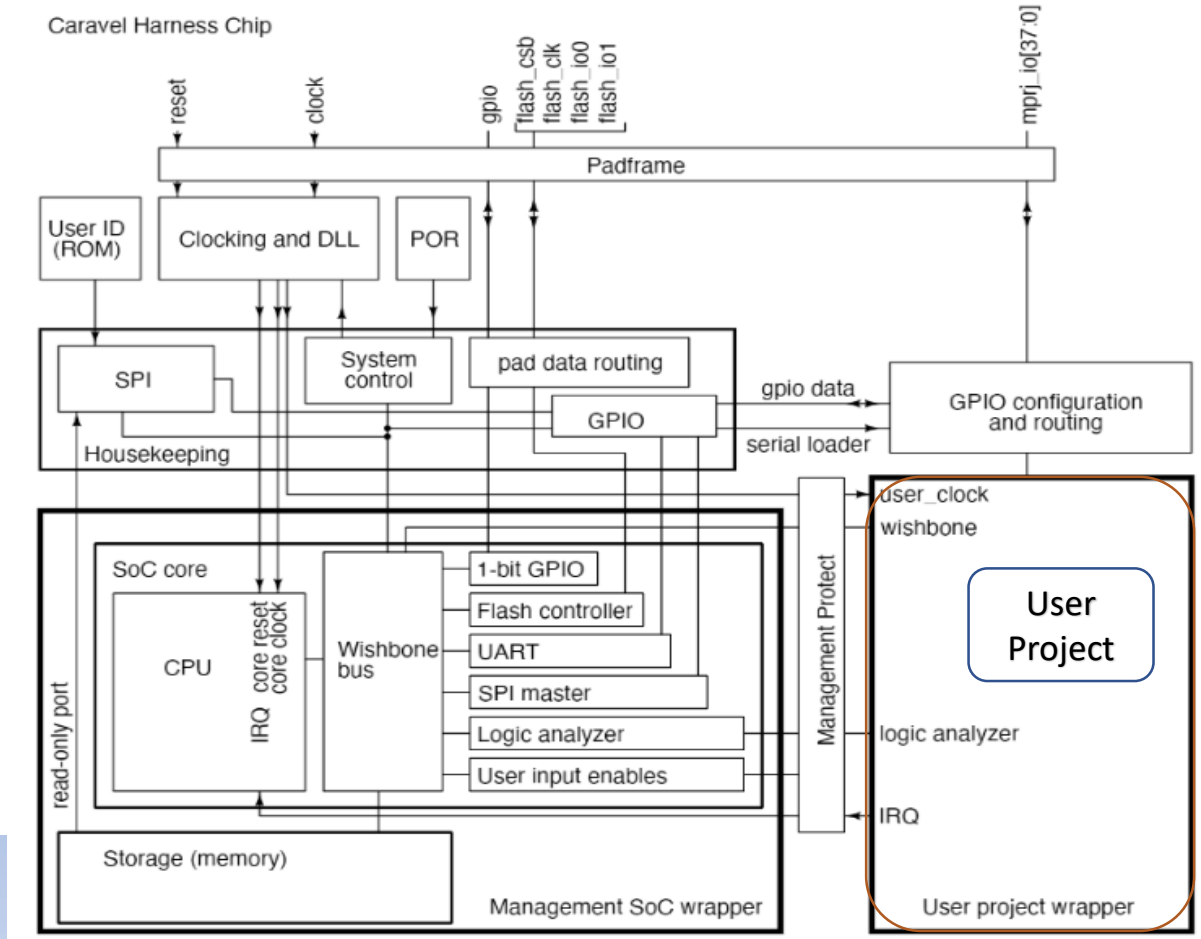
Josh

Outline

- Introduction
- Counter Design
  - Interface
  - Testbench & Firmware

Introduction

- User Project Wrapper
  - Providing the interface between Management Core and User Project
    - Wishbone
      - Range 0x30000000 ~ 0x3FFFFFFF
    - Logic Analyzer
      - [127:0]
    - MPRJ_IO
      - [37:0]
    - User Clock
    - IRQ
      - [2:0]
  - Implementation
    - user_project_wrapper.v



Introduction

- User Project Wrapper Implementation

```
user_proj_example mprj (
`ifdef USE_POWER_PINS
    .vccd1(vccd1),  // User area 1 1.8V power
    .vssd1(vssd1),  // User area 1 digital ground
`endif

    .wb_clk_i(wb_clk_i),
    .wb_rst_i(wb_rst_i),

    // MGMT SoC Wishbone Slave

    .wbs_cyc_i(wbs_cyc_i),
    .wbs_stb_i(wbs_stb_i),
    .wbs_we_i(wbs_we_i),
    .wbs_sel_i(wbs_sel_i),
    .wbs_adr_i(wbs_adr_i),
    .wbs_dat_i(wbs_dat_i),
    .wbs_ack_o(wbs_ack_o),
    .wbs_dat_o(wbs_dat_o),

    // Logic Analyzer

    .la_data_in(la_data_in),
    .la_data_out(la_data_out),
    .la_oenb (la_oenb),

    // IO Pads

    .io_in (io_in),
    .io_out(io_out),
    .io_oeb(io_oeb),

    // IRQ
    .irq(user_irq)
);
```

```
module user_project_wrapper #(
    parameter BITS = 32
) (
`ifdef USE_POWER_PINS
    inout vdda1,    // User area 1 3.3V supply
    inout vdda2,    // User area 2 3.3V supply
    inout vssa1,    // User area 1 analog ground
    inout vssa2,    // User area 2 analog ground
    inout vccd1,    // User area 1 1.8V supply
    inout vccd2,    // User area 2 1.8v supply
    inout vssd1,    // User area 1 digital ground
    inout vssd2,    // User area 2 digital ground
`endif

    // Wishbone Slave ports (WB MI A)
    input wb_clk_i,
    input wb_rst_i,
    input wbs_stb_i,
    input wbs_cyc_i,
    input wbs_we_i,
    input [3:0] wbs_sel_i,
    input [31:0] wbs_dat_i,
    input [31:0] wbs_adr_i,
    output wbs_ack_o,
    output [31:0] wbs_dat_o,

    // Logic Analyzer Signals
    input  [127:0] la_data_in,
    output [127:0] la_data_out,
    input  [127:0] la_oenb,

    // IOs
    input  [`MPRJ_IO_PADS-1:0] io_in,
    output [`MPRJ_IO_PADS-1:0] io_out,
    output [`MPRJ_IO_PADS-1:0] io_oeb,

    // Analog (direct connection to GPIO pad---use with caution)
    // Note that analog I/O is not available on the 7 lowest-numbered
    // GPIO pads, and so the analog_io indexing is offset from the
    // GPIO indexing by 7 (also upper 2 GPIOs do not have analog_io).
    inout [`MPRJ_IO_PADS-10:0] analog_io,

    // Independent clock (on independent integer divider)
    input   user_clock2,

    // User maskable interrupt signals
    output [2:0] user_irq
);
```

User Project Counter Design

- Interface to User Project Wrapper
  - Wishbone
  - Logic Analyzer
  - MPRJ_IO
  - IRQ (No used)

```
// WB MI A
assign valid = wbs_cyc_i && wbs_stb_i;
assign wstrb = wbs_sel_i & {4{wbs_we_i}};
assign wbs_dat_o = rdata;
assign wdata = wbs_dat_i;

// IO
assign io_out = count;
assign io_oeb = {(`MPRJ_IO_PADS-1){rst}};

// IRQ
assign irq = 3'b000;    // Unused

// LA
assign la_data_out = {{(127-BITS){1'b0}}, count};
// Assuming LA probes [63:32] are for controlling the count register
assign la_write = ~la_oenb[63:32] & ~{BITS{valid}};
// Assuming LA probes [65:64] are for controlling the count clk & reset
assign clk = (~la_oenb[64]) ? la_data_in[64]: wb_clk_i;
assign rst = (~la_oenb[65]) ? la_data_in[65]: wb_rst_i;
```

```
counter #(
    .BITS(BITS)
) counter(
    .clk(clk),
    .reset(rst),
    .ready(wbs_ack_o),
    .valid(valid),
    .rdata(rdata),
    .wdata(wbs_dat_i),
    .wstrb(wstrb),
    .la_write(la_write),
    .la_input(la_data_in[63:32]),
    .count(count)
);
```

```
    output [127:0] la_data_out,
    input  [127:0] la_oenb,

    // IOs
    input  [`MPRJ_IO_PADS-1:0] io_in,
    output [`MPRJ_IO_PADS-1:0] io_out,
    output [`MPRJ_IO_PADS-1:0] io_oeb,

    // IRQ
    output [2:0] irq
);
```

```
module counter #(
    parameter BITS = 32
)(
    input clk,
    input reset,
    input valid,
    input [3:0] wstrb,
    input [BITS-1:0] wdata,
    input [BITS-1:0] la_write,
    input [BITS-1:0] la_input,
    output ready,
    output [BITS-1:0] rdata,
    output [BITS-1:0] count

    reg ready;
    reg [BITS-1:0] count;
    reg [BITS-1:0] rdata;

    always @(posedge clk) begin
        if (reset) begin
            count <= 0;
            ready <= 0;
        end else begin
            ready <= 1'b0;
            if (~|la_write) begin
                count <= count + 1;
            end
            if (valid && !ready) begin
                ready <= 1'b1;
                rdata <= count;
                if (wstrb[0]) count[7:0]   <= wdata[7:0];
                if (wstrb[1]) count[15:8]  <= wdata[15:8];
                if (wstrb[2]) count[23:16] <= wdata[23:16];
                if (wstrb[3]) count[31:24] <= wdata[31:24];
            end else if (|la_write) begin
                count <= la_write & la_input;
            end
        end
    end

endmodule
```

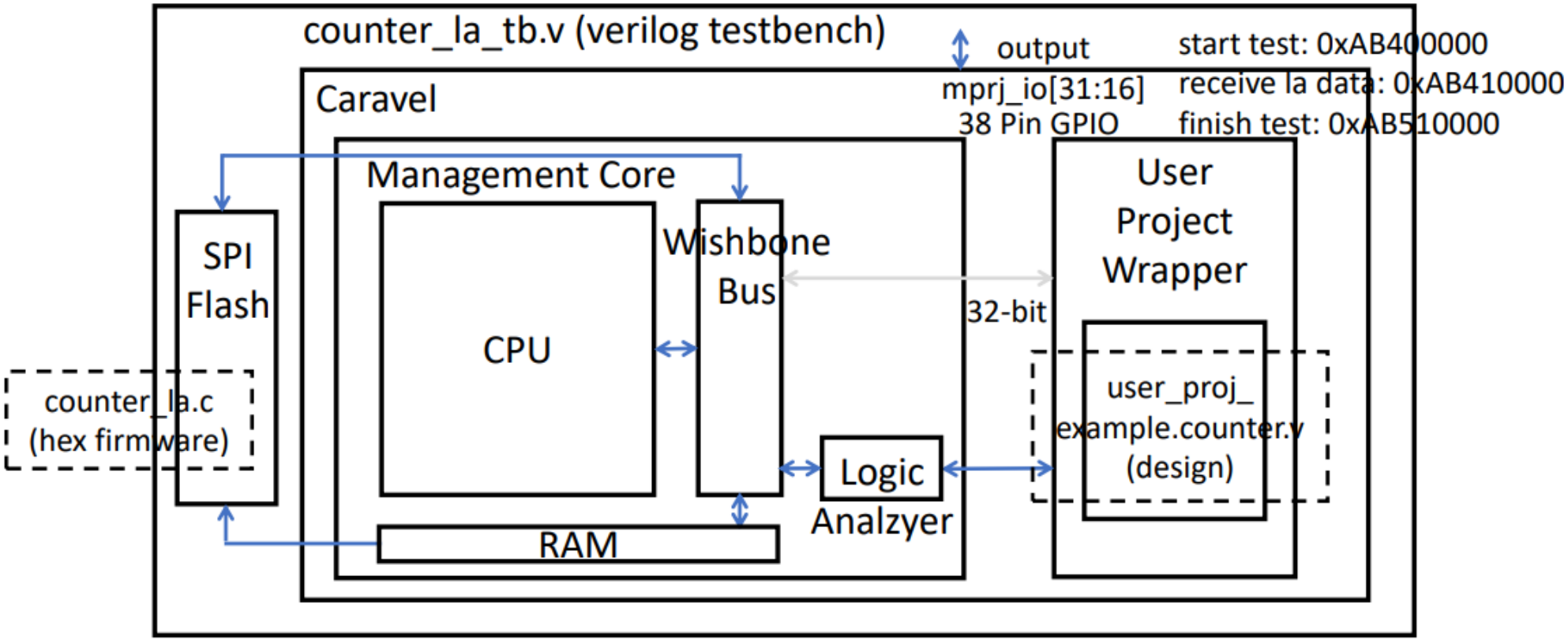
Bridge of Life Education

Caravel SOC – Testbench and Firmware

Josh

https://github.com/bol-edu/caravel-soc/files/11052078/caravel-soc_testbench.firmwave.pdf

Counter - LA



counter_la_tb.v: https://github.com/bol-edu/caravel-soc/blob/main/testbench/counter_la/counter_la_tb.v
counter_la.c: https://github.com/bol-edu/caravel-soc/blob/main/testbench/counter_la/counter_la.c
user_proj_example.counter.v: https://github.com/bol-edu/caravel-soc/blob/main/rtl/user/user_proj_example.counter.v

https://github.com/bol-edu/caravel-soc/files/11052078/caravel-soc_testbench.firmwave.pdf

counter_la_tb.v (verilog testbench)

assign checkbits = mprj_io[31:16] which was outputted from GPIO

```
29          wire [37:0] mprj_io;
30          wire [15:0] checkbits;
31
32          assign checkbits  = mprj_io[31:16];
```

wait checkbits == flag values from firmware (counter_la.c) then dispaly corresponded flag status

```
157         initial begin
158                 wait(checkbits == 16'hAB40);
159                 $display("LA Test 1 started");
160                 wait(checkbits == 16'hAB41);
161                 wait(checkbits == 16'hAB51);
162                 $display("LA Test 2 passed");
163                 #10000;
164                 $finish;
165         end
```

load compiled counter_la.c (counter_la.hex) to spiflash then executed by CPU

```
232         spiflash #(
233                 .FILENAME("counter_la.hex")
234         ) spiflash (
235                 .csb(flash_csb),
236                 .clk(flash_clk),
237                 .io0(flash_io0),
238                 .io1(flash_io1),
239                 .io2(),                 // not used
240                 .io3()                  // not used
241         );
242
```

counter_la_tb.v: https://github.com/bol-edu/caravel-soc/blob/main/testbench/counter_la/counter_la_tb.v

counter_la.c (firmware)

(1) config mprj_io[31:16] as output in counter_la.c (2) apply above configuration

```
61      (1)  reg_mprj_io_31 = GPIO_MODE_MGMT_STD_OUTPUT;
62           reg_mprj_io_30 = GPIO_MODE_MGMT_STD_OUTPUT;
63           reg_mprj_io_29 = GPIO_MODE_MGMT_STD_OUTPUT;
64           reg_mprj_io_28 = GPIO_MODE_MGMT_STD_OUTPUT;
65           reg_mprj_io_27 = GPIO_MODE_MGMT_STD_OUTPUT;
66           reg_mprj_io_26 = GPIO_MODE_MGMT_STD_OUTPUT;
67           reg_mprj_io_25 = GPIO_MODE_MGMT_STD_OUTPUT;
68           reg_mprj_io_24 = GPIO_MODE_MGMT_STD_OUTPUT;
69           reg_mprj_io_23 = GPIO_MODE_MGMT_STD_OUTPUT;
70           reg_mprj_io_22 = GPIO_MODE_MGMT_STD_OUTPUT;
71           reg_mprj_io_21 = GPIO_MODE_MGMT_STD_OUTPUT;
72           reg_mprj_io_20 = GPIO_MODE_MGMT_STD_OUTPUT;
73           reg_mprj_io_19 = GPIO_MODE_MGMT_STD_OUTPUT;
74           reg_mprj_io_18 = GPIO_MODE_MGMT_STD_OUTPUT;
75           reg_mprj_io_17 = GPIO_MODE_MGMT_STD_OUTPUT;
76           reg_mprj_io_16 = GPIO_MODE_MGMT_STD_OUTPUT;
```

:

```
100     (2)  // Now, apply the configuration
101          reg_mprj_xfer = 1;
102          while (reg_mprj_xfer == 1);
```

(3) enable la1 (4) flag start test (5) assign counter value via la (6) disable la1

```
104          // Configure LA probes [31:0], [127:64] as inputs to the cpu
105          // Configure LA probes [63:32] as outputs from the cpu
106          reg_la0_oenb = reg_la0_iena = 0x00000000;    // [31:0]
107     (3)  reg_la1_oenb = reg_la1_iena = 0xFFFFFFFF;    // [63:32]
108          reg_la2_oenb = reg_la2_iena = 0x00000000;    // [95:64]
109          reg_la3_oenb = reg_la3_iena = 0x00000000;    // [127:96]
110
111          // Flag start of the test
112     (4)  reg_mprj_datal = 0xAB400000;
113
114          // Set Counter value to zero through LA probes [63:32]
115     (5)  reg_la1_data = 0x00000000;
116
117          // Configure LA probes from [63:32] as inputs to disable counter write
118     (6)  reg_la1_oenb = reg_la1_iena = 0x00000000;
```

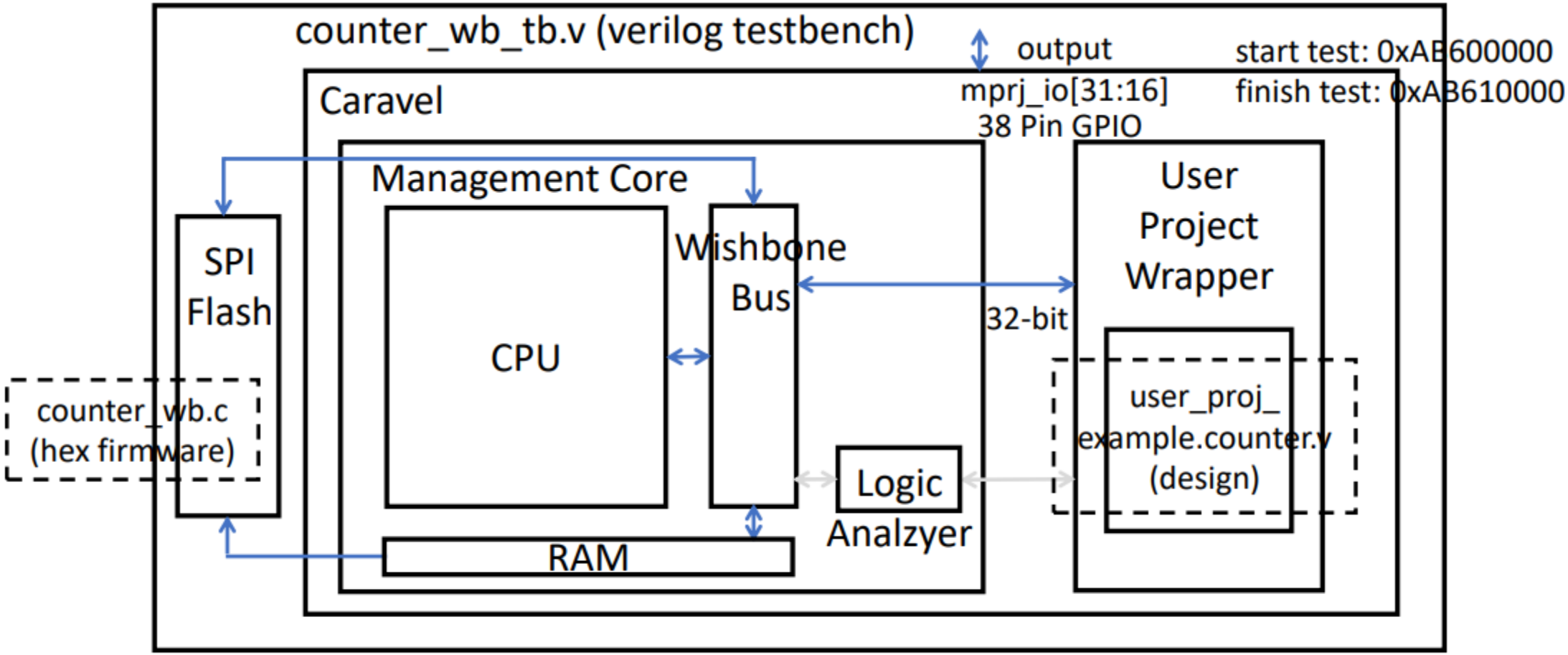
(7) if counter output value > 0x1F4 then flag 0xAB41 and break while (8) flag 0xAB51 at finish

```
120          while (1) {
121     (7)          if (reg_la0_data_in > 0x1F4) {
122                       reg_mprj_datal = 0xAB410000;
123                       break;
124                  }
125          }
126          //print("\n");
127          //print("Monitor: Test 1 Passed\n\n");  // Makes simulation very long
128     (8)  reg_mprj_datal = 0xAB510000;
```

note: reg_mprj_datal = mprj_io[31:16]

counter_la.c: https://github.com/bol-edu/caravel-soc/blob/main/testbench/counter_la/counter_la.c

Counter - WB



counter_wb_tb.v: https://github.com/bol-edu/caravel-soc/blob/main/testbench/counter_wb/counter_wb_tb.v
counter_wb.c: https://github.com/bol-edu/caravel-soc/blob/main/testbench/counter_wb/counter_wb.c
user_proj_example.counter.v: https://github.com/bol-edu/caravel-soc/blob/main/rtl/user/user_proj_example.counter.v

https://github.com/bol-edu/caravel-soc/files/11052078/caravel-soc_testbench.firmwave.pdf

counter_wb_tb.v (verilog testbench)

assign checkbits = mprj_io[31:16] which was outputted from GPIO

```
28          wire [37:0] mprj_io;
29          wire [7:0] mprj_io_0;
30          wire [15:0] checkbits;
31
32          assign checkbits = mprj_io[31:16];
```

wait checkbits == flag values from firmware (counter_wb.c) then dispaly corresponded flag status

```
160         initial begin
161                 wait(checkbits == 16'hAB60);
162                 $display("Monitor: MPRJ-Logic WB Started");
163                 wait(checkbits == 16'hAB61);
164                 `ifdef GL
165                 $display("Monitor: Mega-Project WB (GL) Passed");
166                 `else
167                     $display("Monitor: Mega-Project WB (RTL) Passed");
168                 `endif
169             $finish;
170         end
```

load compiled counter_wb.c (counter_wb.hex) to spiflash then executed by CPU

```
232         spiflash #(
233                 .FILENAME("counter_wb.hex")
234         ) spiflash (
235                 .csb(flash_csb),
236                 .clk(flash_clk),
237                 .io0(flash_io0),
238                 .io1(flash_io1),
239                 .io2(),                 // not used
240                 .io3()                  // not used
241         );
```

counter_wb_tb.v: https://github.com/bol-edu/caravel-soc/blob/main/testbench/counter_wb/counter_wb_tb.v

https://github.com/bol-edu/caravel-soc/files/11052078/caravel-soc_testbench.firmwave.pdf

counter_wb.c (firmware)

(1) enable wb (2) config mprj_io[31:16] as output in counter_wb.c (3) apply above configuration

```
50  (1) reg_spi_enable = 1;
51      reg_wb_enable = 1;
        :
59  (2) reg_mprj_io_31 = GPIO_MODE_MGMT_STD_OUTPUT;
60      reg_mprj_io_30 = GPIO_MODE_MGMT_STD_OUTPUT;
61      reg_mprj_io_29 = GPIO_MODE_MGMT_STD_OUTPUT;
62      reg_mprj_io_28 = GPIO_MODE_MGMT_STD_OUTPUT;
63      reg_mprj_io_27 = GPIO_MODE_MGMT_STD_OUTPUT;
64      reg_mprj_io_26 = GPIO_MODE_MGMT_STD_OUTPUT;
65      reg_mprj_io_25 = GPIO_MODE_MGMT_STD_OUTPUT;
66      reg_mprj_io_24 = GPIO_MODE_MGMT_STD_OUTPUT;
67      reg_mprj_io_23 = GPIO_MODE_MGMT_STD_OUTPUT;
68      reg_mprj_io_22 = GPIO_MODE_MGMT_STD_OUTPUT;
69      reg_mprj_io_21 = GPIO_MODE_MGMT_STD_OUTPUT;
70      reg_mprj_io_20 = GPIO_MODE_MGMT_STD_OUTPUT;
71      reg_mprj_io_19 = GPIO_MODE_MGMT_STD_OUTPUT;
72      reg_mprj_io_18 = GPIO_MODE_MGMT_STD_OUTPUT;
73      reg_mprj_io_17 = GPIO_MODE_MGMT_STD_OUTPUT;
74      reg_mprj_io_16 = GPIO_MODE_MGMT_STD_OUTPUT;
        :
76  (3) /* Apply configuration */
77      reg_mprj_xfer = 1;
78      while (reg_mprj_xfer == 1);
```

(4) disable la (5) flag start test
(6) assign counter value via wb slave

```
80  (4) reg_la2_oenb = reg_la2_iena = 0x00000000;    // [95:64]
81
82      // Flag start of the test
83  (5) reg_mprj_datal = 0xAB600000;
84
85  (6) reg_mprj_slave = 0x00002710;
```

(7) If counter output value >= 2B3D) then flag 0xAB61 at finish

```
//reg_mprj_datal = 0xAB600000;
while(1){
    if (reg_mprj_slave >= 0x2B3D) {
        reg_mprj_datal = 0xAB610000;
        break;
    }
}
```

note: reg_mprj_datal = mprj_io[31:16]

counter_wb.c: https://github.com/bol-edu/caravel-soc/blob/main/testbench/counter_wb/counter_wb.c