1. PS Mode programming

Introduction

HME-HR devices have 3 configuration modes: JTAG, AS and PS mode. In the PS mode, HME-HR family works as a slave device, receives configuration data from external master controller passively. SPI Master cannot read configuration data from HME-HR family.

Configuration Mode and Pins

The bitstream is loaded into the device through special configuration pins. These configuration pins serve as the interface for a number of different configuration modes:

There are 3 configuration modes: JTAG, AS and PS mode which are defined by two dedicated JM_B and SS pins. After exiting the Power-On Reset (POR) state or when nCONFIG returns High after being held Low, the HME-HR device samples the logical value on its SS and JM_B pins.

The configuration modes are described in table below.

<table>
<thead>
<tr>
<th>Mode pin</th>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JM_B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>1</td>
<td>AS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Active Serial mode. The chip will be configured automatically. Configuration data is stored in the SPI flash.</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td>PS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chip acts as slave. External microcontroller feeds configuration data into the chip.</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td>JTAG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JTAG-based configuration. This mode takes high precedence over AS and PS modes.</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>AS/PS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JTAG can’t configure the HME-HR.</td>
</tr>
</tbody>
</table>
### Table 2 Configuration Pins

<table>
<thead>
<tr>
<th>SPI serial configuration Pins</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SCLK</td>
<td>input/output</td>
</tr>
<tr>
<td>SDI</td>
<td>output</td>
</tr>
<tr>
<td>SDO</td>
<td>Input</td>
</tr>
<tr>
<td>SS</td>
<td>output or input</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dedicated Configuration Pins</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CFGDONE</td>
<td>output</td>
</tr>
<tr>
<td>nCONFIG</td>
<td>input</td>
</tr>
</tbody>
</table>

**PS Scheme**

The PS schematic is shown below. The HME-HR’s SPI configuration interface is a separate, independent I/O bank, powered by the VDDIO_33 supply input. Typically, VDDIO_33 is the same voltage as the application processor’s I/O. The configuration control signals, CFGDONE and nCONFIG, are supplied by the separate I/O Bank 1 voltage input, VDDIO_1.
PS Configuration Process

You can perform PS configuration on HME-HR device family with an external intelligent host, such as a microprocessor with flash memory. In the PS scheme, an external host controls the configuration process. After configuration, the SPI port pins are available to the user-application as additional PIO pins, supplied by the VDDIO_33 input voltage.

Figure 2 illustrates the interface timing for the PS mode.

1) Reset process. The application host begins by driving the HME-HR nCONFIG pin low, resetting the HME-HR FPGA. Similarly, the host holds the HME-HR’s SS pin Low. The host must hold the nCONFIG pin Low for at least 500 ns. Ultimately, the host either releases the nCONFIG pin and allows it to float High via the 10 KOhm pull-up resistor to VCCIO_1 or drives nCONFIG High. The HME-HR FPGA enters PS mode when the nCONFIG pin returns High while the SS pin is Low and last at least 10us and less than 125us.

2) Initialization process. The SS pin must set to High and remain a minimum of 1250 µs to wait the Initialization process finish. Then the SS is driven to Low for configuration process.

3) Configuring process. The host sends the configuration image(*acf.hex) generated by the Fuxi (See the chapter 2 for the generation method) to HME-HR FPGA when the SS is driven to low. The host must place the configuration data one bit at a time on the SDO pin at the negedge of the SCLK. The most-significant bit(MSB) of each byte must be sent first. For example, if the configuration data *.acf contains the byte sequence 02 1B EE 01 FA, the serial bitstream you must send to the device is:

0000-0010 0001-1011 1110-1110 0000-0001 1111-1010

HME-HR device family receives configuration data on the SDO pin and the clock is received on the SCLK pin. Data is latched into the device on the rising edge of SCLK. Data is continuously clocked into the target device until all the configuration data is sent and the SS is driven to high.

4) Addition dummy process. After sending the entire image, the HME-HR FPGA CFGDONE goes high
and the device enters the user mode. If the CFGDONE pin remains Low, then an error occurred during configuration and the host should handle the error accordingly for the application. The host must send extra 64 SCLK clock cycles measured from rising-edge to rising-edge to HME-HR device. After the additional SCLK cycles, the SPI interface pins then become available to the user application loaded in FPGA.

![Figure 2 PS Configuration Waveform](image)

To reconfigure the HME-HR FPGA or to load a different configuration image, merely restart the configuration process by pulsing nCONFIG Low or power-cycling the FPGA.

The max PS SCLK clock frequency is about 25MHz.

**PS Configuration Reference Code**

The PS code example is shown below. The 8051 MCU PS host uses the GPIO to control the PS master’s signals. The below code signals are from the master’s point of view.

```c
#define SDO P0_0 // SDO
#define CS P0_1 // CS
#define SCLK P0_2 // SCLK
#define nCONFIG P0_3 // nCONFIG
#define P0_ENN P0_5 // OEN
#define START P2_0 // start
#define CS_HIGH 1
#define CS_LOW 0
#define HR3 1 // 1 for HR3 0 for HR2
#define HR3_SF_ADDR 0x30000
#define HR3_FILE_LEN 0xcf00 // HR3 bitstream length
#define HR2_FILE_LEN 0x6d00 // HR2 bitstream length
#define HR2_SF_ADDR 0x60000
#define RESET_LOW 1500 //
#define DELAY_T1 200 //
#define DELAY_T2min 150
#define DELAY_T3 650
#define DELAY_T

void delay(unsigned int times) // cpu clock is 100MHz
{

```
unsigned int i;
unsigned char j;
for(i = 0; i < times; i++)
    for(j = 0; j < 2; j++)
    {
        j=j;
    }
}

void HR_PSConfig_8b(void)
{
    unsigned int HR_PS_pre_clk;
    unsigned char sf_buf[256];
    unsigned int i,ps_x;
    unsigned char i_mask;
    unsigned char bit_mask;
    unsigned long read_addr,read_len;
    unsigned int pcnt;

    nCONFIG = 0;
    CS=CS_LOW;  //cs must 0
    SDO =0;      //sdo must 0
    SCLK =0;
    delay(RESET_LOW);  //1us
    //Reset process,nCONFIG 1us low level
    CS=CS_LOW;
    nCONFIG = 1;
    //Initialization process
    delay(DELAY_T2min);  //<125us

    // Clock switch process
    CS=CS_HIGH;
    delay(DELAY_T);
    for(HR_PS_pre_clk=0;HR_PS_pre_clk<8;HR_PS_pre_clk++) //HR3 8=pass //>1250
    {
        SCLK=1;//clk
        delay(100*DELAY_T);
        SCLK=0;//clk
        delay(100*DELAY_T);
// Configuring internal configuration memory process, the HR3/2 image is stored in M5 spi flash 0x30000/0x60000

if (HR3){
    read_len = HR3_FILE_LEN;
    read_addr = HR3_SF_ADDR;
}
else {
    read_len = HR2_FILE_LEN;
    read_addr = HR2_SF_ADDR;
}

pcnt = (read_len >> 8) & 0xFFFFFFFF;
for(i=0;i<= pcnt;i ++)
{
    //Get next from spi flash 256 bytes data to LBA_Buff buffer.
    //GetNextBlockData();
    SpiFlashRead(read_addr,sf_buf,256);
    for(ps_x=0;ps_x<256;ps_x++)
    {
        bit_mask=0x80;
        for(i_mask=0;i_mask<8;i_mask++)
        {
            if(sf_buf[ps_x] & bit_mask)//data
                SDO=1;
            else
                SDO=0;
            delay(DELAY_T);
            SCLK=1;//clk
            delay(DELAY_T);
            SCLK=0;//clk
            bit_mask>>=1;
        }
        read_addr = read_addr +256;
    }
    CS=CS_HIGH;
// Addition dummy process
for(i_mask=0;i_mask<64;i_mask++)
{
    delay(DELAY_T);
    SCLK=1;//clk
    delay(DELAY_T);
    SCLK=0;//clk
}
}

void main(void)
{
    CKCON = 0x00;
    HR_PSConfig_8b(); // Host PS 配置 HR device.
    while(1)
    {
        P1 = 0xff;
        delay(65535);
        P1 = 0x00;
        delay(65535);
    }

2. “ps.bin” file generated by Fuxi

- Step 1: Click the tools menu to "Command Line Window..."Command, then a command window will be opened, as shown in the figure 3
- Step 2: Key "cd outputs" and enter the output folder under the project directory.
- Step 3: key "acftbin *.acf"
- Step 4: The *.acf.bin file will be generated in the output folder under project directory.