



1

# Computer Organization

## Lab14 Practice on Uart port with CPU

2021 Spring term

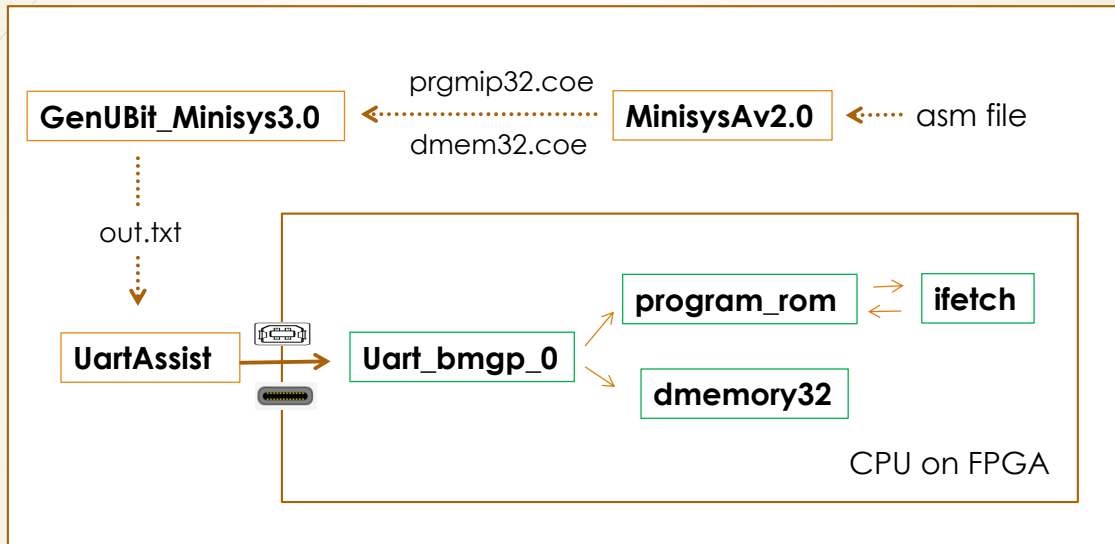
wangw6@sustech.edu.cn

# Topics

## ➤ CPU

- What is a CPU (General purpose processor)
- How to make CPU work on a new program
  - communicate with UART port to get coe file to rewrite ProgramRAM and DataRAM
- Let's Do it
  - 2 tools and modification on CPU code

## Update The Data In Instruction And Data Memory



"GenUBit\_Minisys3.0", "UartAssist" and "MinisysAv2.0" could be found in the "Uart tools" of sakai site:

<https://sakai.sustech.edu.cn/portal/site/d50211ea-1586-4344-9d92-a6c42eb7f4e0/tool/b47e4c13-4220-4f61-b881-a211abee2a96?panel=Main>

# Changes on Single Cycle CPU

- 1. There should be two working modes on the CPU
  - normal mode & Uart communication mode
- 2. A new module which works as Uart interface
- 3. Changes on CPU top
  - New module, new ports, new internal connection and new logic
- 4. Changes on Data-memory and IFetch
  - Data-memory, working mode: normal mode & uart communication mode
  - IFetch
    - Change IP core "program" from ROM to RAM
    - Working mode: normal mode & uart communication mode
    - Separate "program" from IFetch (optional)

# Changes on CPU Top Module

```

module CPU_TOP(
  input fpga_rst, //Active High
  input fpga_clk,
  input[23:0] switch2N4,
  output[23:0] led2N4,

  // UART Programmer Pinouts
  // start Uart communicate at high level
  input start_pg, //Active High
  input rx, // receive data by UART
  output tx // send data by UART
);

```

```

// UART Programmer Pinouts
wire upg_clk, upg_clk_o;
wire upg_wen_o; //Uart write out enable
wire upg_done_o; //Uart rx data have done
//data to which memory unit of program_rom/dmemory32
wire [14:0] upg_adr_o;
//data to program_rom or dmemory32
wire [31:0] upg_dat_o;

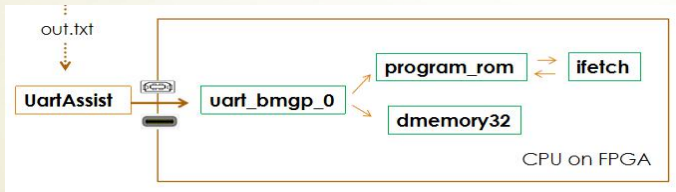
```

```

wire spg_bufg;
BUFGLUT1(I(start_pg), .O(spg_bufg)); // de-twitter
// Generate UART Programmer reset signal
reg upg_rst;
always @ (posedge fpga_clk) begin
  if (spg_bufg) upg_rst = 0;
  if (fpga_rst) upg_rst = 1;
end
assign rst = fpga_rst | !upg_rst;

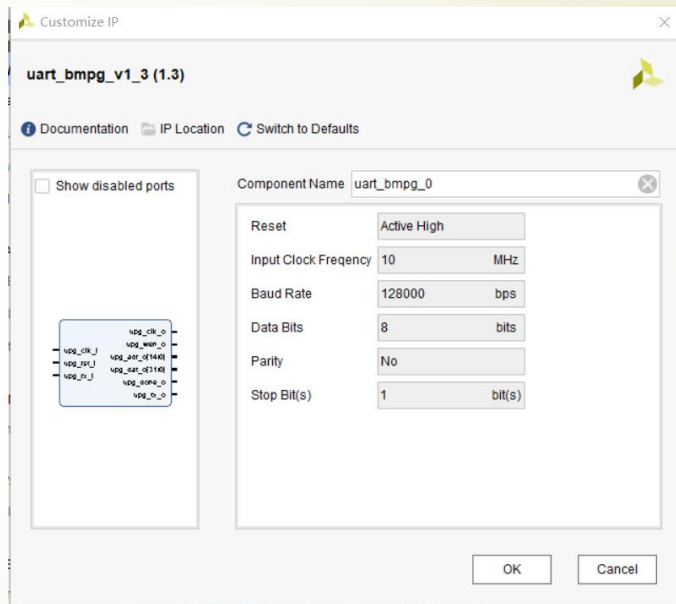
```

set\_property -dict {IOSTANDARD LVCMOS33 PACKAGE\_PIN Y19} [get\_ports rx]  
 set\_property -dict {IOSTANDARD LVCMOS33 PACKAGE\_PIN V18} [get\_ports tx]



# Add an IP core Which Processes Uart Data

- Add the IP core to IP catalog of vivado
- Add the IP core(uart\_bmpg\_0) from IP catalog into vivado project, then instance it
- **NOTICE: Don't change the settings of this IP core**
- The Communication between this IP core and Uart port:
  - receive data from Uart port and forward to data-memory and instruction-memory
  - send data back to uart port to info that all the data has been received.




Get the IP core from:

<https://sakai.sustech.edu.cn/portal/site/d50211ea-1586-4344-9d92-a6c42eb7f4e0/tool/b47e4c13-4220-4f61-b881-a211abee2a96?panel=Main>

# Add a New Clock For The New IP core

- Reset the “cpuck” IP core to make a new clock
  - Add a new clk\_out (clk\_out2) whose frequency is 10 Mhz for the IP core which is used for Uart communication (page 6)

>  cpuck : cpuck (cpuck.xci)



Component Name: cpuck

Clocking Options | **Output Clocks** | Port Renaming | PLLE2 Settings | Summary

The phase is calculated relative to the active input clock.

Output Clock	Port Name	Output Freq (MHz)		Phase (d)
		Requested	Actual	
<input checked="" type="checkbox"/> clk_out1	clk_out1	23.0	23.000	0.000
<input checked="" type="checkbox"/> clk_out2	clk_out2	10.0	10.000	0.000

Handwritten red annotations: 'single\_cycle\_cpu\_clk' with an arrow pointing to the 'Requested' column header, and 'uart\_clk' with an arrow pointing to the 'Requested' value '10.0' in the second row.

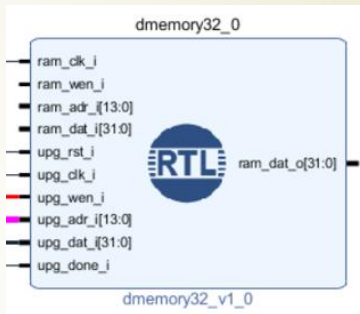
# Changes on Demeory32

```
module dmemory32 (
    input      ram_clk_i,    // from CPU top
    input      ram_wen_i,    // from controller
    input [13:0] ram_adr_i,   // from alu_result of ALU
    input [31:0] ram_dat_i,   // from read_data_2 of decoder
    output [31:0] ram_dat_o,  // the data read from ram
```

// UART Programmer Pinouts

```
input      upg_rst_i,    // UPG reset (Active High)
input      upg_clk_i,    // UPG ram_clk_i (10MHz)
input      upg_wen_i,    // UPG write enable
input [13:0] upg_adr_i,  // UPG write address
input [31:0] upg_dat_i,  // UPG write data
input      upg_done_i    // 1 if programming is finished
```

```
);
```



```
wire ram_clk = !ram_clk_i;
```

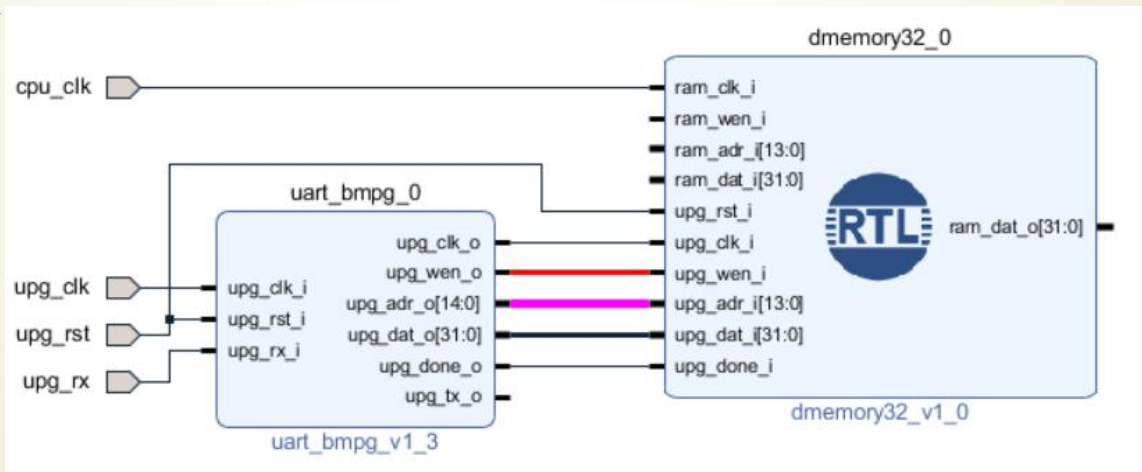
```
// CPU work on normal mode when kickOff is 1.
//CPU work on Uart communicate mode when kickOff is 0.
wire kickOff = upg_rst_i | (~upg_rst_i & upg_done_i);
```

```
ram ram (
    .clka (kickOff ? ram_clk : upg_clk_i),
    .wea (kickOff ? ram_wen_i : upg_wen_i),
    .addra (kickOff ? ram_adr_i : upg_adr_i),
    .dina (kickOff ? ram_dat_i : upg_dat_i),
    .douta (ram_dat_o)
);
```



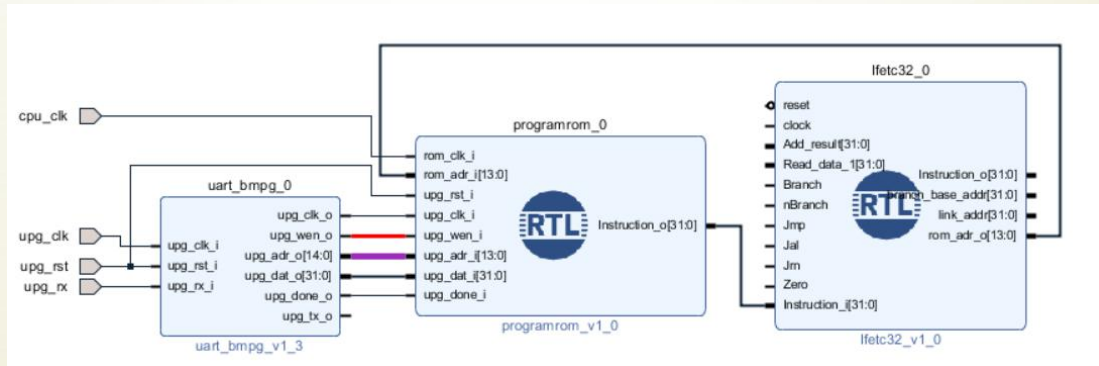
# Changes on Demeory32 continued

- ▶ **upg\_wen\_i** (uart write enable on Dmemory32) :
  - ▶ determined by: **upg\_wen\_o**(from uart\_bmpg) & **upg\_adr\_o[14]** (from uart\_bmpg)
- ▶ **upg\_adr\_i[13:0]** (uart write address on Dmemory32):
  - ▶ connect with: **upg\_adr\_o[13:0]** (from uart\_bmpg)



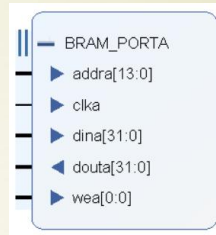
# Changes on IFetch

- Separate IP core which stores the Instruction from IFetch(optional)
- Change program from ROM to RAM(writeable while work on Uart communication mode, read only while work on normal mode)
- upg\_wen\_i** (uart write enable on "programrom"), determined by: **upg\_wen\_o**(from uart\_bmpg) & **(!upg\_adr\_o[14])** (from uart\_bmpg)
- upg\_adr\_i[13:0]** (uart write address on "programrom"), connect with **upg\_adr\_o[13:0]** (from uart\_bmpg)



# Changes on IFetch continued

- Make a new programrom(which is a RAM memory)



Basic	Port A Options	Other Options	Summary
Interface Type	Native	<input type="checkbox"/> Generate address interface with 32	
Memory Type	Single Port RAM	<input type="checkbox"/> Common Clock	
<b>ECC Options</b>			
ECC Type	No ECC		
<input type="checkbox"/> Error Injection Pins	Single Bit Error Injection		
<b>Write Enable</b>			
<input type="checkbox"/> Byte Write Enable			
Byte Size (bits)	9		
<b>Algorithm Options</b>			
Defines the algorithm used to concatenate the block RAM primitives. Refer datasheet for more information.			
Algorithm	Minimum Area		
Primitive	8lx2		

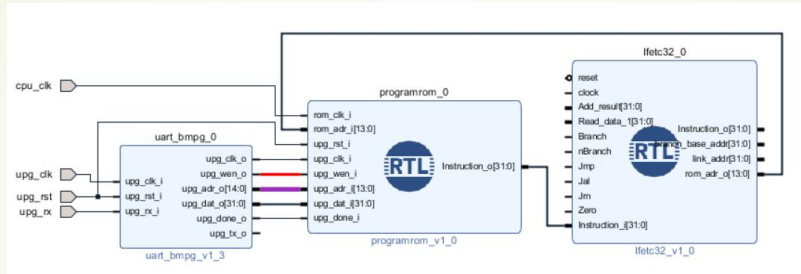
  

Basic	Port A Options	Other Options	Summary
<b>Memory Size</b>			
Write Width	32	Range: 1 to 4608 (bits)	
Read Width	32		
Write Depth	16384	Range: 2 to 1048576	
Read Depth	16384		
<b>Operating Mode</b> Write First <input type="checkbox"/> Enable Port Type Always Enabled			
<b>Port A Optional Output Registers</b>			
<input type="checkbox"/> Primitives Output Register	<input type="checkbox"/> Core Output Register		
<input type="checkbox"/> SoftECC Input Register	<input type="checkbox"/> REGCEA Pin		
<b>Port A Output Reset Options</b>			
<input type="checkbox"/> RSTA Pin (set/reset pin)	Output Reset Value (Hex)	0	
<input type="checkbox"/> Reset Memory Latch	Reset Priority	CE (Latch or Register Enable)	
<b>READ Address Change A</b>			

Basic	Port A Options	Other Options	Summary
Pipeline Stages within Mux	0	Mux Size: 4x1	
<b>Memory Initialization</b>			
<input checked="" type="checkbox"/> Load Init File			
Coe File	../../../../minisys.srscs/sources_1/ip/prgrom/prgmip32.coe	<input type="button" value="Browse"/>	
<input checked="" type="checkbox"/> Fill Remaining Memory Locations			
Remaining Memory Locations (Hex)	0		
<b>Structural/UniSim Simulation Model Options</b>			
Defines the type of warnings and outputs are generated when a read-write or write-write collision occurs.			
Collision Warnings	All		
<b>Behavioral Simulation Model Options</b>			
<input type="checkbox"/> Disable Collision Warnings	<input type="checkbox"/> Disable Out of Range Warnings		

# Changes on IFetch continued



```

module programrom (
    // Program ROM Pinouts
    input          rom_clk_i,      // ROM clock
    input  [13:0]  rom_adr_i,      // From IFetch
    output [31:0]  Instruction_o,  // To IFetch
    // UART Programmer Pinouts
    input          upg_rst_i,      // UPG reset (Active High)
    input          upg_clk_i,      // UPG clock (10MHz)
    input          upg_wen_i,      // UPG write enable
    input [13:0]   upg_adr_i,      // UPG write address
    input [31:0]   upg_dat_i,      // UPG write data
    input          upg_done_i      // 1 if program finished
);

```

//if kickOff is 1 means CPU work on normal mode, otherwise CPU work on Uart communication mode

```

wire kickOff = upg_rst_i | (~upg_rst_i & upg_done_i);

```

```

prgrom instmem (
    .clka (kickOff ? rom_clk_i : upg_clk_i),
    .wea (kickOff ? 1'b0 : upg_wen_i),
    .addra (kickOff ? rom_adr_i : upg_adr_i),
    .dina (kickOff ? 32'h00000000 : upg_dat_i),
    .douta (Instruction_o)
);
endmodule

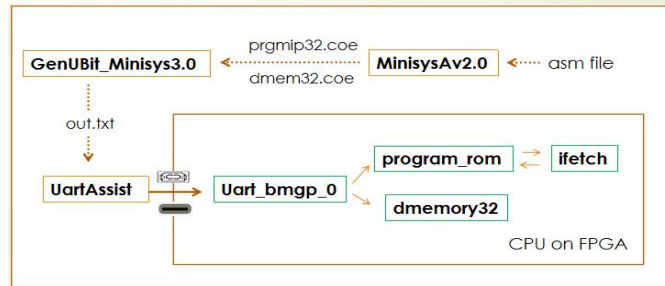
```

# Tips: Generate the Data For Uart Port

- **Step1: Using “MinisysAv2.0” to assemble the asm file and generate the coe files(prgmip32.coe and dmem32.coe)**
- **Step2: Using “GenUBit\_Minisys3.0” to merge the coe files(prgmip32.coe and dmem32.coe) into one file “out.txt”**

➤ Tips on Step2:

put “prgmips32.coe” and “dmem32.coe” into the same directory with “UARTCoe\_v3.0” and “GenUBit\_Minisys3.0”, or you will need to make some modification on GenUBit\_Minisys3.0



LENOVO (D:) > UartTools

名称

- dmem32.coe
- GenUBit\_Minisys3.0
- prgmip32.coe
- UARTCoe\_v3.0

```
d:\UartTools>GenUBit_Minisys3.0.bat
2 files are read successfully
Hexadecimal file(s) detected.
Done.
```

“GenUBit\_Minisys3.0”, “UartAssist” and “MinisysAv2.0” could be found in the “Uart tools” of sakai site:

<https://sakai.sustech.edu.cn/portal/site/d50211ea-1586-4344-9d92-a6c42eb7f4e0/tool/b47e4c13-4220-4f61-b881-a211abee2a96?panel=Main>

## Tips: Using “UartAssist”

- **Step1: Connect** the **Computer** which runs “UartAssist” with **Minisysboard** on which your designed CPU has already been programmed on its FPGA chip.
- **Step2:** Double click on “**UartAssist**” to open it
- **Step3: Set** the items in “串口设置” as the settings of screen snap on the right hand, then click on “**打开**”
  - **NOTICE:** “**串口号**” **could be an serial port other than “COM4”**, which is up to your Computer. The port which you choose here and then click on “**打开**” hasn't report error is the right port.



“GenUBit\_Minisys3.0”, “UartAssist” and “MinisysAv2.0” could be found in the “Uart tools” of sakai site:  
<https://sakai.sustech.edu.cn/portal/site/d50211ea-1586-4344-9d92-a6c42eb7f4e0/tool/b47e4c13-4220-4f61-b881-a211abee2a96?panel=Main>

# Tips: Using “UartAssist” continued

- **Step4: Make sure the CPU on FPGA works on uart communication mode.**
- **Step5: Set** the items in “发送区设置” as the screen snap on the right hand. **Click on** “启用文件数据源”, find the data file which is to be transformed by uart port to FPGA chip.
- **Step6: Wait until a notice info “Program done!” has appeared in the “串口数据接收” window as bellow screen snap.**

