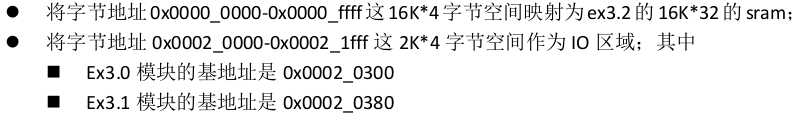
标题：ex3.3实验报告

日期：2023年8月30日

作者：chms

练习3.3要求：

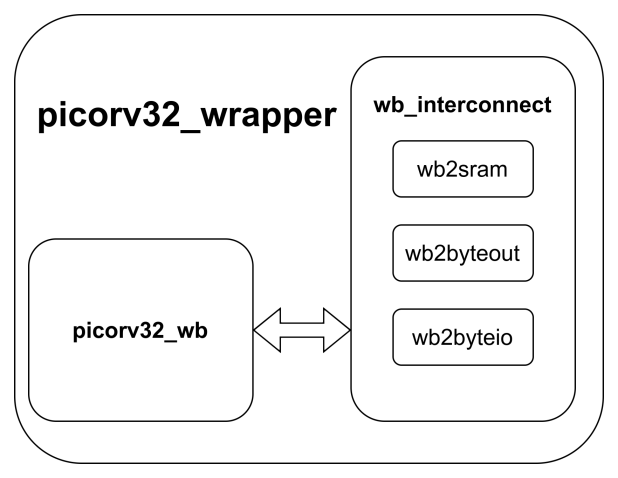
1. 下载并试用picorv32开源核以及工具链。
2. 用picorv32\_wb与wb2sram32、wb2byteout、wb2byteio，完成系统连接，完成总线译码等模块和顶层模块、测试模块，地址映射规则为：



1. 学习 Verilog 的系统任务$readmemh，并考虑软硬件如何联调。
2. 在这个系统上跑一个简单的 main()函数，也就是 C 语言一路编译、装载到 sram内，并在 verilog 模拟器上跑起来。

模块名称picorv32\_wrapper

设计模块接口：



1. picorv32\_wb

|  |  |  |  |
| --- | --- | --- | --- |
| **信号名称** | **方向** | **宽度** | **说明** |
| i\_clk、i\_rst\_n | 输入 | 1bit | 时钟和复位（低有效） |
| wbm\_adr\_o | 输出 | 32bit | 地址 |
| wbm\_cyc\_o | 输出 | 1 | 自Wishbone主接口的CYC，高有效 |
| wbm\_stb\_o | 输出 | 1 | 自Wishbone主接口的STB，高有效 |
| wbm\_we\_o | 输出 | 1 | 自Wishbone主接口的写请求，高有效 |
| wbm\_dat\_o | 输出 | 32 | 自Wishbone主接口的写数据。 |
| wbm\_dat\_i | 输入 | 32 | 返回Wishbone主接口的读数据 |
| wbm\_ack\_i | 输入 | 1 | 返回Wishbone主接口的应答信号 |

1. wb\_interconnect

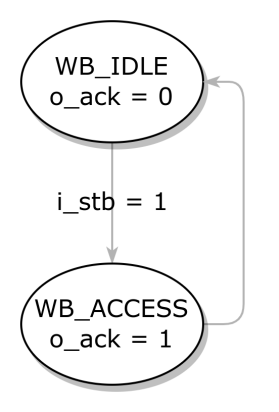
|  |  |  |  |
| --- | --- | --- | --- |
| **信号名称** | **方向** | **宽度** | **说明** |
| i\_clk、i\_rst\_n | 输入 | 1bit | 时钟和复位（低有效） |
| i\_wbm\_adr | 输入 | 32bit | 地址 |
| i\_wbm\_cyc | 输入 | 1 | 自Wishbone主接口的CYC，高有效 |
| i\_wbm\_stb | 输入 | 1 | 自Wishbone主接口的STB，高有效 |
| i\_wbm\_we | 输入 | 1 | 自Wishbone主接口的写请求，高有效 |
| i\_wbm\_dat | 输入 | 32 | 自Wishbone主接口的写数据。 |
| o\_wbm\_dat | 输出 | 32 | 返回Wishbone主接口的读数据 |
| o\_wbm\_ack | 输出 | 1 | 返回Wishbone主接口的应答信号 |

设计：

输入i\_stb以及当前状态。输出o\_ack。转移表：

|  |  |  |  |
| --- | --- | --- | --- |
| **当前状态** | **输出** | **说明** | **输入和转移状态** |
| WB\_IDLE | 0 | 空闲，稳定态，也是复位状态 | 0，无请求，WB\_IDLE |
| 1，读/写请求，WB\_ACK |
| WB\_ACK | 1 | 产生o\_ack脉冲 | WB\_IDLE |
| *default* | *-* |  | *WB\_IDLE* |

状态转移图：



模块的SRAM：

|  |  |  |
| --- | --- | --- |
| **RAM名称** | **宽度** | **说明** |
| sram16kx8a、sram16kx8b、sram16kx8c、sram16kx8d | 16kx32 | 在时钟上升沿：  o\_ack高电平时，读出相应地址的数据，或更新为i\_dat的值。  o\_ack低电平时，保持原值 |

1. memfile拆分脚本

存储模块由4个16kx8 sram组成，故需要将memfile拆分成4份读取。

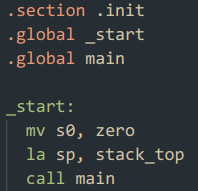
测试：

Timescale：采用1ns/10ps

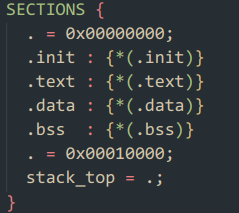
Define：无

Testbench采用10MHz时钟，激励在第一个时钟上升沿后给出。

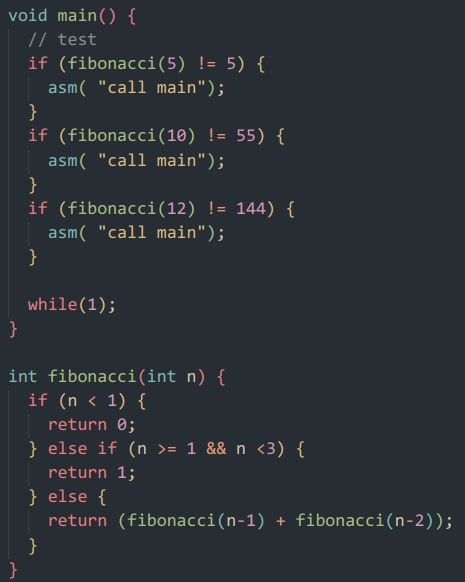
1. 启动汇编



1. 链接脚本



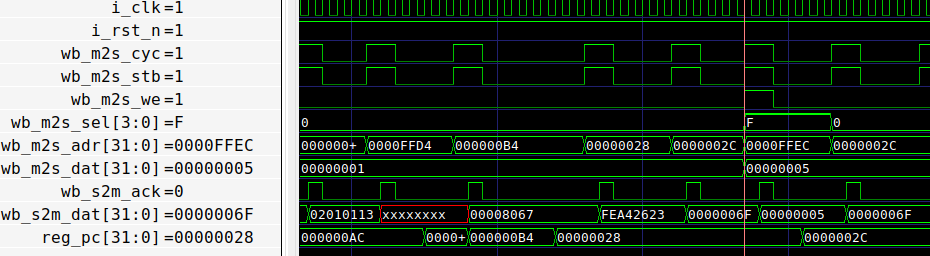
1. 测试程序



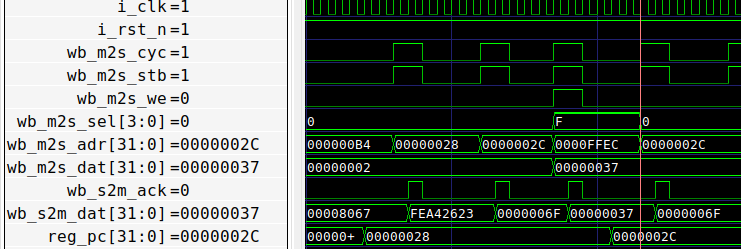
计算斐波那契数列，若结果正确，则进入while(1)循环，否则回到main函数入口。

计算结果：

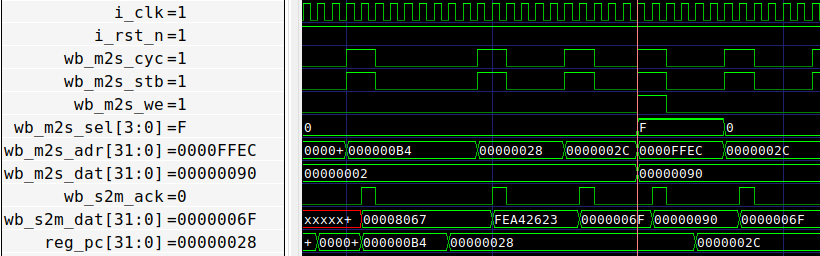
fibonacci(5) = 5



fibonacci(10) = 0x37 = 55



fibonacci(12) = 0x90 = 144



成功进入while(1)循环

