

**《数字VLSI系统的高层次综合》**

**课程设计**

清华大学 集成电路学院

王焕宇 金楚丰

2020211075 2020211138

2021年春季学期

目 录

[目 录 II](#_Toc75724508)

[第1章 硬件架构 1](#_Toc75724509)

[1.1 顶层架构与系统功能定义 1](#_Toc75724510)

[1.2 FIR架构 3](#_Toc75724511)

[1.3 ROOT架构 4](#_Toc75724512)

[1.4 附属系统结构 6](#_Toc75724513)

[1.4.1 RTC (Real Time Clock)模块 6](#_Toc75724514)

[1.4.2 译码模块 6](#_Toc75724515)

[第2章 建模与验证 7](#_Toc75724516)

[2.1 系统功能验证 7](#_Toc75724517)

[2.2 时间显示验证 8](#_Toc75724518)

[2.3 MATLAB行为级建模 9](#_Toc75724519)

[2.4 FIR与ROOT的MATLAB验证结果 9](#_Toc75724520)

[2.5 FIR与ROOT的Verilog验证结果 10](#_Toc75724521)

[第3章 综合结果 12](#_Toc75724522)

[第4章 感想 14](#_Toc75724523)

本作中：

金楚丰同学负责了FIR与求根模组的MATLAB建模、Verilog描述与他们的联合验证；

王焕宇同学负责了时间控制、显示转码的Verilog实现，顶层整合、系统联合调试以及DC综合工作。

# 硬件架构

## 顶层架构与系统功能定义

本作的外部引脚定义如图 1‑1所示。输入信号中，initial\_time\_\*三个信号用于时钟模组设定初值用。PEbar信号是计算模组（FIR和求根）的启动控制信号，data\_\*和B\*这9个信号均为计算模组的数值输入信号。上述这些讯号均来自实验指导书中的Intel 8051。

输出信号中h\*\_7和m\*\_7四个信号用于驱动4个静态数码管显示时间。FIR\_\*\_7三个信号用于驱动3个静态数码管显示FIR模组的输出。

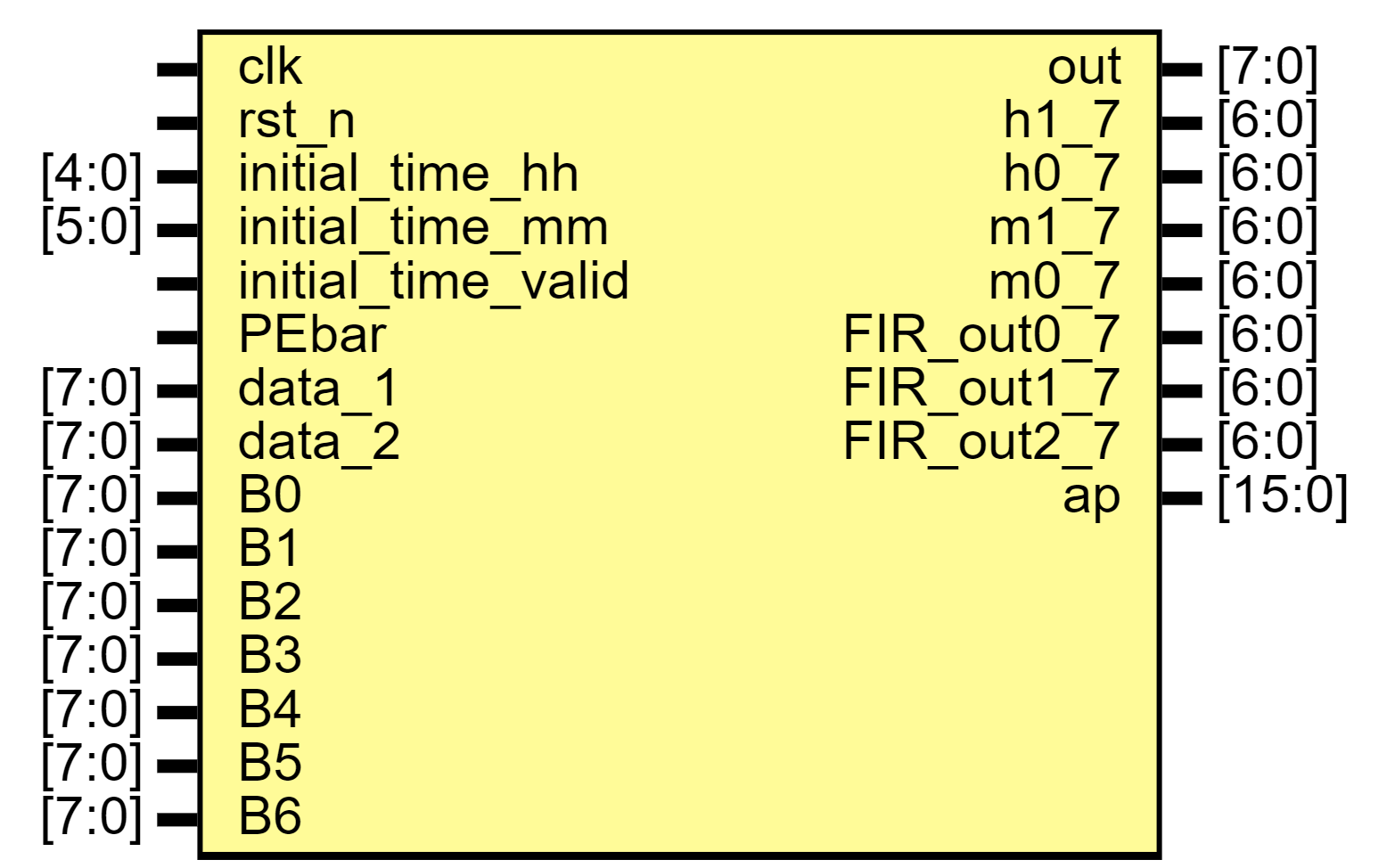


图 1‑1 系统接口定义

系统内部的顶层模组如图 1‑2所示。



图 1‑2 系统抽象图

如图所示，数据通路首先分为两路，一路为FIR滤波，其生成out和其压缩结果ap，一路为时钟，其产生小时和分钟信号，两个模块的输出经过与显示相关的处理（BCD与显示译码）后按照规定的格式输出。

其中，数据输入data\_\*的小数放在第5位右边，最小精度0.125，最大值31.875。数据输入B\*的小数放在第2位右边，最小精度0.015625，最大值3.984375。最小值均为0。数据输出out为整数，最小精度为1，最大值255。

图 1‑3是Vivado出具的系统框图。

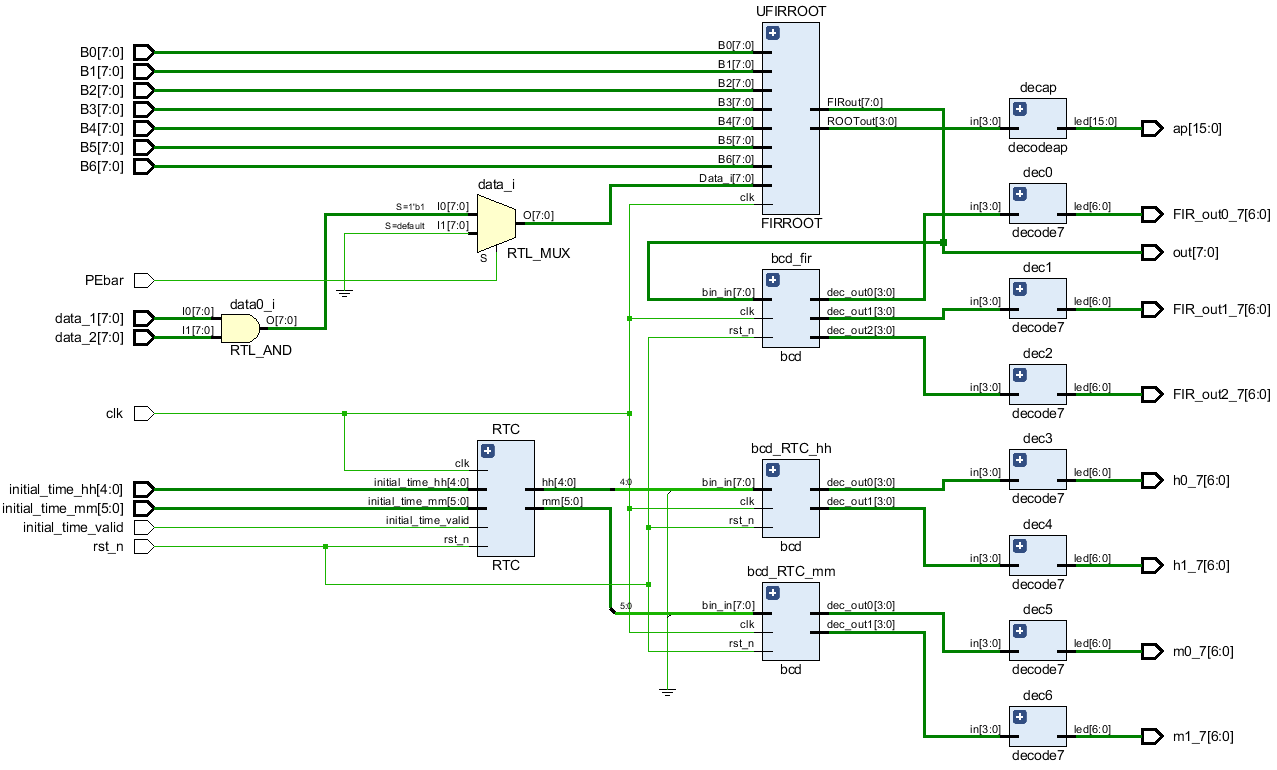


图 1‑3 Module级框图（由vivado生成）

## FIR架构

根据Z域转移函数公式：

逆变换得到时域转移函数：

这是一类在我们目前的《高层次综合》课上没有触及的类型。我们课堂上所探讨的所有数字算法对数据的输入时刻没有要求，是一种类似组合逻辑电路输入的处理方式。我们从算子调度开始的一系列操作可以说是从0开始分割时钟周期、根据数据依赖建立时序关系的。

但是FIR的输入有严格的时序约束，如果仍用控制步的概念描述时钟周期，那么硬件模块至少需要每个控制步处理平均一个输入数据。

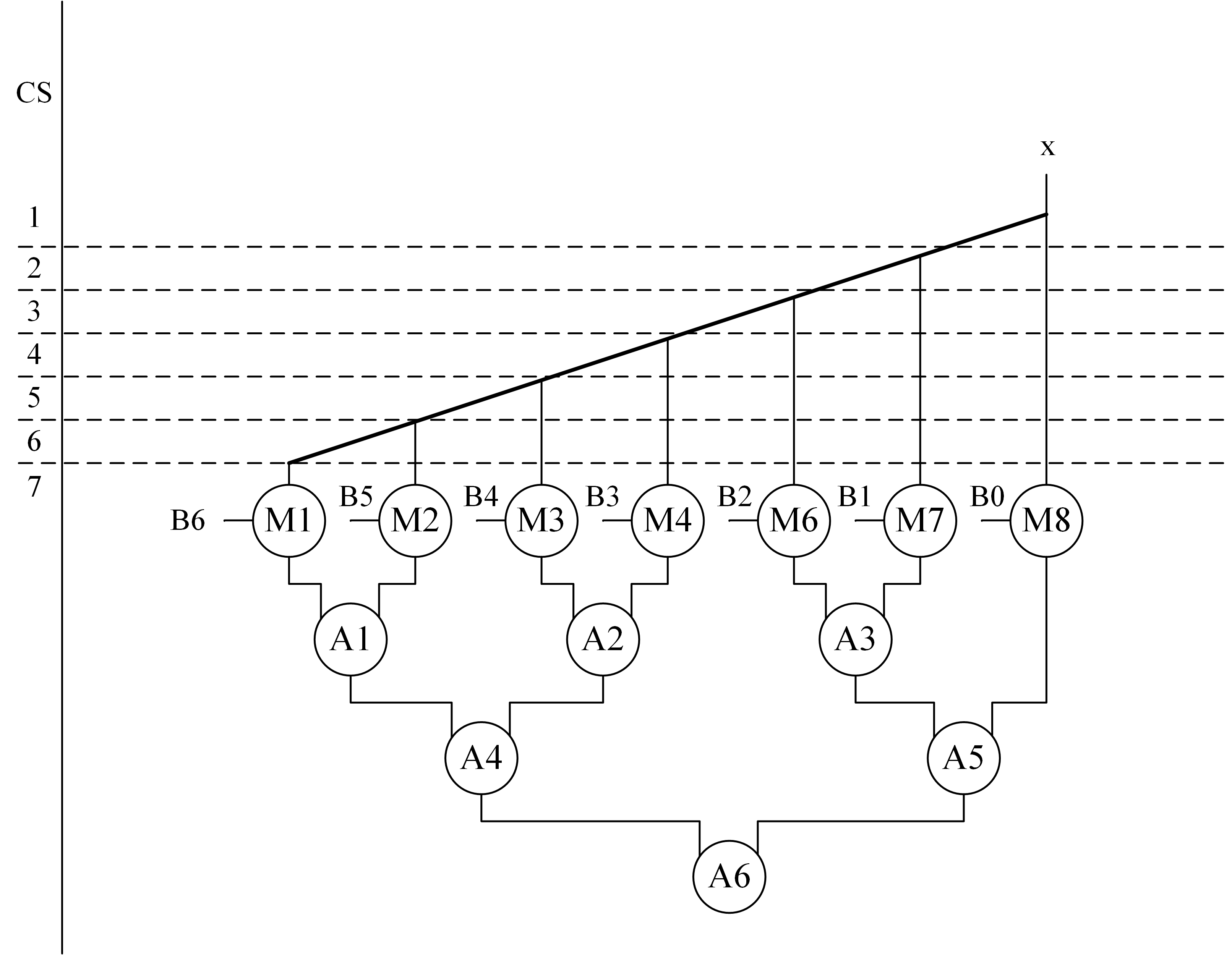


图 1‑4 HLS算子调度图

算法中总共涉及乘法和加法两种运算。假设乘法器资源在同一组数据（比如B[0:6]和x[0:6]的FIR）的计算中在不同控制步被复用，乘法器在第一个周期计算B[0]与x[0]的计算，第二个周期计算B[1]与x[1]的计算，依次类推；下一组数据（B[0:6]和x[1:7]的FIR）的计算中，这组数据的第一个周期和上一组数据的第二个周期重合，这组数据中第一个周期的计算是B[0]与x[1]相乘，所以如果乘法器资源在同一组数据内复用的话，将无法及时处理后续数据组。

用算子调度的概念来说，就是同时需要进行的乘法数是7。如果用算子调度的方式进行操作，这7个乘法，连同7个积之间的加法，应该发生在同一个控制步内。使用高层次综合得到的算子调度图如图 1‑1。图中所有运算单元都分布在控制步7，输入x的斜线经历延迟单元，竖线不经历延迟单元，所以M1-M7所得到的是不同时钟周期x的输入。A6输出即为FIR的滤波结果。关键路径长度为1个乘法器加上3个加法器。



图 1‑5 流水线重定时算子调度图

刘雷波老师的《VLSI数字信号处理》课中讲解过一种割集流水线重定时方法，将算子调度修改为图 1‑2。这种修改之后的关键路径缩短为1个乘法器加上1一个加法器。

本作最终采用了图 1‑2方案以获取最佳性能。

## ROOT架构

求根模组也有两种实现方案，一种采取课堂一元二次方程组例子中的牛顿迭代法，另一种采取多级比较器的方式通过数据选择器得到最终的结果。

值得注意的是，无论选择哪种方案，由于FIR模组已经确保了处理能力与采样输入速度匹配，也就是吞吐率达到一个周期处理完一个数据，求根模组必须也要到相同的吞吐率。

在实际设计中，我们首先采取了迭代次数为3的牛顿迭代法进行部署。这种方法的算子分配图如图 1‑3。

这种方法在乘法和加法之外，还需要使用到左移、除法、右移三种运算单元。3次迭代的总控制步数为6，计算初值也需要两个控制步，所以总控制步数量为8。

在MATLAB建模阶段我们对这种算法的精度进行了定性分析。在ROOT模块的输入值超过我们所划定的256时，这种算法的精度会下降。但是在256之内时可以得到一个精确到十进制个位的计算结果。

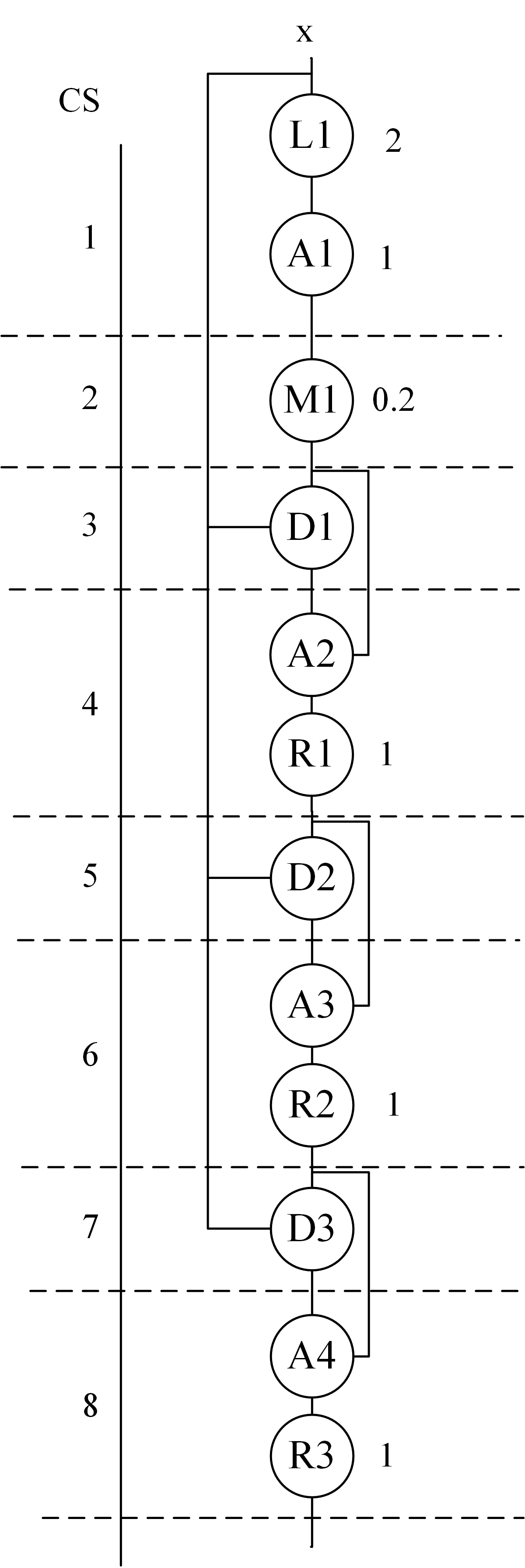


图 1‑6 牛顿迭代法算子调度图



图 1‑7 牛顿迭代法MATLAB建模与验证

不过我们随后考察了另一种算法，也就是比较器方案。比较器方案的算子调度图见图 1‑5。

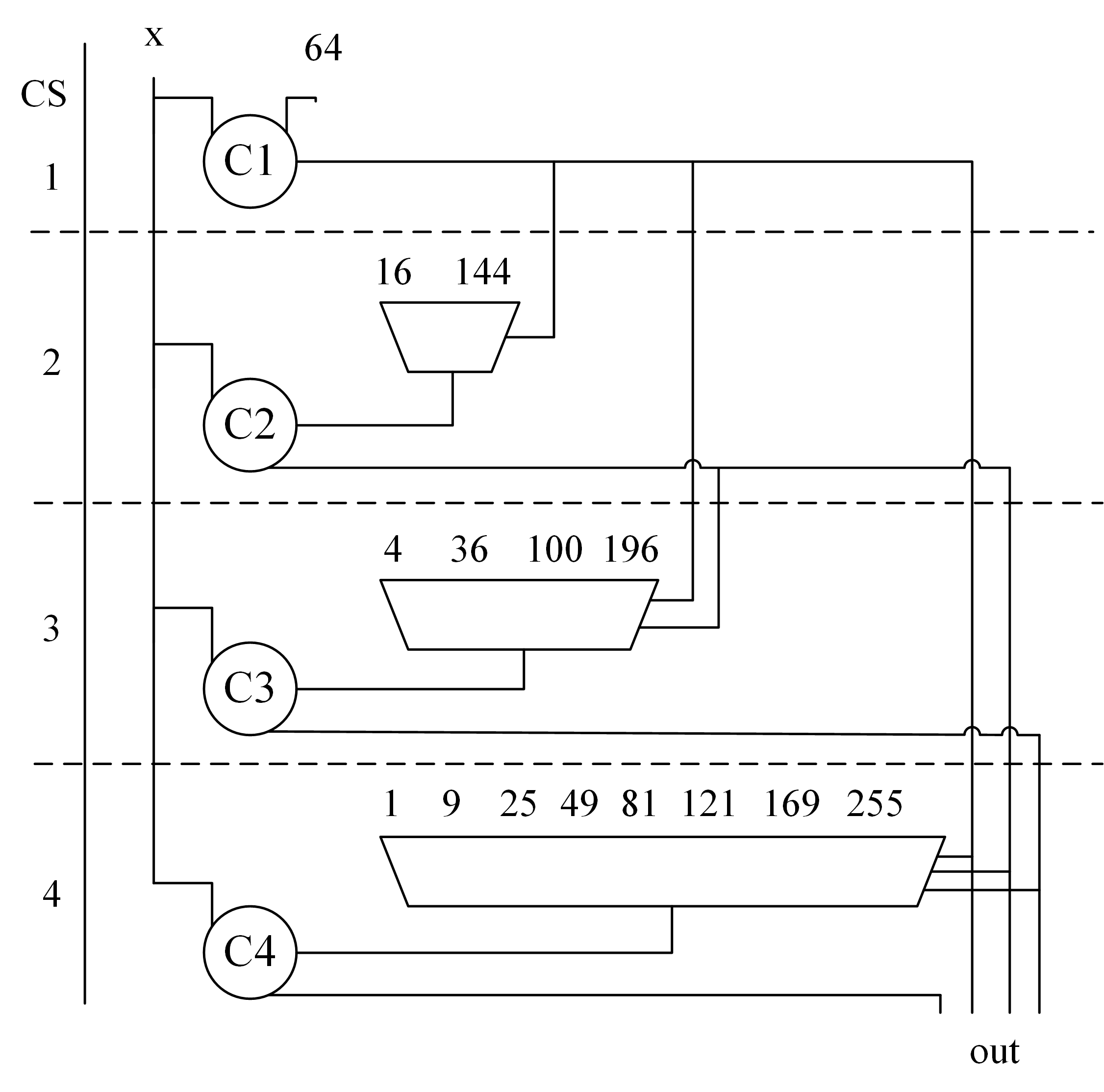


图 1‑8 比较器算子调度图

这种方案在256内的计算精度是与牛顿迭代法相同，也是精确到十进制个位。但是硬件开销和延迟都优于牛顿迭代法。这种算法只需要4个控制步就可以完成运算，关键路径等于一个8选1 MUX加一个比较器。

最终的输出来自四个比较器1 bit输出的合并。

## 附属系统结构

除FIR滤波与ROOT功能外，本系统其它主要模块及其设计理念简要介绍如下：

### RTC (Real Time Clock)模块

RTC模块按照分频-计数-触发思想设计，计数器依据芯片实际时钟进行累加，在达到设定好的分频系数后，触发记满标志，得到真实的分信号。分信号按照0-59的顺序进行循环，并在满59时出发时信号加一。时信号在满23后归零。

RTC模块使用valid标志从上层模块获得时间更新，由于由内部、外部两种修改内部计时器的路径，实现此功能时需十分仔细，注意避免产生锁存器。

### 译码模块

译码模块包括BCD译码、AP的4-16译码、BCD译码后的7段译码。

BCD译码将2进制数转为按照10进制进位的、数个位于0-9区间的4bit二进制数。

BCD译码可以由单个时钟周期获得，速度较慢，也可以通过多个时钟周期获得。考虑到该系统主要的延迟来源于FIR与ROOT的流水周期，故BCD只是流水的一级，自身速度预计并非敏感条件，所以使用单周期方案，配合一个输入寄存器形成一级流水实现。

4-16译码和7段译码较为简单，仅需注意本设计要求的7段译码并非典型情况，需按照设计要求里的液晶屏管脚分布进行设计。考虑到其结构简单，故位单独作为流水线的一级，而是直接以组合逻辑实现，以最大限度减少该系统的latency。

# 建模与验证

## 系统功能验证

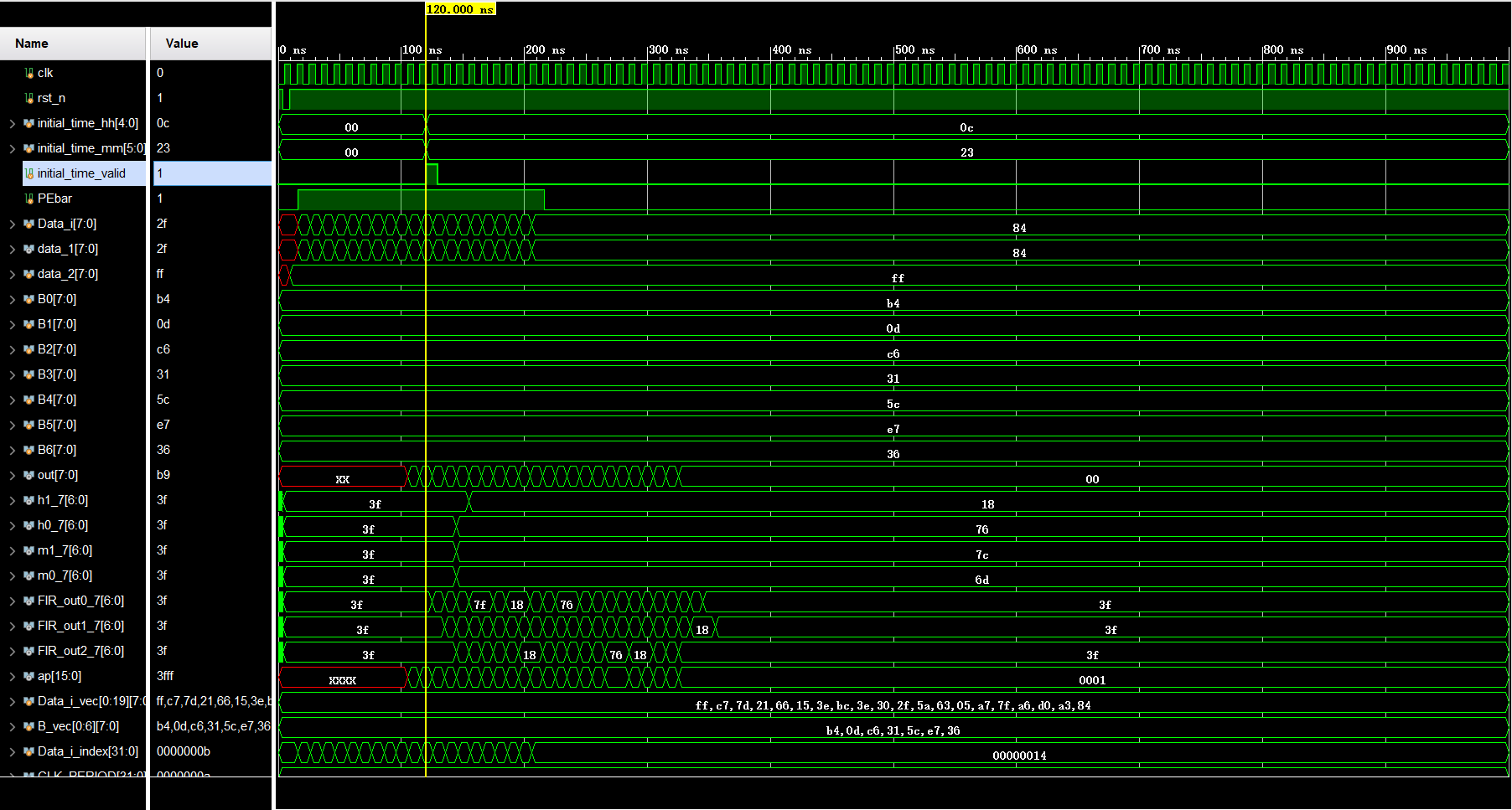


图 2‑1 总体运行图

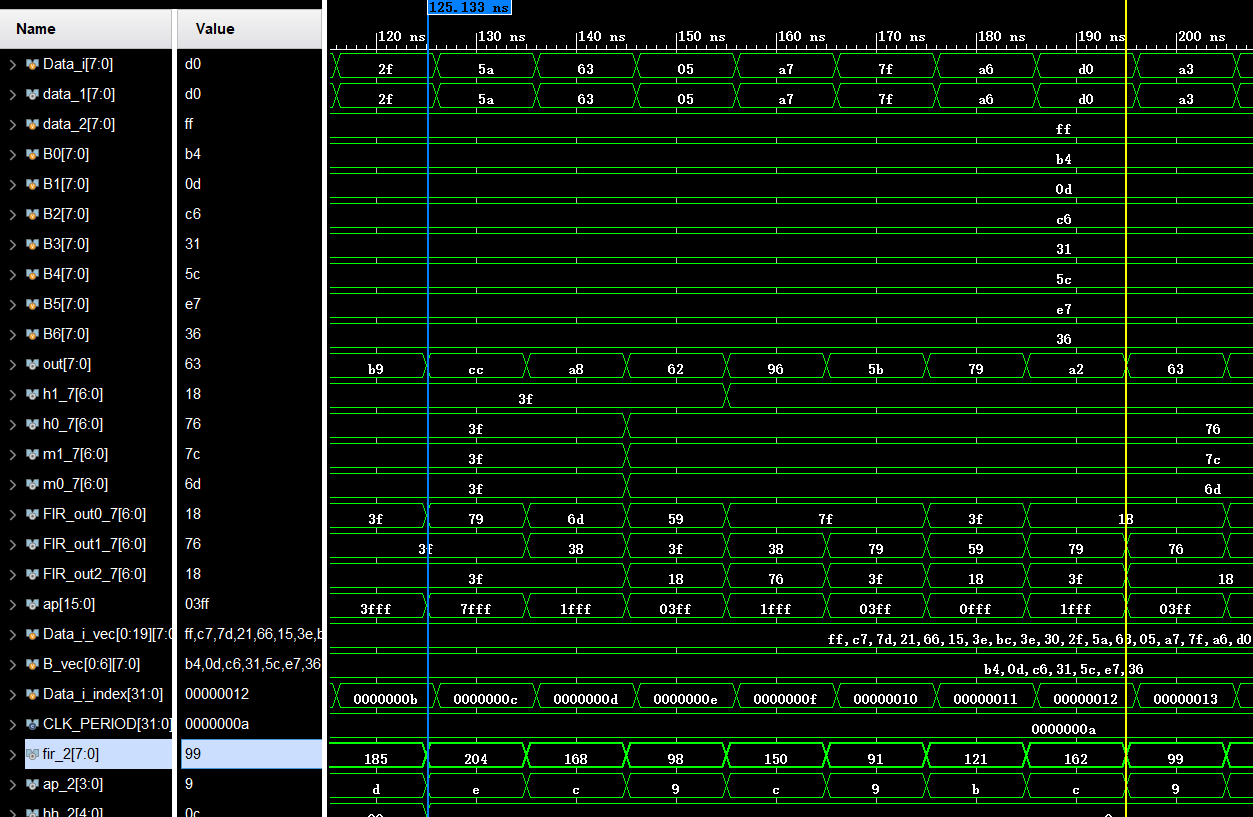


图 2‑2 输入→FIR

该步的正确性在后文Matlab建模分析中进行论证，此处可见输出与预期相符。

比较内部未译码信号与最终显示信号，可证明显示结果完全正确。

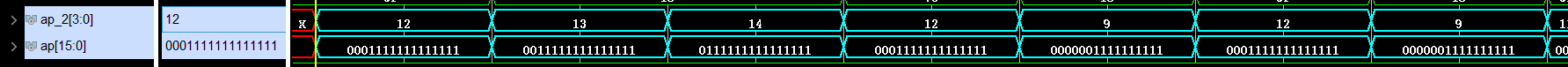


图 2‑4 ap的4-16译码

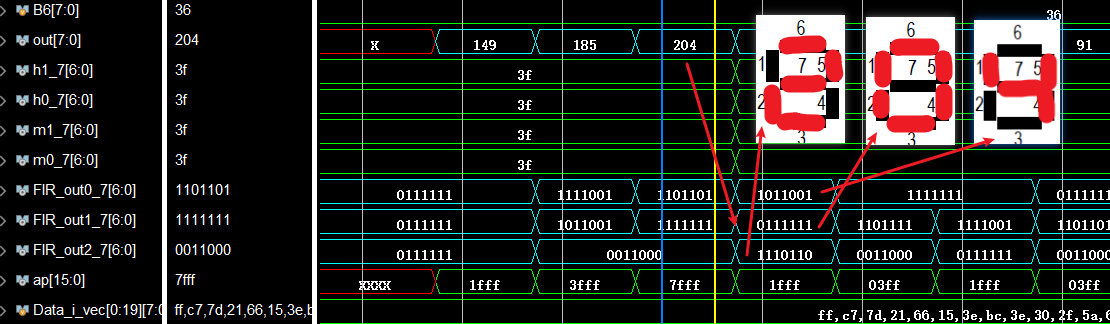


图 2‑3 FIR→显示

## 时间显示验证

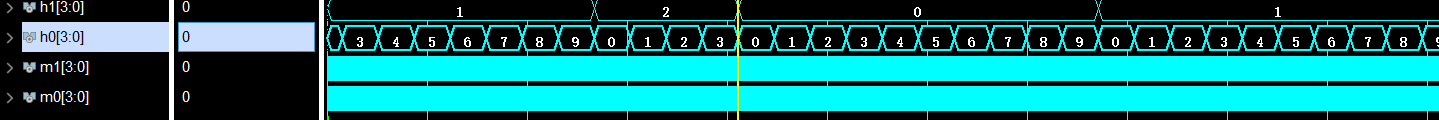


图 2‑7 跨24小时

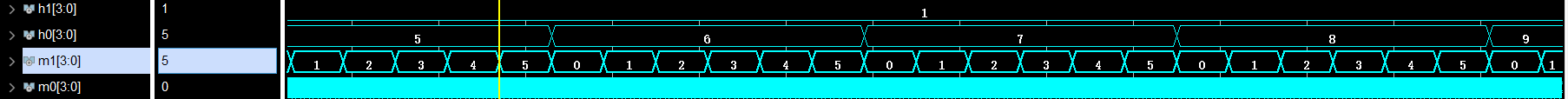


图 2‑6 分钟到小时进位

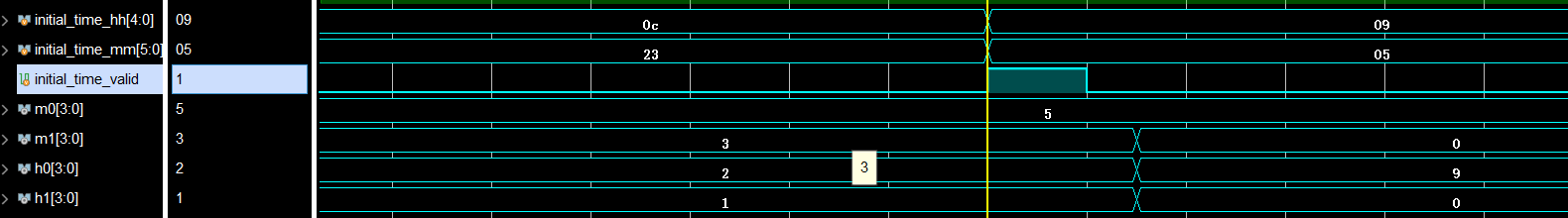


图 2‑5 任意时刻修改时间

时间的初始化、累加以及任意时刻修改能力均正确。

## MATLAB行为级建模

本次实验中，在完成Verilog HDL描述的同时，也使用MATLAB建立了相同硬件的行为级描述。MATLAB描述无法反映信号的时序表现，但是所有最终结果和中间结果保持了与硬件相同的精度。

同时，使用MATLAB撰写了激励生成器，生成的激励将保存在文本文档里。MATLAB和Verilog先后读入相同的激励组，对比MATLAB模型的结算结果和Verilog仿真结果以确认主要模组的功能正确性。

激励生成器会生成一个长度为7的B向量和一个长度为20的x向量。其中B作为FIR的乘法器参数，x作为FIR的输入。对于MATLAB模型来说，每次从20个元素中取连续的7个元素作为模组输入，用来拟真一次硬件的计算过程。对于Verilog的FIR模组来说，每个周期读入一个新的x向量元素。

## FIR与ROOT的MATLAB验证结果

我们选择了一组没有溢出的激励组作为MATLAB和Verilog联合仿真的比较对象。

这组向量的二进制表示如下。其中B\_vec\_bin是B向量的数值，Data\_i\_vec\_bin是x向量的数值。

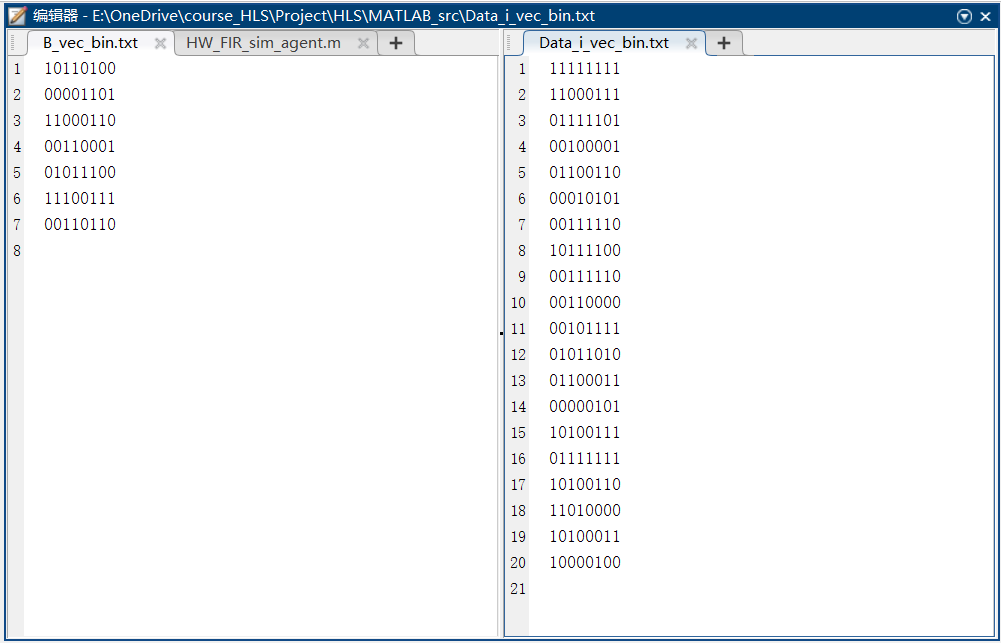


图 2‑8 MATLAB激励生成器生成的激励

这组激励在MATLAB中仿真得到的结果见图 2‑2。

## FIR与ROOT的Verilog验证结果

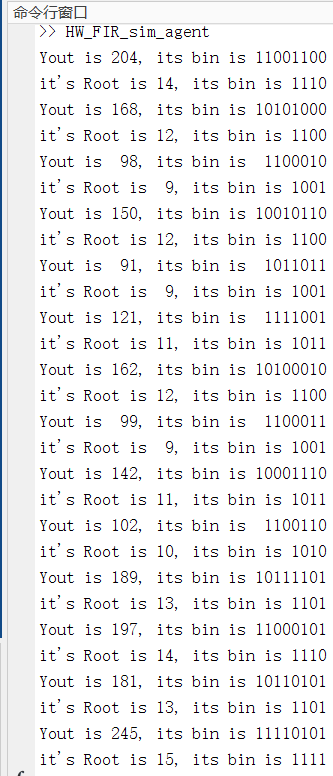


图 2‑9 MATLAB模型运行结果

在Vivado仿真器中，进行了FIR与ROOT模组的验证工作。原本，ROOT模组的输出比它对应的FIR数值要迟4个时钟周期，也就是ROOT模组自身的延迟数。所以我们对FIR的结果也延迟了4个时钟周期，用于对齐。

Vivado对2.3节中图 2‑1相同激励的仿真结果见图 2‑3。图中只有游标时刻之前的FIRout和ROOTout数据是有效的，其后的数据无效。其中数据部分放大节选在图 2‑4，可以看到图中的数据与MATLAB仿真结果吻合。

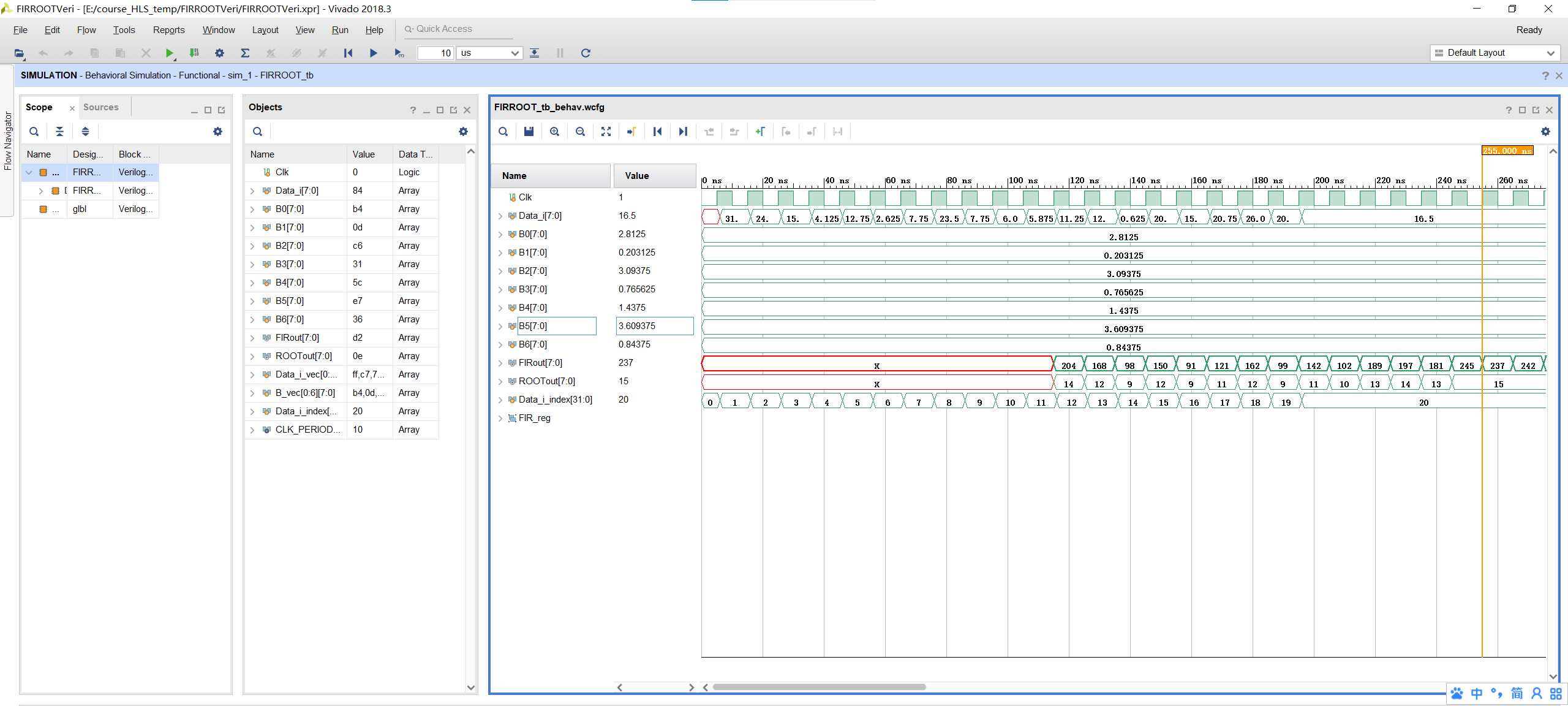


图 2‑10 Vivado仿真结果

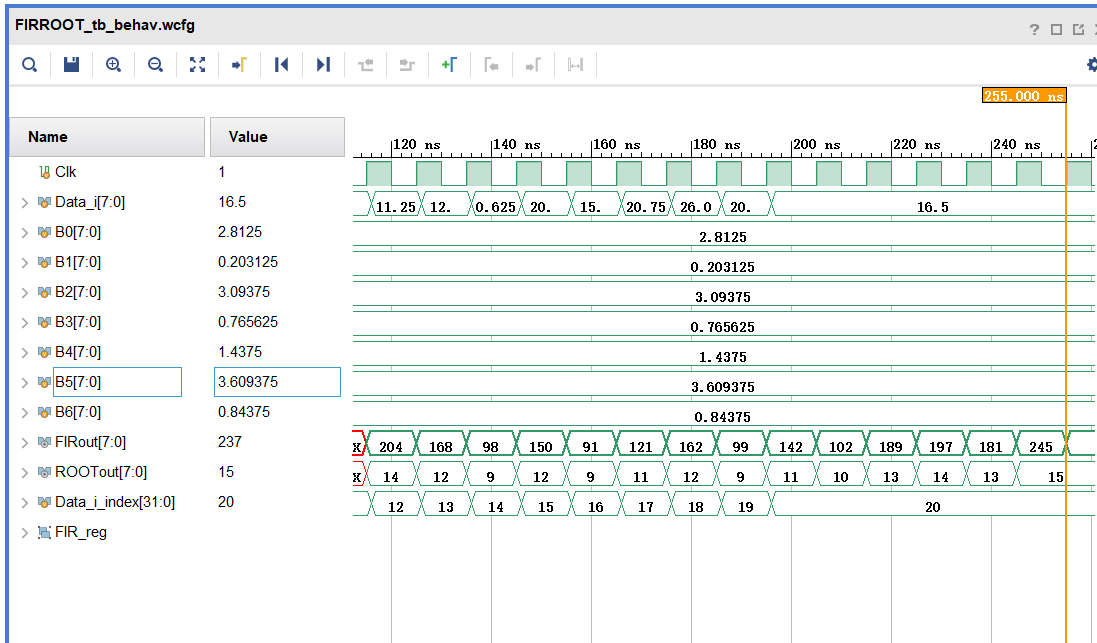


图 2‑11 Vivado仿真结果放大

# 综合结果

DC综合使用了UMC0.18um\_slow工艺，设置基本参数如下：

|  |  |  |  |
| --- | --- | --- | --- |
| CLK\_UNCERTAINTY | MAX\_AREA | INPUT\_DELAY | OUTPUT\_DELAY |
| 2 | 0 | 16 | 16 |

在此约束下，我们尝试可达到的最小时钟周期，最终确定为22ns。几个测试点的数据如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试点(时钟周期约束) | Timing  (min slack) | cell | area  (cell area) | Power  (mW) |
| 40 | 10.78 | 160 | 45664.820349 | 0.6597 |
| 30 | 0.67 | 160 | 50547.975641 | 0.8227 |
| 22 | 0.00 | 2022 | 54449.842566 | 1.2421 |
| 4\* | 0.00 | 2219 | 59266.469330 | 8.0086 |

\*去除所有约束，仅保留时间约束，可达到的最小时钟周期。

报告截图：（CLK\_PERIOD =22ns）

|  |  |
| --- | --- |
| Timing |  |
| Power |  |
| Cell |  |
| Area |  |

# 感想

本次课程设计两位组员最大的收获在于使用了GitHub进行远程协作和版本管理。

这是我们的repo的URL，<https://github.com/HLS-UESTHU/HLS>。GitHub的使用极大减轻了我们文件交互的管理压力，并也一定程度上增加了我们产出的积极性。

本文1.3中撰写关于ROOT模组方案的探讨时，已经弃用的旧模型源文件是从历史版本中回溯出来用的。

想必我们以后的职业生涯中还会有很多很多需要使用版本管理软件进行版本管理和多人协作的时候。此时的锻炼会派上用场。