Xilinx FPGA有两种片上RAM的实现方式，分别是Block RAM和Distributed RAM。前者使用片上固定的BRAM资源（SRAM块）来实现RAM，这些BRAM资源是预留在芯片上的专用RAM单元，集中而紧凑，具有相对较大的容量，且内部具有优化过的固定布线，块内延迟很小，但同一个BRAM资源是不能拆成多个RAM实例使用的，具体使用中存在BRAM资源浪费问题。后者是使用片上零碎的可编程逻辑块（CLB）实现的，每个CLB实现位锁存或译码选择逻辑，通过小的CLB互联，实现完整RAM功能，所以分布式RAM的内部延迟取决于CLB布局和互联布线。

Block Memory Generator是用来产生Block RAM实现的RAM结构的基本IP核，它在配置时可以配置为单端口（读写共用一组地址线，数据线分为写入数据线和读出数据线），简化双端口（一种减少结构冒险的设计，一个端口只用于读，另一个端口只用于写，两个端口都有自己的地址线，写端口还有数据输入线）或者双端口模式（两个端口可用于独立的读和写，我自己没用过，可能会用在多处理器设计上）。IP核的控制信号是可选的。我记得这个IP核设置里面有个读优先还是写优先的选项需要注意下，因为可能出现在一个周期里读的同时使能写信号向读的那个地址写入了新数据的情况，读优先保证持续输出原来的数据直到本周期结束，而写优先会在写入完成后立即将输出数据换成写入的新数据。我们的五级流水线设计里面的RAM不会在一个周期里同时读和写，相邻的读写一定会处于两个不同周期，所以不用考虑这个问题，先写的数据后读一定能读到最新数据。

对于一个手工使用reg数组实现的RAM结构（也有可能是一组寄存器，取决于它在你设计中的地位），Vivido会自行选择使用Block RAM或者Distributed RAM实现（貌似reg的数组长度\*向量长度比较小的时候，都会选择使用Distributed RAM实现）。不过我们可以通过在reg关键字前添加(\* RAM\_STYLE="block" \*)或者(\* RAM\_STYLE="distributed" \*)来手动指定其实现方式。

对于延迟问题，个人感觉在时钟频率不太高（100MHz上下），电路不太复杂的情况下，时序逻辑之间的组合逻辑的传播延迟决定了电路总的延迟（Block RAM和锁存器这些基本逻辑中的延迟可以忽略，也不用考虑Setup Time和Hold Time这些东西，它们远小于时钟周期）。如果时序上出问题，上板发现丢数据或者数据不一致，应该重点检查和思考“有没有给RegFile的读电路、ALU电路这些内部结构复杂的组合逻辑留足够的信号传播时间”，让组合逻辑有足够的时间完成选择和计算，直到数据稳定。比如给RegFile送去addr的瞬间从输出端只能取到上一次读出的数据而不是新数据，在寄存器写操作时钟边沿到来的同时做一次对写入新数据的读操作并取走也是不合适的。一种常见的做法是利用半个时钟周期的时间去等组合逻辑得到正确稳定的输出，把组合逻辑电路的送输入和取输出分配到全局时钟信号的两个不同边沿上。实在无法满足延迟上的要求，就需要分频降低时钟频率或者增加多级缓冲。

对于仿真，我的建议是在出现问题时尽量选择时序仿真而不是行为仿真做精细调试。在启动仿真时，Vivado会让我们选仿真的类型：

1. 行为仿真（使用HDL源码，不考虑延迟）

2. 综合后功能仿真（使用网表，不考虑延迟）

3. 综合后时序仿真（使用网表，考虑延迟）

4. 实现后功能仿真（使用位流，不考虑延迟）

5. 实现后时序仿真（使用位流，考虑延迟）

其中1最常用，启动仿真很快，用于查找代码语法和逻辑错误，但完全没有对延迟的建模。

建议使用3来查找潜在的时序问题，理由是网表有是具体的电路结构，是对时序做较为完整模拟的基础，且综合操作一般不会太久，但实现操作可能会花费大量时间运行多次迭代的布线算法，5虽然考虑了走线的延迟，最接近上板情况，但花费时间太久，不如上板，只适合于手头没板或硬件验证成本太高（如流片）的情况。

对于硬件实验，还有两个额外的小建议：

1. 重视Warning，搞清楚它们的含义。老师给的代码模板有很多不规范的地方，导致大量Warning出现。很多同学做实验久了就会下意识无视Warning（反正最后下板能用）。但实际上EDA工具给的Warning的严重程度比软件Compiler要高得多，其中不乏对模块漏连、错连和时序上的严重问题的提示。对于无关紧要的Warning（对于它们的含义，可以去Xilinx的论坛上看一看其他人的讨论），好像是可以右键设为忽略的。切忌因为Warning太多就不看Warning。

2. 仿真时额外检查一下仿真目录下的coe文件是不是最新的（和源码目录下的一致），防止因软件没有更新仿真用到的ROM和RAM的coe造成分析波形时感觉有问题的假象。