一、 研究背景

1.1 研究意义

回顾集成电路的发展历史，“功耗”一直是设计者关心的问题。随着纳米级工艺的逐渐减小，这一问题在设计中面临这更多残酷考验。以往，更多的优化设计中心主要集中在对电路芯片的运行速度和面积上，而功耗由则放在较次要的位置上。但随着许多便携式的电子用品（如 MP5、手机以及笔记本电脑）和医疗器件（如助听器、心脏起搏器）的应用发展，以及集成电路自身规模越来越大，使得功耗问题变得日益突出。2005 年国家自然科学基金委员会发布的重大研究计划项目申请指南中，就已把 “高性能、低功耗电路与系统”列为《半导体集成化芯片系统基础研究》的重要研究内容。

随着集成电路规模和复杂度的巨大发展的同时，设计要求也在发生着变化。上世纪 80 年代到 90 年代，在集成电路设计过程中，面积和速度是首要的考虑指标。但随着设计技术的发展和特征尺寸的缩小，芯片的规模、复杂度和工作频率不断提高，引起芯片的功耗急剧增大。以 CPU 为例，在近年来，每 2-3 年更新一代，每一代新产品的主频和晶体管密度增加约一倍。如 Intel 公司于 2000 年推出 Pentium IV 处理器的晶体管数量和功耗分别是 Pentium III 的 1.5 倍和 2 倍。研究表明，功耗的增大，会引起芯片温度的上升，这些因素一方面导致器件的可靠性降低以及由此引起的芯片稳定性下降，给芯片特别是对可靠性要求极高的军用品带来严峻的挑战。另一方面，这些因素也将导致对芯片的封装材料的要求越来越高，成本越来越昂贵等问题。

乘法器由庞大的加法器阵列组成。作为语音与图像等数字信号处理模块的核心，同时也是微处理器、RISC、DSP 和 FIR 数字滤波器等各种电路中的不可或缺的算术运算单元。乘法器的性能好坏对整个芯片都有着十分重要的影响。从电路设计层面上，数字集成电路由组合逻辑、时序逻辑和存储器等基本电路构成，而乘法器就是数字电路中最重要的组合逻辑电路。其所消耗的功率在整个电路中占有较大的比重。近些年来，随着便携式可移动数字产品市场不断发展壮大，人们对便携可移动式电子设备需求越来越大，对产品功耗和芯片散热的要求也越来越苛刻。低功耗乘法器已嵌入到这些系统中，因此对乘法器进行低功耗设计有着重要的现实需求和意义。

1.2 乘法器种类

（1）移位相加乘法器：乘法器实现的一种基本方式。其大致原理为：从被乘数的最低位开始判断，若为 1，则乘数左移 i（i=0,1...(WIDTH-1)）位后，与上一次和相加；若为 0，则乘数左移 i 位后，以 0 相加。如此循环移位相加，直至被乘数的最高位。

（2）流水线乘法器：一般的快速乘法器通常采用逐位并行的迭代阵列结构，将每个操作数的 N 位都并行地提交给乘法器。但是一般对于 FPGA 来讲，进位的速度快于加法的速度，这种阵列结构并不是最优的。所以可以采用多级流水线的形式，将相邻的两个部分乘积结果再加到最终的输出乘积上，即排成一个二叉树形式的结构。流水线乘法器比串行乘法器的速度快很多很多，在非高速的信号处理中有广泛的应用。

（3）Wallace 树乘法器：在乘法器的设计中采用树形乘法器，可以减少关键路径和所需的加法器单元数目，Wallace 树乘法器就是其中的一种。从数据最密集的地方开始，不断地反复使用全加器、半加器来覆盖“树”。全加器是一个 3 输入 2 输出的器件，因此全加器又称作 3—2 压缩器。通过全加器将树的深度不断缩减，最终缩减为一个深度为 2 的树。最后一级则采用简单的 2 输入加法器组成。

（4）booth 乘法器：布斯(Booth)算法采用相加和相减的操作计算补码数据的乘积，Booth 算法对乘数从低位开始判断，根据后两个数据位的情况决定进行加法、减法还是仅仅进行移位操作。

（5）查找表乘法器：查找表乘法器就是先将乘法的所有可能结果存储起来，然后将两个相乘的数据组合起来作为“地址”找到相应的结果。但是随着乘数位宽的增加，需要存储的结果迅速增加，不利于实现，因此该方式适用于位宽很小的情况。

二、 设计方案

2.1 乘法器原理

对于基本的二进制乘法器而言，其乘法计算过程可以视为两部分组合而成：（1）将被乘数 x 与乘数 y 每位依次对应相乘（相与），得到部分积，如图 2-1 所示。假设为 N 位乘法器，则部分积的个数为N^2。（2）将产生的部分积相加得出相应位置的乘法结果 P。且保留相加产生的进位输出，作为下一位计算过程中加法器的进位输入。如图 2-1 中，部分积 y_1x_0、y_0x_1 相加结果和为 P_1 位，而产生的进位输出则作为下一位计算中 y_2x_0、y_1x_1、y_0x_2，的进位输入。如此传递则可得出每一位的结果。

不同的乘法器方法的不同，则基本上是对上述过程进行了优化。如采用并行结构提升计算速度、使用不同的加法器，对加法过程优化等等。

								y_7	y_6	y_5	y_4	y_3	y_2	y_1	y_0
								x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0
								y_7x_0	y_6x_0	y_5x_0	y_4x_0	y_3x_0	y_2x_0	y_1x_0	y_0x_0
							y_7x_1	y_6x_1	y_5x_1	y_4x_1	y_3x_1	y_2x_1	y_1x_1	y_0x_1	
						y_7x_2	y_6x_2	y_5x_2	y_4x_2	y_3x_2	y_2x_2	y_1x_2	y_0x_2		
					y_7x_3	y_6x_3	y_5x_3	y_4x_3	y_3x_3	y_2x_3	y_1x_3	y_0x_3			
				y_7x_4	y_6x_4	y_5x_4	y_4x_4	y_3x_4	y_2x_4	y_1x_4	y_0x_4				
			y_7x_5	y_6x_5	y_5x_5	y_4x_5	y_3x_5	y_2x_5	y_1x_5	y_0x_5					
		y_7x_6	y_6x_6	y_5x_6	y_4x_6	y_3x_6	y_2x_6	y_1x_6	y_0x_6						
	y_7x_7	y_6x_7	y_5x_7	y_4x_7	y_3x_7	y_2x_7	y_1x_7	y_0x_7							
p_{15}	p_{14}	p_{13}	p_{12}	p_{11}	p_{10}	p_9	p_8	p_7	p_6	p_5	p_4	p_3	p_2	p_1	p_0

图 2-1　8 位乘法器部分积示意图

2.2 低功耗方案

针对低功耗的需求，在传统乘法器设计基础上进行了改进。以达到了减少功耗的效果。主要作了如下结构化的优化：

（1）双通道串行结构设计：

在以往的乘法器优化中，主要考虑因素为提升计算速度和性能。然而为了提升速度采用了过多的并行结构。串行计算的方法即是让结果随时钟一个一个输出。相较而言，减少了很多重复的结构，节省了大量的面积，使得功耗大幅下降。使用了双通道的串行设计，相较一位串行输出的乘法器，将计算速度提升了一倍，同时其将结构利用最大化与传统的双串行/并行乘法器不同，此设计在最小数据锁存的情况下实现了双倍吞吐量。更进一步，这种乘法器的硬件可以简单地通过重复一个单元来扩展。可以组合成所需任何位数乘法器。

（2）门控时钟：

芯片功耗组成中，有高达 40%甚至更多是由时钟树消耗掉。为了减少时钟网络的功耗消耗，最直接的办法就是如果不需要时钟的时候，就把时钟关掉。此种方法即为门控时钟，利用时钟使能信号，来决定是否开启时钟，以减少不必要的功耗。如图 2-2 所示即为锁存门控时钟的示意图。将时钟使能信号通过锁存器后与时钟相与，且使用 CLK 的非作为锁存器控制信号。只有在 CLK 为高的时候，GCLK 才可能会输出高，这样就能消除 EN 带来的毛刺。这是因为 D 锁存器是电平触发，在 CLK=1 时，数据通过 D 锁存器流到了 Q；在 CLK=0 时，Q 保持原来的值不变。

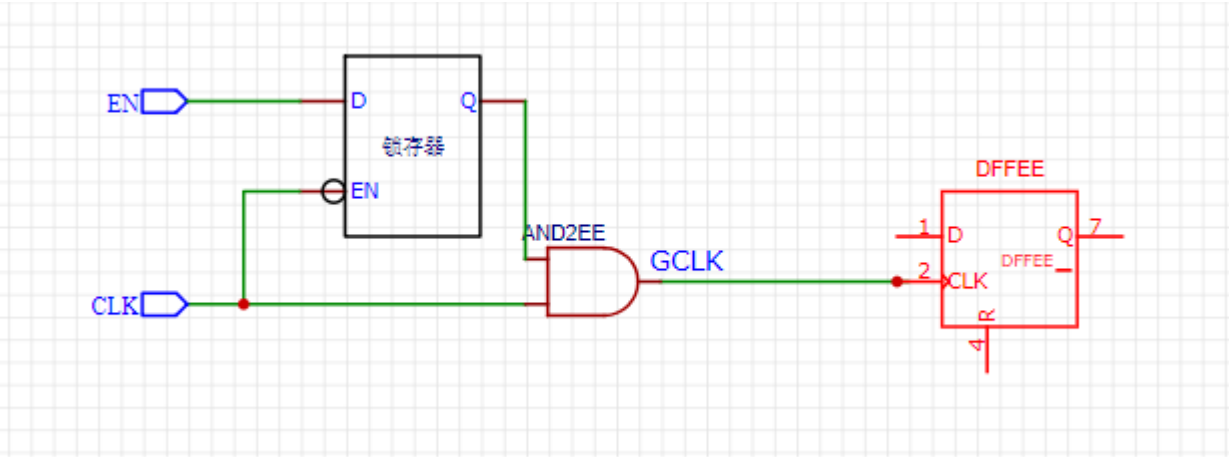

图 2-2　门控时钟示意图

2.3 设计结构

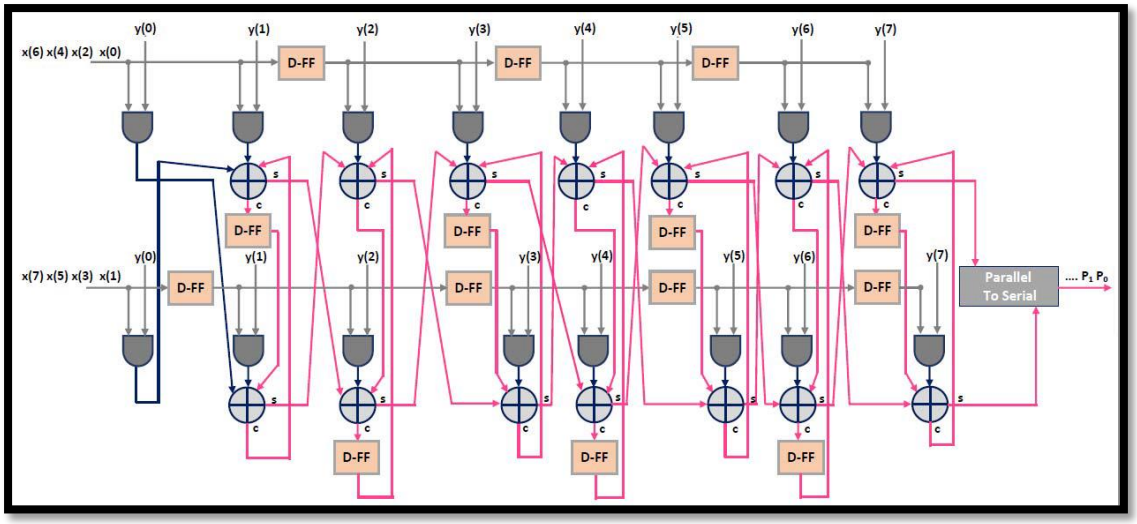

图 2-3　乘法器结构示意图（8 位）

以图 2-3 所示 8 位数的双通道串行乘法器为例。该方案具有简单而统一的结构。x 和 y 分别是串行输入和并行输入，P 为乘法器结果。每个时钟周期处理两

个串行输入位。串行输入的偶数位（……x6、x4、x2、x0）输入至上通道，而奇数位（……x7、x5、x3、x1）在下通道中被处理。两个通道同时处理。从第一个时钟周期开始，生成部分积 PP_0

$$PP_0 = \begin{cases} y_0x_0, & y_1x_0 \\ y_0x_1. \end{cases}$$

将部分积（y_0x_1）与部分积（y_1x_0）相加并传递到输出。此外，部分积（y_0x_0）直接传递到输出。乘积的最低有效位（P_0 和 P_1）在以下等式中同时生成：

$$P_0 = y_0x_0$$
$$P_1 = y_0x_1 + y_1x_0.$$

在下一个时钟周期中，串行输入数据在每个延迟元件之后向右移动一个相位。与是下一周期上下通道分别产生的部分积如下所示：

$$PP_1 = \begin{cases} y_0x_2, & y_1x_2, & y_2x_0, & y_3x_0 \\ y_0x_3, & y_1x_1, & y_2x_1. \end{cases}$$

对于上通道，从部分乘积（y_0x_2）开始，执行两个完整的加法。将部分积（y_0x_2）加到（y_1x_1）上，然后加到（y_2x_0）上，产生了最终结果 P2。对于下通道中的部分积（y_0x_3）执行三次完整的加法。将部分积（y_0x_3）加到（y_1x_2），然后加到（y_2x_1），最后加到（y_3x_0）。

$$P_2 = y_0x_2 + y_1x_1 + y_2x_0$$
$$P_3 = y_0x_3 + y_1x_2 + y_2x_1 + y_3x_0.$$

在加法的过程中，要使用全加器对每次计算的进位也带入计算。在偶数位计算向奇数位计算进位的过程中（如计算 P_2 产生的进位向 P_3 进位），因其在一个周期内完成，可直接使用全加器完成。在奇数位计算向偶数位计算进位的过程中（如计算 P_1 产生的进位向 P_2 进位），因其在两个周期内完成，需要在全加器后使用寄存器来向下一周期的全加器提供进位输入。设计过程中，有一半的全加器后面需添加寄存器。

对于 N 位数据的乘法计算，重复上述过程 N 个周期后，即可得出 2N 位的结果。乘法器总体由最小计算单元连接而成，64 位乘法器，可使用 32 个乘法单元构成。最小乘法单元如图 2-4 所示。其包含 4 个与门、4 个全加器、4 个触发器。

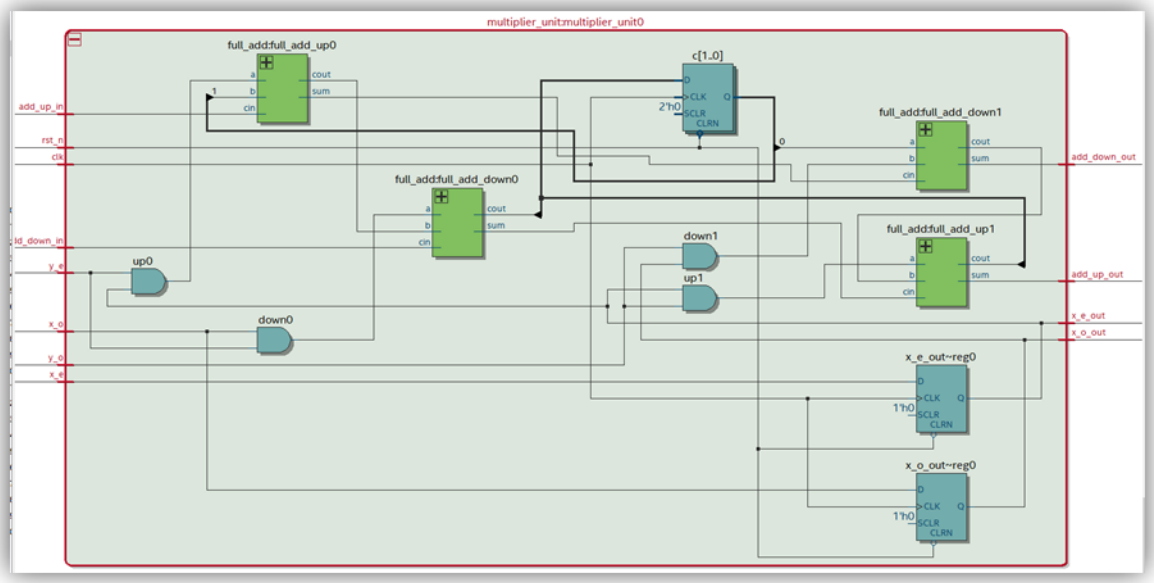

图 2-4　乘法器最小单元结构

三、 仿真结果

3.1 波形仿真

编写 testbench，并在 modelsim 环境下对设计乘法器进行了功能验证。

对于 x、y,分别设置 64 位输入数字为 64’d8563214857120369541 以及 64’d6579858412322574896。

根据波形可见，在时钟使能信号拉高后，输入时钟开始有效。在复位信号拉低清零之后，开始进行计算。对于输入：并行输入 64 位后。利用计数器 cnt_in 每一个周期从 x 中依次提取两位进入乘法器。cnt_in 每个周期加 2，在 32 个周期后加到 64 停止计数，此时 x 输入完毕，停止输入。 对于输出：从端口 add_down_out30、add_up_out30，分别输出下通道和上通道的结果，通过计数器 cnt_out，分别赋予结果 P 的奇数位和偶数位。cnt_out 每个周期加 2，在 64 个周期后加到 128，停止计数，此后乘法器通道输出为 0。得出结果 P 的数值。

根据波形显示，最后得出乘法结果 P 为

128’d56344741314149119762858686956169642736

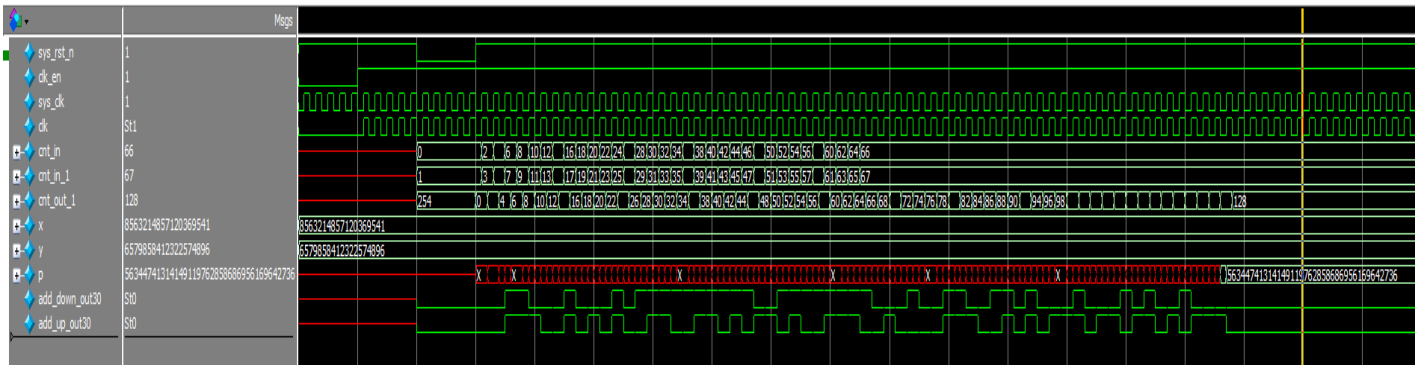

图 3-1 仿真完整波形

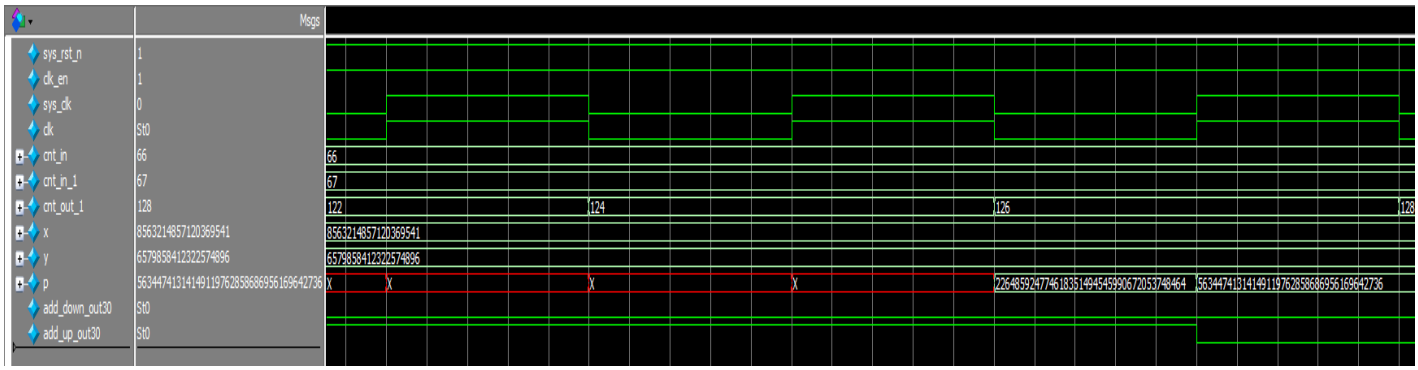

图 3-2 仿真局部波形