# 训练任务：导航用莫尔斯信标

文档版本 2025，2025年9月5日

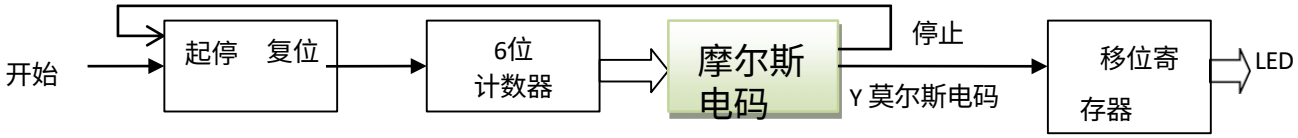
## 目录



## 图示目录

## 作业

设计一个组合逻辑电路，将6位二进制计数器的输出编码为分配给你的摩尔斯电码序列。该序列由0和1组成，并出现在摩尔斯电路的Y输出端。该序列通过你名字的排列组合生成，确保所有练习中序列唯一，且尽可能使所有学生的序列具有近似相同的最小化复杂度。




*注：您只需设计**摩尔斯**电码部分，其余电路已在DCE库中提供。*

图1 — 摩尔斯电码信标原理图

### 设计要求

推导逻辑方程并用VHDL代码描述。先通过GHDL仿真器验证，再在演示电路中验证。

由于任务是训练，摩尔斯电码部分的VHDL代码需满足若干约束条件。

- 在摩尔斯电码部分，可使用任意数量的非门(NOT)、与门(AND)、非与门(NAND)、或门(OR)、非或门(NOR)及异或门(XOR)，以及表示输入(input)、输出(output)、逻辑1(VCC)和逻辑0(GND)的符号。
- 但摩尔斯部分中**四输入多路复用器**不得超过**一个**。多路复用器的优势将在后续课程中讲解。

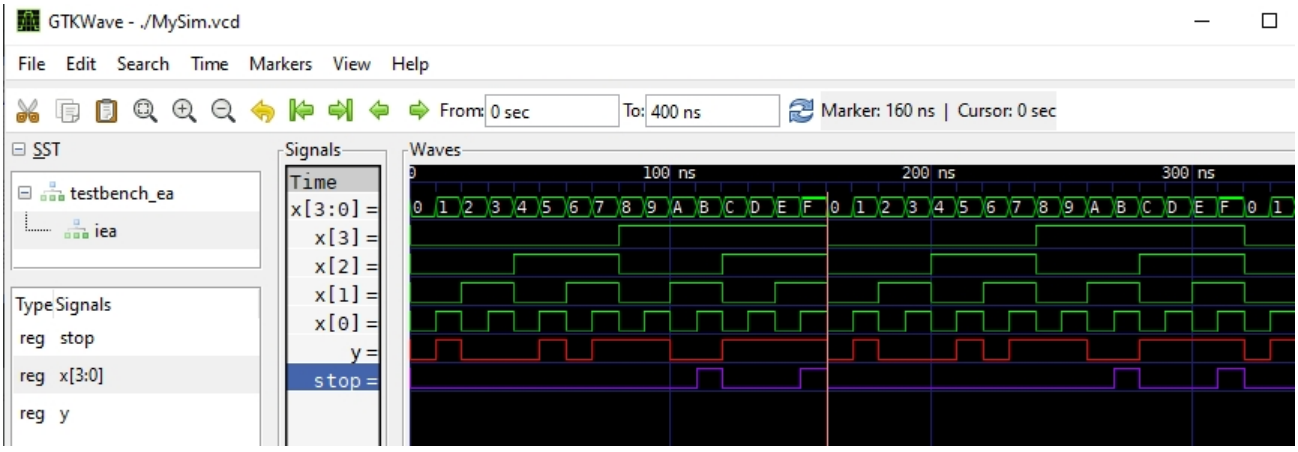
**注：**演示电路将在实践课中共同搭建。届时您只需将MorseMHL2替换为自定义的MorseXXX。另附*.sof Quartus编程文件，可加载至Veek-MT2开发板以参考最终效果。

## 教程：信标EA用摩尔斯电码电路

EA信标（详见第二讲）传输序列010001011100。

| 输入 **x** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 莫尔斯输出**停止** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **1** |
| 摩尔斯EA的**Y**输出 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |

STOP输出仅在序列的最后一位时为逻辑'1'，后续位也可能为'1'，但我们不关心这些输出，因为它们不适用。




逻辑方程在专门讲解逻辑函数最小化的讲座中提出：

图2 — MorseEA - 输入与输出（GHDL仿真器）

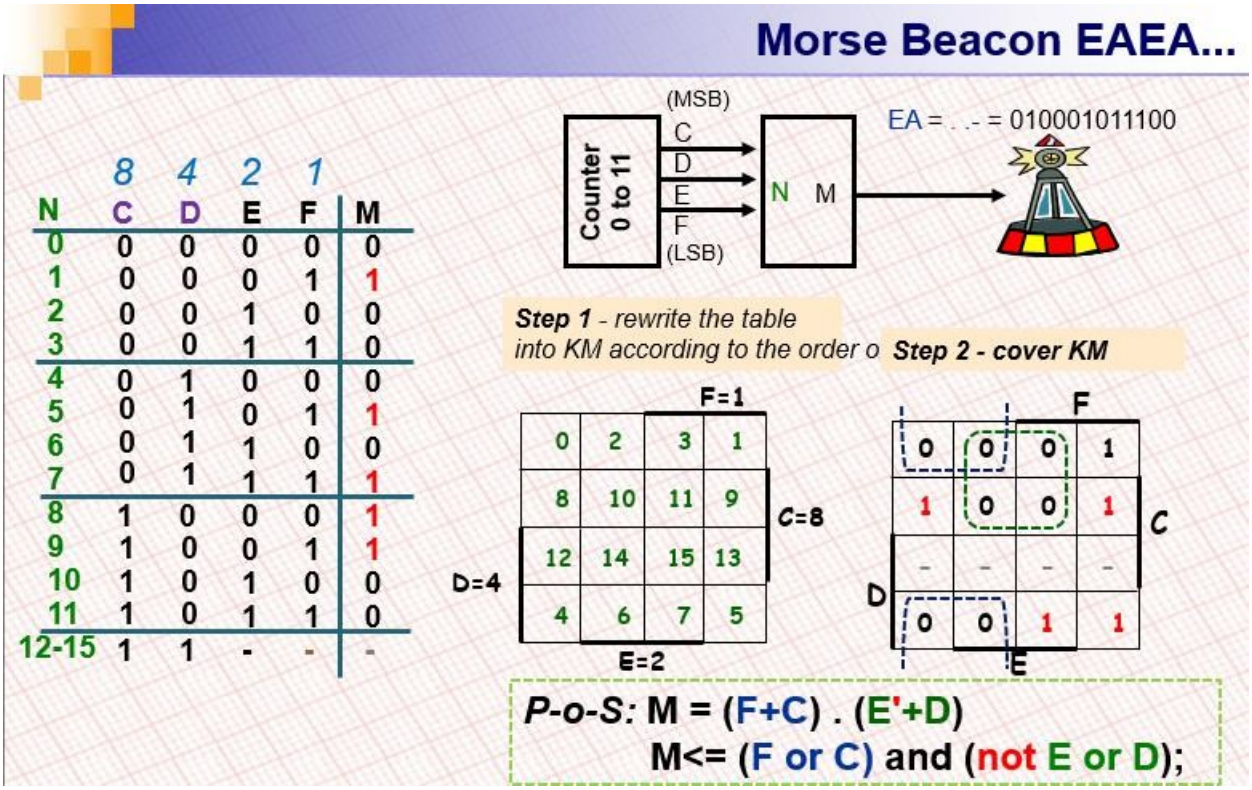



图3 — 来自第2次讲座的EA信标

向量X即为输入信号，其中信号F、E、D和C（标记方式与本文后续MHL2方案一致）将X的最低位X(0)映射至f，X(1)映射至e，X(2)映射至d，X(3)映射至最高位c。为简化GHDL仿真，我们定义X为无符号类型。

为表示莫尔斯电码结束，我们在状态1110=$2^3+2^1+2^0$→$10112_{中}$添加STOP输出。通过与运算将这些位组合为'1'，因为该组合是自$0000_2$起升序排列的二进制数序列中的首个数。

我们将该电路命名为**MorseEA**，并直接用VHDL编写其方程。

```
library ieee;use ieee.std_logic_1164.all;use ieee.numeric_std.all;
实体 MorseEA 定义为
        port (X       : in unsigned(3 downto 0):=(others> '0'); --CDEF
                Y, STOP : out std_logic:='0');
end entity;
MorseEA的行为架构为信号c,d,e,f:
std_logic:='0';begin
        f<=X(0); e<=X(1); d<=X(2); c<=X(3);
        Y <= (f or c) and (not e or d); STOP <= c
        and e and f;
end architecture;
```

请**注意VHDL中括号是必需的。**仅一元非运算符具有更高优先级。在C语言中，与运算符(&&)优先级高于或运算符(||)，但在VHDL和布尔代数中并非如此。逻辑运算符之间不存在优先级关系。

我们建议使用免费的VSC（Visual Studio Code）环境作为编辑器，并配合GHDL仿真器使用。其安装步骤详见GHDL安装指南末尾的说明页面：https://dcenet.fel.cvut.cz/edu/fpga/install_en.aspx。该页面还提供了带注释的摩尔斯电码仿真示例。

在VEEK-MT2开发板上，可通过以下电路同时测试MorseEA：

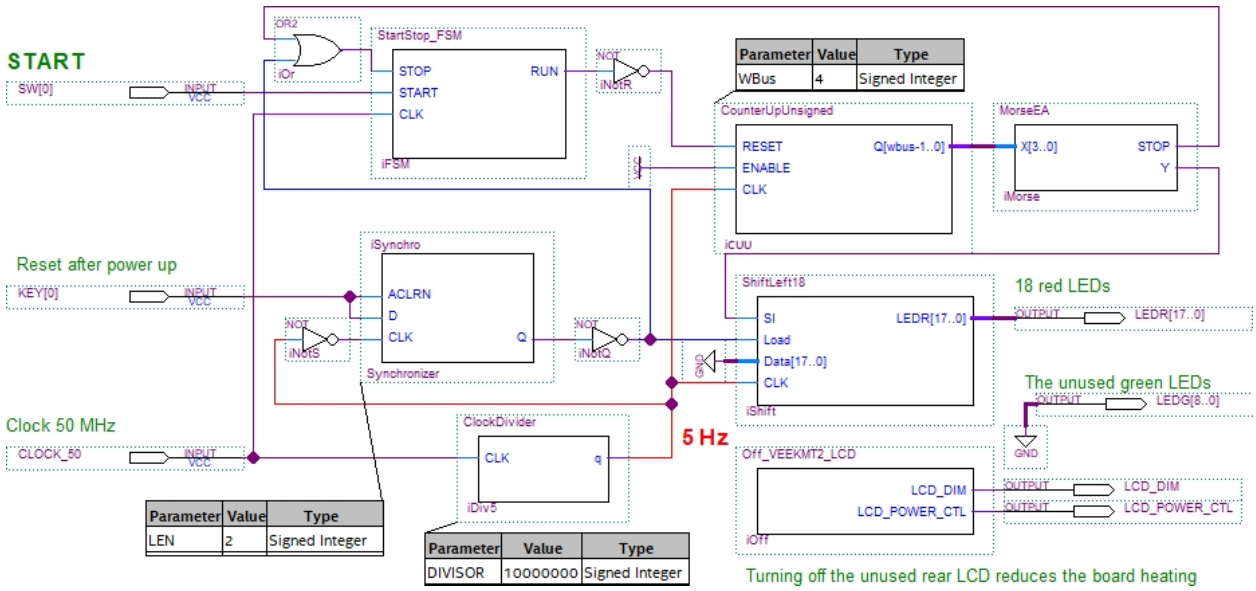



图4 — 摩尔斯EA测试

START-STOP电路作为有限自动机需要更高时钟频率，因此直接连接至Veek-MT2板载的50MHz时钟发生器。

## MorseEA原理图符号

在Quartus的项目导航器窗口中创建该符号：于"文件"选项卡列表中右键单击文件名，选择"为当前文件创建符号文件"。

*注意：此上下文菜单仅适用于VHDL文件。原理图文件（*.bdf - 模块图文件）需要更复杂的操作方式，该方式同样适用于VHDL文件。*

*首先需打开文件。随后在主菜单中选择Qurtus菜单选择"文件"，在其子菜单中选择"为当前文件创建符号文件"。*

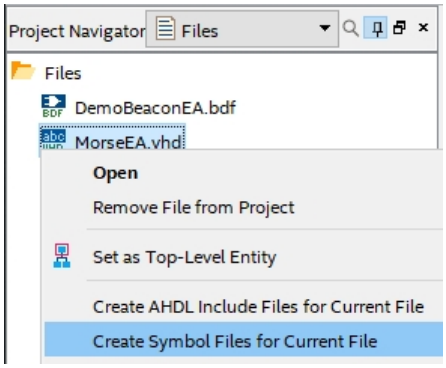



图5 - 创建原理图符号

生成的符号将显示在项目文件的符号编辑器中：

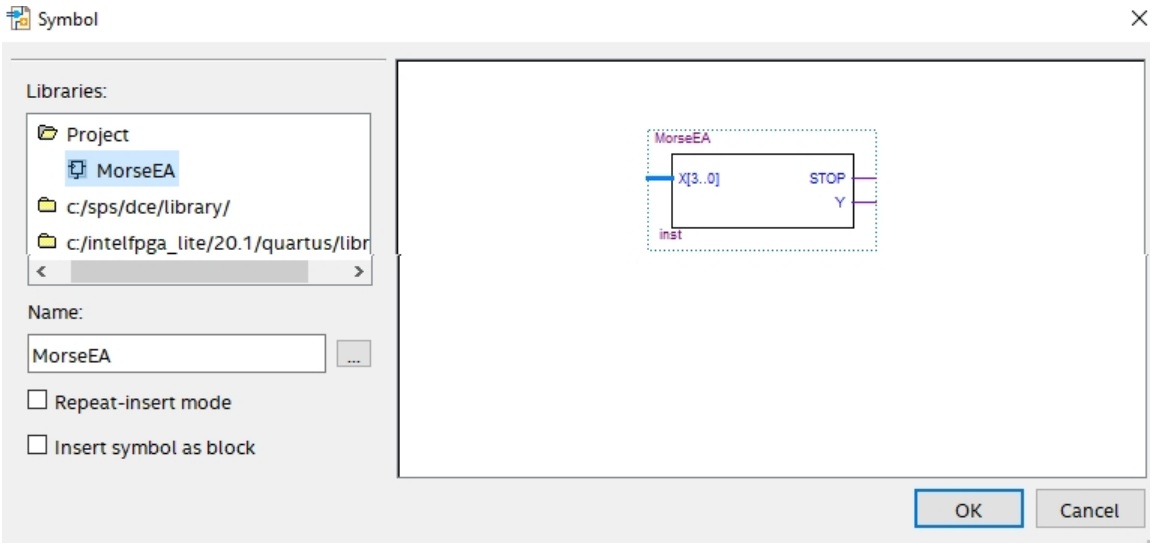
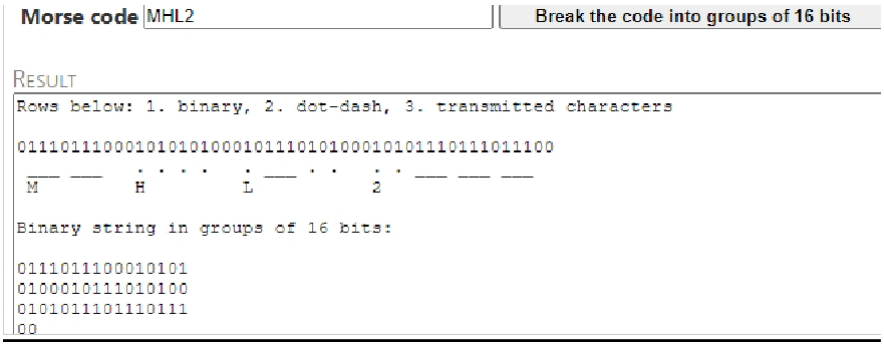



图6 - 符号在原理图编辑器中的显示

### 注：VHDL与Schema中索引的区别

- 原理图编辑器依赖于内部HDL语言AHDL（Altera硬件描述语言）的语法，详见AHDL或AHDL维基，该语言旨在尽可能精简以将描述压缩至原理图中。目前仅由Quartus用于存储器和引脚分配等巨型功能的原理图与原型设计。
- 在VHDL实体中，**输出LEDR: out std_logic_vector(17 downto 0);**于原理图中将简写**为LEDR[17..0]**。例如VHDL写法**为索引LEDR(17)**，而原理图在AHDL中则表示**为LEDR[17]**。
- 同理，输入KEY: in std_logic_vector(3 downto 0);在AHDL中表示为KEY[3..0]，而KEY[0]在VHDL中对应KEY(0)。
- **注意！**方案中使用的向量输出SW、LEDR、LEDG和KEY必须构成一个整体。
  **连续片段的序列**，即类似于其PIN分配定义的方式。
- 仅连接KEY[2]和KEY[1]是可行的，但不能仅使用KEY[0]和KEY[2]（即省略KEY[1]）。必须插入KEY[1]——即使未连接——它在图中存在仅为保持连续向量片段。

模块图原理图编辑器略显陈旧。它仅因全球高校的要求才保留在Quartus中。其原理图不会直接编译，而是转换为某种尚未翻译的HDL语言。而在VHDL中，我们始终必须使用向量的连续部分。




## 教程：MHL2莫尔斯序列

图7 – 在线转换器MHL2

*莫尔斯电码时序符合国际海事标准：*

- *短音符、点或"滴"：点持续时间为一个时间单位；*
- *长音符、划号或"达"：持续三个时间单位；*
- *字符内点与划之间的间隔：为一个点持续时间或一个单位长度；*
- *字母间短间隔：三个时间单位长度。*

*为何序列以'0'开头并以一对"00"结束？*

*初始时计数器为0，摩尔斯字母Y为0，因此无信号输出。但当信标持续运行（START状态恒定开启）时，由于二进制序列始于0且终于00，故在重复序列间插入三次间隔。如此便通过三个单位长度的字符间隔（即000）将末尾字符与首字符正确分离。*

*如何将其他文本转换为摩尔斯电码？*

LSP网站提供文本转摩尔斯码转换器：https://dcenet.fel.cvut.cz/edu/fpga/Morse_en.aspx

### 逻辑函数设计

为简化方程，引入临时变量标签以避免频繁的输入错误。我们将编写

| **MorseMHL2输入：** | **X(0)** | **X(1)** | **X(2)** | **X(3)** | **X(4)** | **X(5)** |
|---|---|---|---|---|---|---|
| **我们的关联：** | **f** | **e** | **d** | **c** | **b** | **a** |

注：当然，网络上能找到许多在线简化工具；有些甚至以Excel表格形式呈现，只需粘贴输入值，结果便即刻呈现。

**然而期末考试将考察卡诺图（KM）**，其验证过程比表达式更便捷。☺ 正因如此，在当今计算机时代，专业设计人员仍广泛使用卡诺图——毕竟它能更清晰地呈现函数全貌。请务必亲手绘制任务所需的卡诺图，掌握其应用方法。

为便于练习，我们首先将给定的二进制序列转换为真值表，详见下页。

请仔细核对所有摩尔斯电码转卡诺图的转换过程。

过去几个学期表明，这里最常见的就是错误！

| 状态 | a | b | c | d | e | f | Y（输出信标） | STOP输出 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 2 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 4 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 6 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 7 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 8 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 10 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 11 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 12 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 13 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 14 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 15 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 16 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 18 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 19 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 20 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 21 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 22 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 23 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 24 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 25 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 26 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 27 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| 28 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 29 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 30 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 31 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 32 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 33 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 34 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 35 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 36 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 37 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 38 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 39 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 40 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 41 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 42 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 43 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 44 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 45 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 46 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 47 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 48 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 49 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 50 | 1 | 1 | 0 | 0 | 1 | 0 | X | X |
| 51 | 1 | 1 | 0 | 0 | 1 | 1 | X | X |
| 52 | 1 | 1 | 0 | 1 | 0 | 0 | X | X |
| 53 | 1 | 1 | 0 | 1 | 0 | 1 | X | X |
| 54 | 1 | 1 | 0 | 1 | 1 | 0 | X | X |
| 55 | 1 | 1 | 0 | 1 | 1 | 1 | X | X |
| 56 | 1 | 1 | 1 | 0 | 0 | 0 | X | X |
| 57 | 1 | 1 | 1 | 0 | 0 | 1 | X | X |
| 58 | 1 | 1 | 1 | 0 | 1 | 0 | X | X |
| 59 | 1 | 1 | 1 | 0 | 1 | 1 | X | X |
| 60 | 1 | 1 | 1 | 1 | 0 | 0 | X | X |
| 61 | 1 | 1 | 1 | 1 | 0 | 1 | X | X |
| 62 | 1 | 1 | 1 | 1 | 1 | 0 | X | X |
| 63 | 1 | 1 | 1 | 1 | 1 | 1 | X | X |

对于如此庞大的真值表，使用卡诺图来最小化逻辑函数会相当困难。我们更倾向于采用香农展开法处理Y'输出（详见课程讲义或教材《FPGA逻辑电路》第49页）。同样地，我们通过运用电路行为知识来生成STOP输出。

**首先进行信标输出的香农展开**：

我们将真值表拆分为四个较小的真值表（"香农协因子"），每个包含4个变量，其余两个变量保持常量值。这些变量将作为"分割变量"，详见第二讲：

a = 0, b = 0

| 状态 | c | d | e | f | Y（摩尔斯电 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 | 1 |
| 3 | 0 | 0 | 1 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | 1 |
| 7 | 0 | 1 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 0 |
| 10 | 1 | 0 | 1 | 0 | 0 |
| 11 | 1 | 0 | 1 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 | 0 |
| 13 | 1 | 1 | 0 | 1 | 1 |
| 14 | 1 | 1 | 1 | 0 | 0 |
| 15 | 1 | 1 | 1 | 1 | 1 |

a = 0, b = 1

| 状态 | c | d | e | f | Y（摩尔斯电 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | 0 |
| 7 | 0 | 1 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 | 1 |
| 9 | 1 | 0 | 0 | 1 | 1 |
| 10 | 1 | 0 | 1 | 0 | 0 |
| 11 | 1 | 0 | 1 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 | 0 |
| 13 | 1 | 1 | 0 | 1 | 1 |
| 14 | 1 | 1 | 1 | 0 | 0 |
| 15 | 1 | 1 | 1 | 1 | 0 |

a = 1, b = 0

| 状态 | c | d | e | f | Y（摩尔斯电码 |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

a = 1, b = 1

| 状态 | c | d | e | f | Y（摩尔斯电 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 | X |
| 3 | 0 | 0 | 1 | 1 | X |
| 4 | 0 | 1 | 0 | 0 | X |
| 5 | 0 | 1 | 0 | 1 | X |
| 6 | 0 | 1 | 1 | 0 | X |
| 7 | 0 | 1 | 1 | 1 | X |
| 8 | 1 | 0 | 0 | 0 | X |
| 9 | 1 | 0 | 0 | 1 | X |
| 10 | 1 | 0 | 1 | 0 | X |
| 11 | 1 | 0 | 1 | 1 | X |
| 12 | 1 | 1 | 0 | 0 | X |
| 13 | 1 | 1 | 0 | 1 | X |
| 14 | 1 | 1 | 1 | 0 | X |
| 15 | 1 | 1 | 1 | 1 | X |

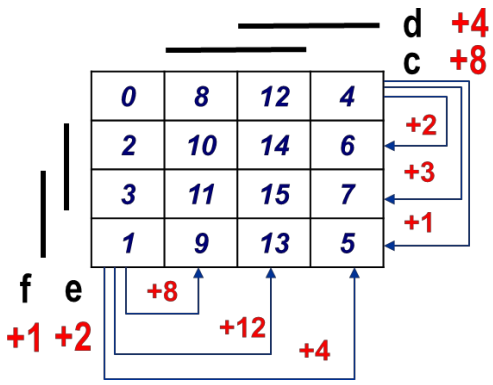
我们现在使用简单的卡诺图来求解逻辑函数的最小化问题，其中输出Y（摩尔斯电码的摩尔斯输出）由四个变量决定，记为Y(a,b) = **f(c,d,e,f)**，其中符号**a**,**b**代表给定图中输入a,b的逻辑值。

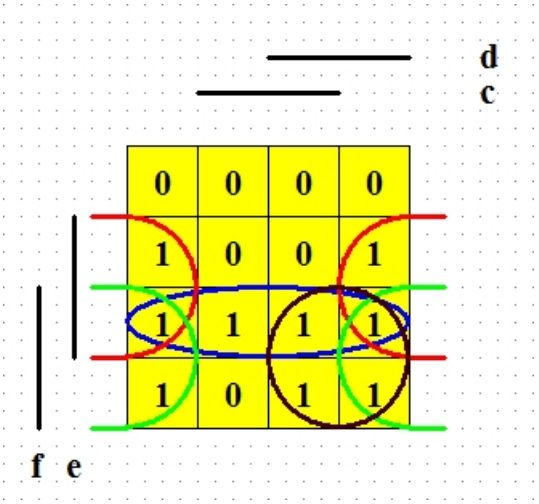
在卡诺图中，值并非按顺序排列，而是按位权重顺序排列，详见

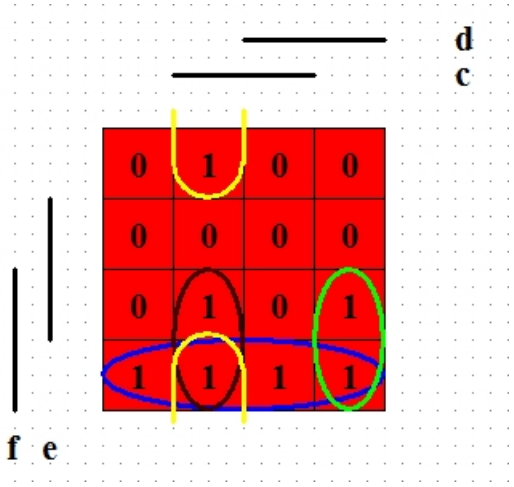
**《FPGA逻辑电路》第34页起。**

我们建议首先通过在单元格中写下序列号来明确顺序。编号取决于输入f、e、d和c的排列方式。若改变其排列，编号也将随之改变。考试中编号可能不同，因此需掌握如何根据卡诺图中任意输入排列构建编号体系。

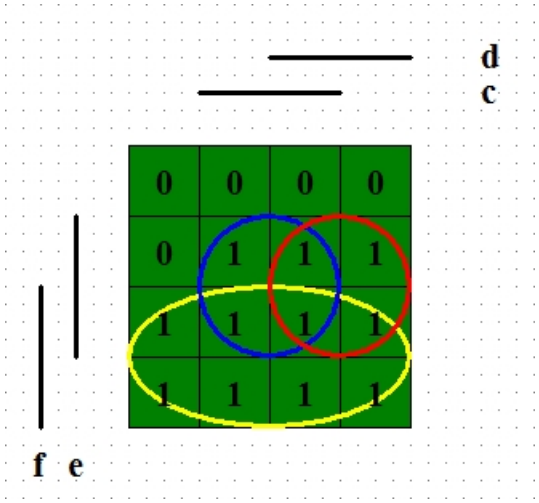
下列卡诺图的序列对应于cdef权重8421。其总和指代卡诺图行号。

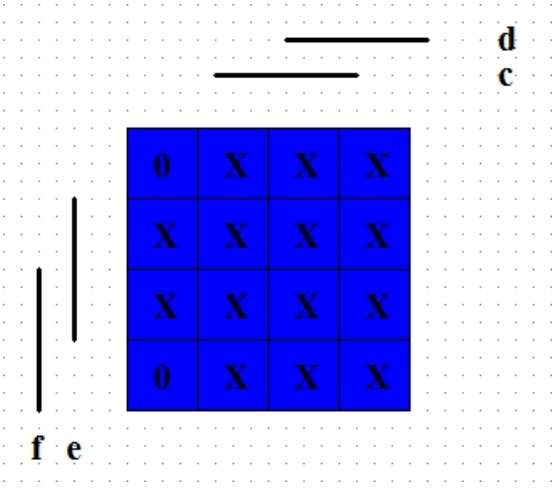



$$Y00 = ef + \bar{c}\bar{e} + \bar{c}\bar{f} + df \quad Y01 = \bar{e}\bar{f} + c\bar{d}\bar{f} + \bar{c}df + cd\bar{e}$$




$Y10 = ce + de + f$ 我们无需解决 Y11 = 0 的矛盾

**注：**分配给您的字母代码无论莫尔斯电码长度如何，均可分解为四个卡诺图（4x4）。无人获得超过64位的序列。

对于**输出STOP，我们利用对电路行为的认知。**已知计数器从零计数至STOP状态后复位，因此序列终止于数字49，其二进制编码为：$49_{10} = 1100012 = 2^5 + 2^4 + 1$

数字49将是0至49数列中首个在第5位、第4位和第0位均出现1的数值。其他数值均无法出现在它之前，因为这将导致数值超过49，违背算术数列的本质规律。因此，该方程足以作为STOP的KOM输出：

$$STOP = abf$$

## 在Quartus开发环境中验证MHL2

首先，我们检查VHDL中的方程，确保逻辑功能设计正确。注：1/ /* */ 注释仅在VHDL2008中有效，而我们正在使用该版本。

***2/ VHDL 中仅使用ASCII 字符，Quartus 不识别特殊字符！***

```
library ieee; use ieee.std_logic_1164.all; use ieee.numeric_std.all;
实体 MorseMHL2 定义如下:
port ( X : in unsigned(5 downto 0):=(others->'0'); -- I/O 初始化仅适用于仿真。
        Y, STOP : out std_logic:='0' );
end entity;
MorseMHL2的行为架构为信号 a, b, c, d, e, f :
std_logic:='0';信号 Y00, Y01, Y10, Y11 :
std_logic:='0';开始
-- 信号重命名
a<= X(5); b<= X(4); c<= X(3); d<= X(2); e<= X(1); f<= X(0);
Y00 <= (e且f) 或 (非c且e) 或 (非c且f) 或 (d且f);
Y01 ≤ (非e且f) 或 (c且非d且f) 或 (非c且d且f) 或 (c且非d且非e);Y10 ≤ (c且e) 或 (d且e) 或 f;
Y11 <= '0';
-- 多路复用器 ——————————
 使用 X(5 至 4) 选择
 Y ≤ Y00 当 "00" 时，Y01 当 "01" 时，Y10 当 "10" 时，
        Y11 当其他情况；-- 2个标准逻辑位可取9×9种值！！！
 STOP<= a and b and f; -- 2**5 + 2**4 + 1 = 49
end architecture;
```

我们编写用于仿真测试的测试台，具体将在课堂上讲解。它代表生成仿真输入的部分，本例中即生成向量x从0到63的取值

```
library ieee; use ieee.std_logic_1164.all; use ieee.numeric_std.all;
库 work;
实体 testbench_MHL2 是 结束实体
;

架构 rtl of testbench_MHL2 是
信号 x: 无符号(5 至 0):=(others => '0');
信号 y, stop : std_logic:='0';
begin
   imhl2 : 实体 work.MorseMHL2 端口映射 (x, y, stop); x<= x+1
            after 10 ns;
end architecture;
```

**我们将在GHDL仿真器中测试该设计。**

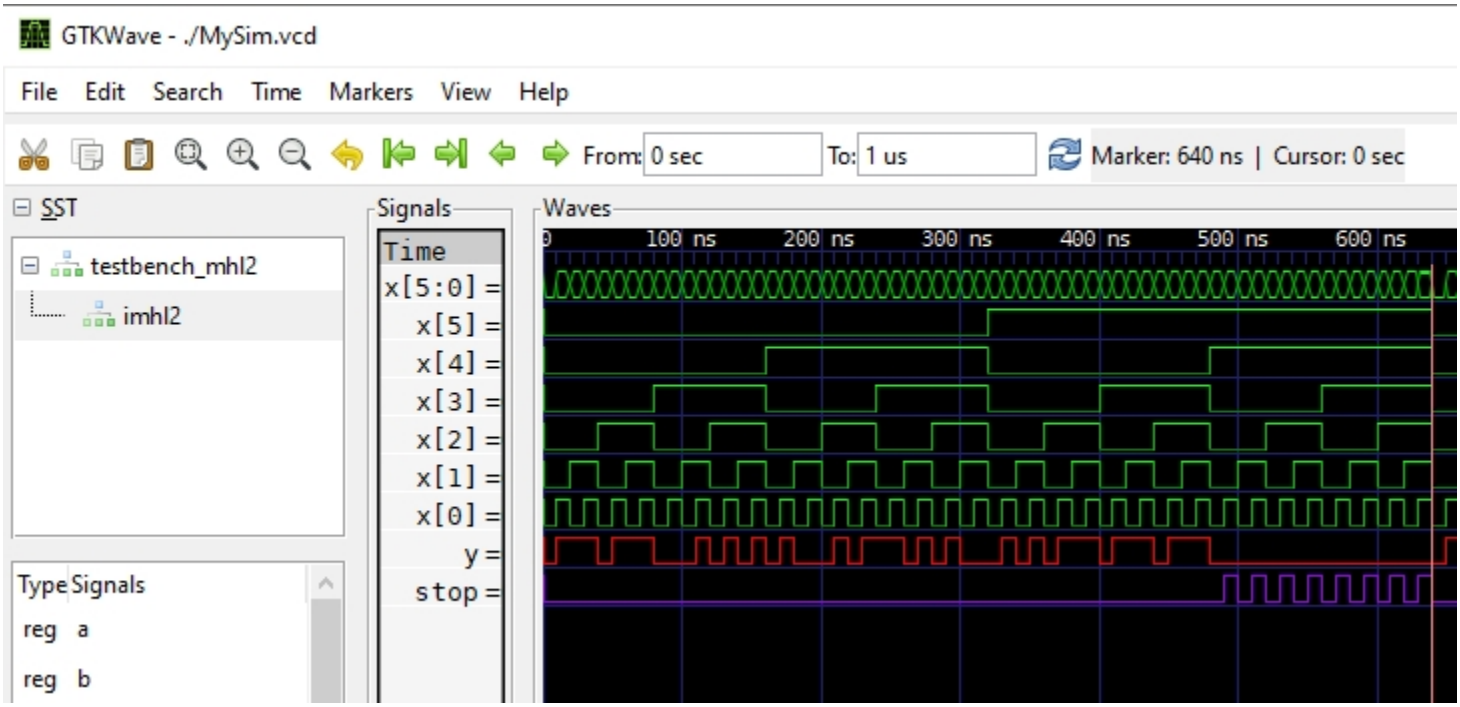



图1 - MorseMHL2仿真输出

**我们可进一步优化方案的用户友好性。**添加蜂鸣器实现摩尔斯电码的蜂鸣提示，同时仍能显示计数器的瞬时读数。在LCD上同步显示静态文本与传输电码也颇有裨益（详见下页示意图）。此外，关闭未使用的绿色LED（LEDG[8..0]）可避免其微弱发光。

**常见错误：**

若电路开始加密，可能是计数器输出至摩尔斯电码电路输入端的位连接错误，导致位序混淆。请注意计数器Q0输出为最低位，必须连接至X0（原f位），而Q4连接至多路复用器的低位，Q5连接至高位。请重新检查！

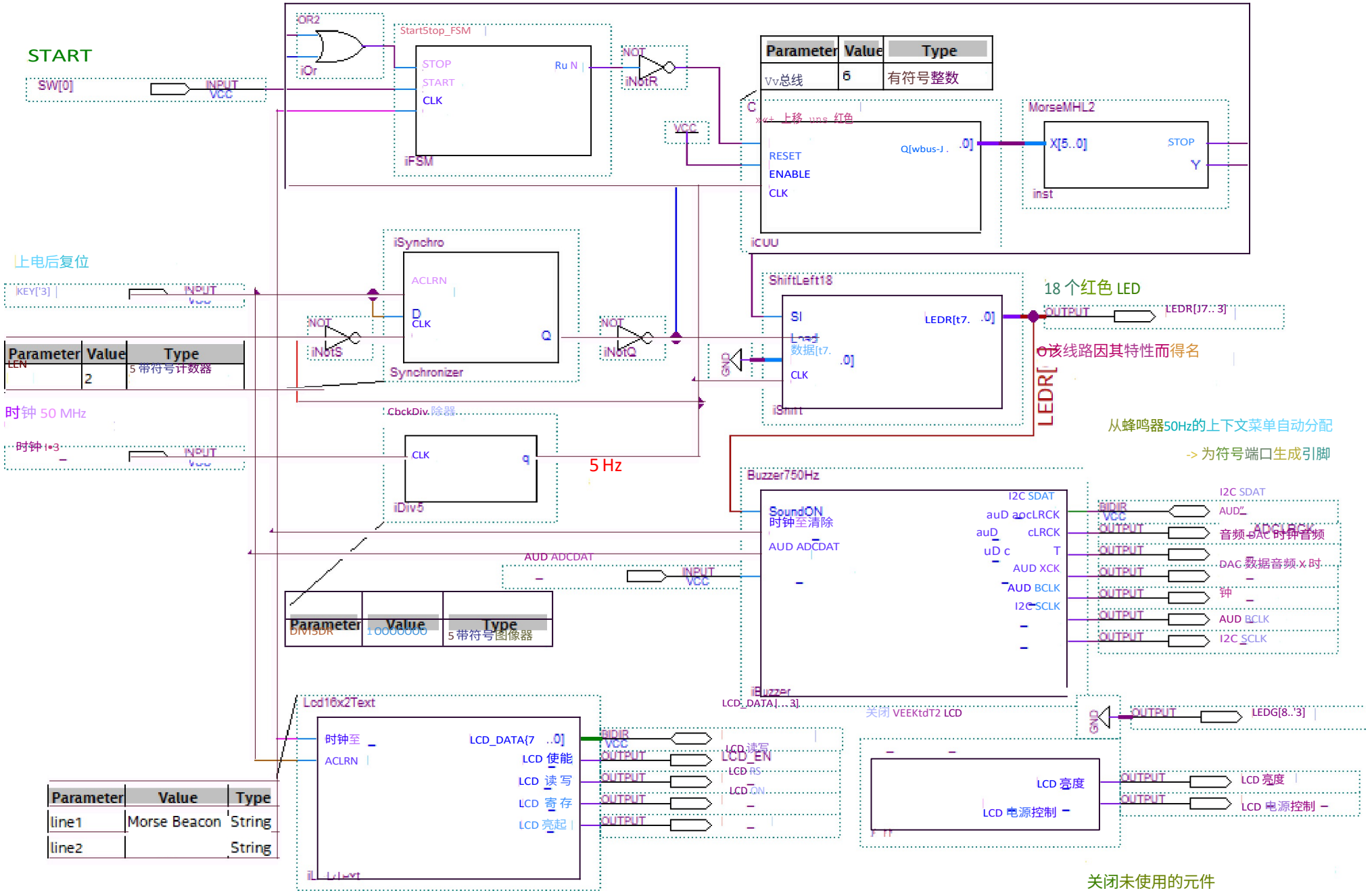



图 8 — 扩展 MHL2 测试方案

## 高级任务-扩展 + 3分

将莫尔斯方程转换为方框图（参见下方MHL2示例），以此练习方案与方程之间的关系。绘制过程可快速完成，下方方案耗时不足半小时。输出部分请使用DCE/library库中的4x1多路复用器。

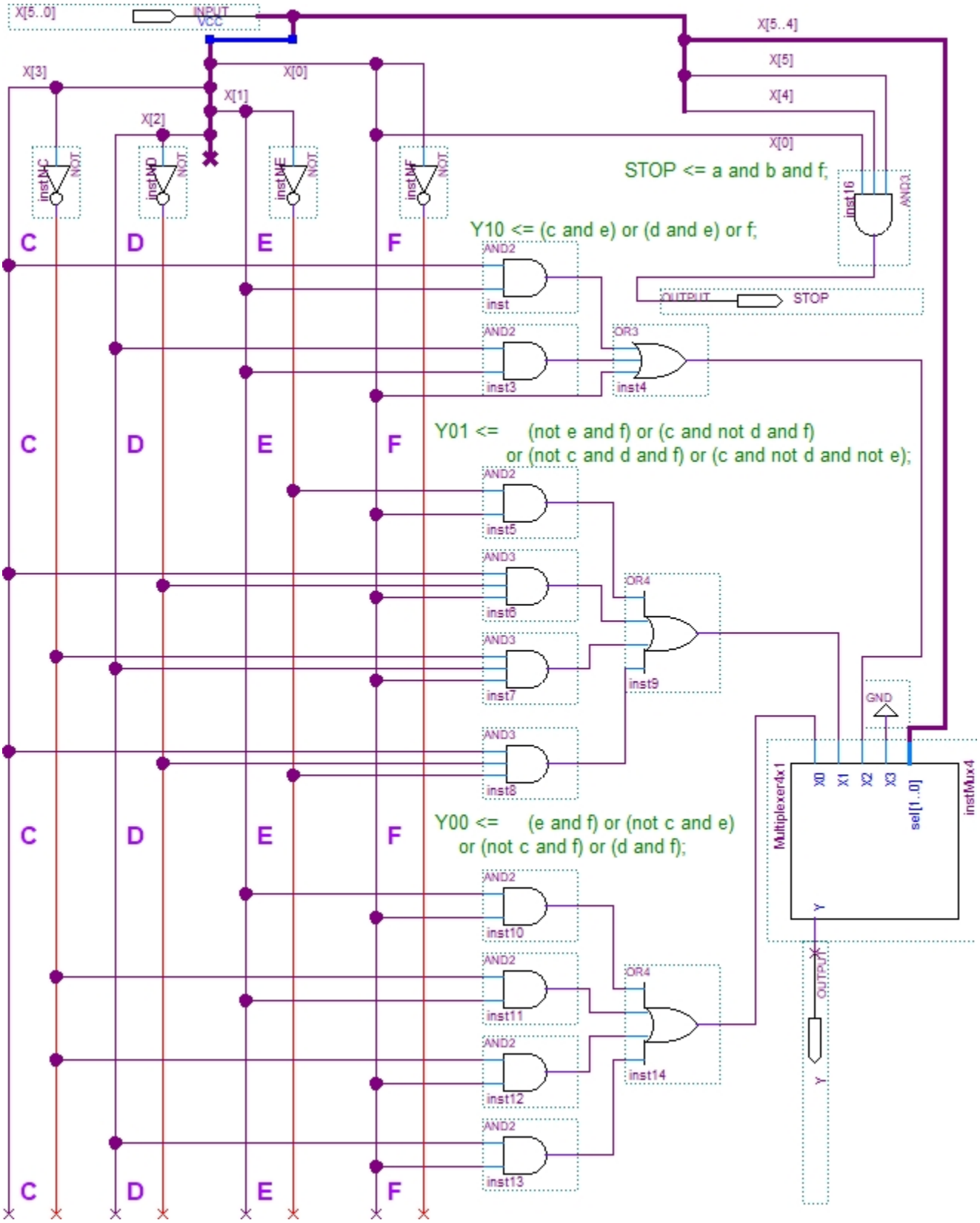



图9 — MHL2的莫尔斯BDF图

为方便操作，您可在任务网站下载预先命名的总线信号压缩原型文件。在图形编辑器中，通过将导线垂直连接至总线并通过"属性"上下文菜单正确命名即可完成总线分支连接。

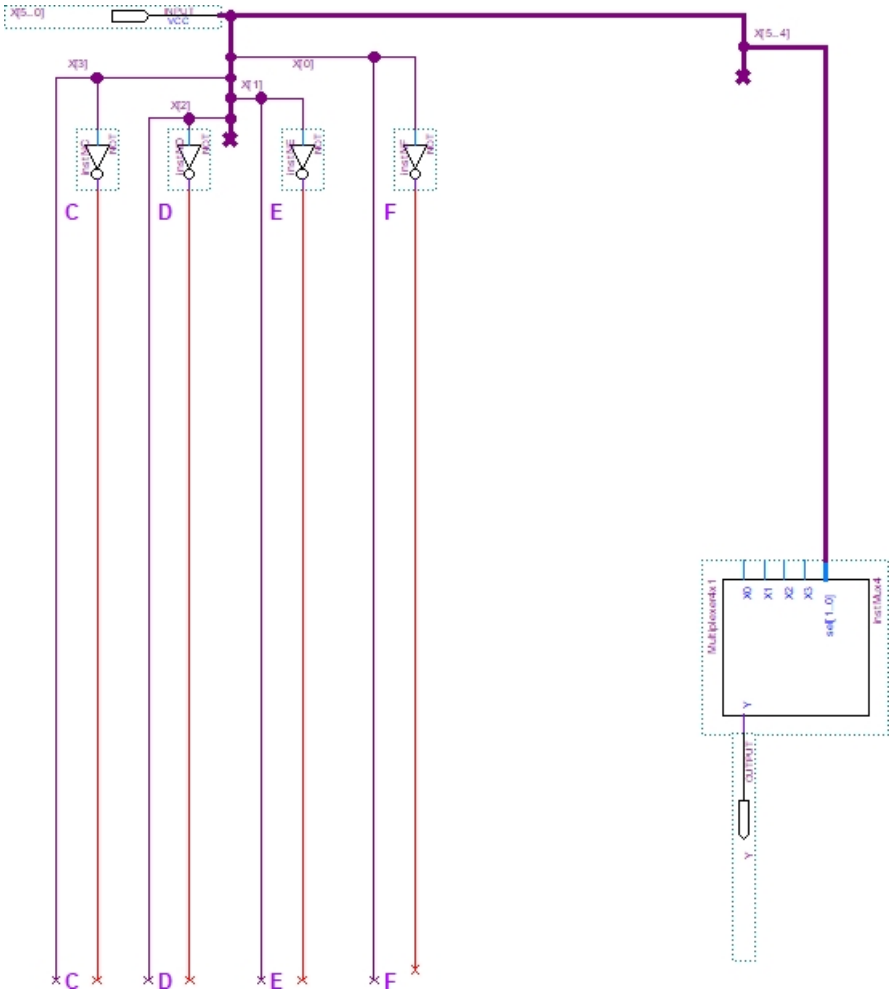



图10 — 原型文件

最终电路文件 MorseBDF_MHL2.bdf 必须采用与 MorseMHL2.vhd 相同的测试方案进行验证。

将两者并联插入电路，通过异或门比较其输出结果。错误（即输出不一致）将显示为'1'。

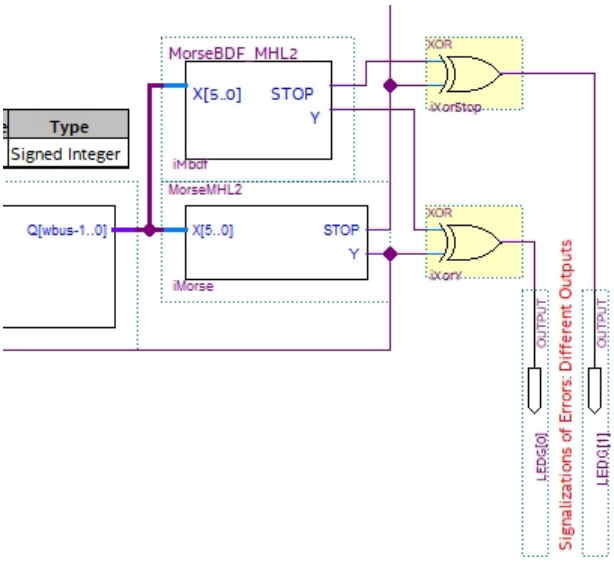



图11 — 双MHL2测试

## 警告：方波门并非标准逻辑门！

请勿使用(N)AND门（5、7、9输入）、与5门及其他标记为**方形块的**逻辑门，而应通过加法器门进行组合。**方形门并非标准门！**它们属于MAX2+库，该库自Quartus 20版起已被移除且未被替代——因符号编辑器仅为高校保留，未再进行维护。

*上述说明仅针对方形门。多路复用器等电路通常采用方形符号标记。*

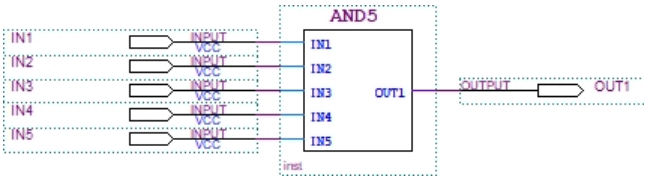
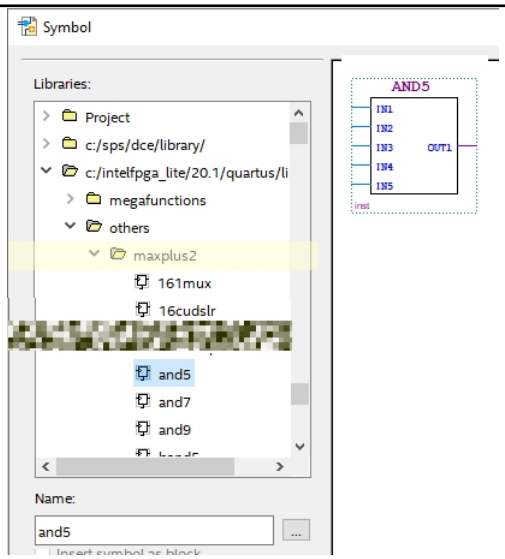
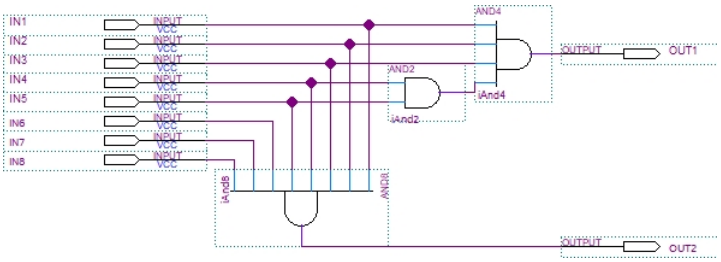
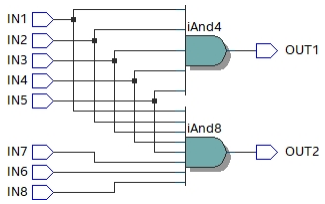
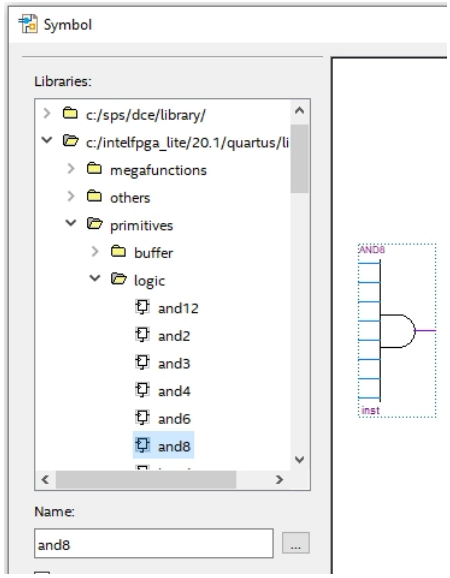
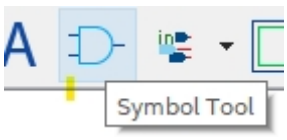
| 属于古老MAX2+库的方形门 | 其他→Max2plus |
|---|---|
|  <br>*错误(12004)：端口"OUT1"不存在于实例"inst"的基元"AND5"中* |  |
| **唯有圆角门能满足我们的需求！** | 基本单元→逻辑 |
|  <br>信息 (293000)：...编译成功。0 个错误。<br>Quartus参与处理（其RTLViewer工具）<br>  |  |

**图12 — 错误的方形门与正确的圆角门**




原理图编辑器中的符号工具虽能提供MAX2+的功能，但您不必全盘接受他人塞给您的东西。☺

**~完　　结束　　~**