目录

概述

为了使用户可以清楚的了解京微雅格（Capital Microelectronics）可配置片上系统（CSoC）之Astro系列芯片的应用方法，在文档的开头，将简要介绍Astro系列芯片的主要内部资源和运行模式。

Astro系列芯片的主要内部资源：

- 1K FP Logic Cells
- 64KB FP配置数据存储器（OTP型，可分为3块）
- 2块9Kb可配置双端口EMB
- 1 FP JTAG配置接口
- 100MIPS高速8051 MCU
- 16KB 8051数据存储器
- 64KB 8051代码存储器（OTP型，可分为2块）
- 1 8051 片上JTAG调试接口（OCDS）
- 512KB FP和8051公用SPI Flash（可分为8块）

Astro系列芯片的主要运行模式：

- 小模式
  不外挂存储器，FP配置数据和8051代码（小于1KB）存储于SPI Flash中，系统上电FP配置数据先从SPI Flash中导出配置FP，完成后再从SPI Flash中导入8051代码到1块EMB中开始运行（参考文档：*AN-Astro系列之EMB用做8051的代码存储器*，即本文档要介绍的内容）。主要用于8051代码较少时，调试生产两便，成本也比较低。
- 调试模式

外挂SRAM存储器，FP Bootloader配置数据和FP用户程序配置数据和8051用户代码分段存储于SPI Flash中，系统上电先从SPI Flash中导出FP BootLoder配置数据（*内含Loader部分*）配置FP并把Loader部分装入2块EMB，完成后EMB中的Loader程序将搬运8051用户代码到外挂 SRAM中，之后再从SPI Flash中导入FP用户程序并重新配置FP，8051代码在外挂SRAM中开始运行（参考文档：*AN-Astro系列之SRAM用做8051的代码存储器*）。
主要用于8051代码超过1KB调试时用，当然也可用于生产，只是外挂SRAM增加了成本。

- 最终生产模式
  不外挂存储器，FP配置数据和8051代码存储于OTP存储器中，因为OTP存储器是一次性的，所以不能用于调试阶段，一般用于产品最终定型后，使用OTP存储器可以获得最优的性能和最低的成本。

EMB简介

Capital Microelectronics Astro系列CSoC芯片内部有2块 9Kbit的EMB，设计人员可以把他们作为8051的代码存储器。当应用程序8051代码比较小（*小于1KB或2KB*）而且要求内嵌8051单片机性能比较高时，本应用文档介绍的方法是一个较好的选择，它可以使Astro内嵌8051单片机达到70MIPS的性能，无需额外的硬件成本（*Astro内部有内置SPI Flash来保持代码掉电不丢失*），应用起来也比较简单。

8051 和EMB的内部互联

- **使用 1 块 EMB 作为 8051 的代码存储器**

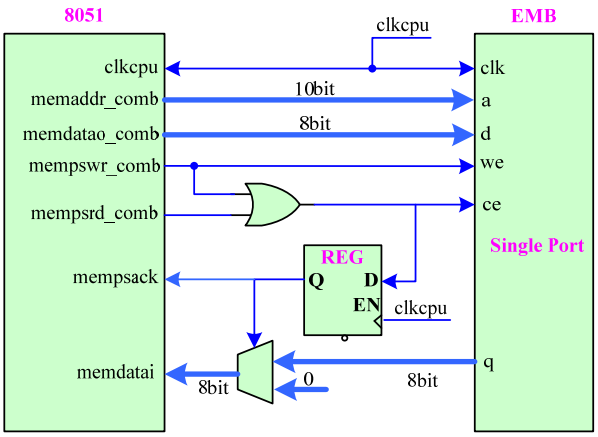


图 1 ： 8051 和 1 块 EMB 的内部互联结构

像图 1 所示一样， EMB 可以作为 SRAM 使用，所以 8051 的存储器数据和地址接口可以直接连接到 EMB 端口上，此时 EMB 需要配置为单端口工作模式。

要使用 EMB 作为 8051 的代码存储器，8051 和 EMB 的内部互联是必须建立的（*通过 FP 实现*），当使用 1 块 EMB 时（*8051 代码小于 1KB，推荐使用*），Primace（*京微雅格的 EDA 工具软件*）系统模板可以帮助设计者轻松实现这种内部互联而无需设计者再为此编写任何代码。

- **使用 2 块 EMB 作为 8051 的代码存储器**

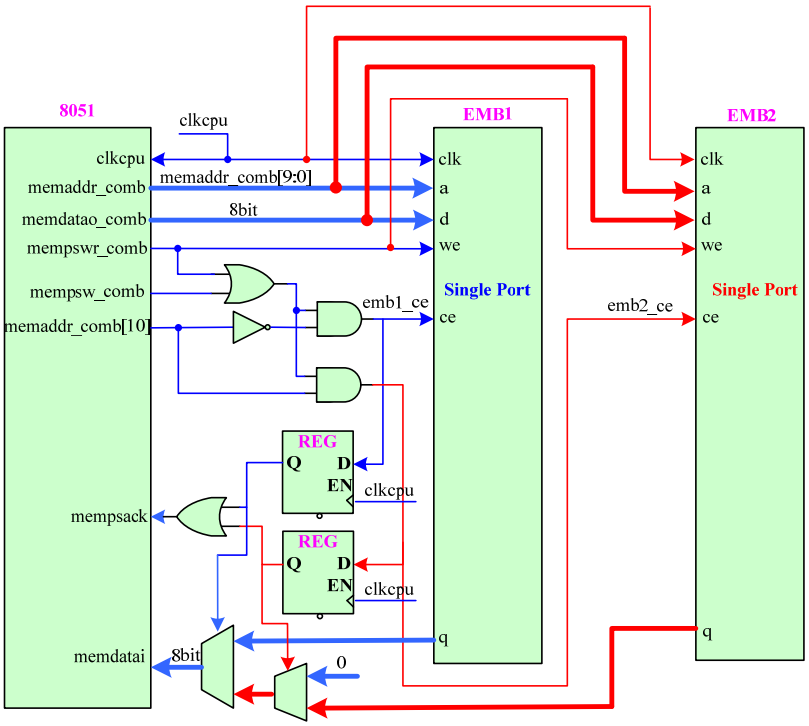


图 2 ： 8051 和 2 块 EMB 的内部互联结构

当使用 2 块 EMB 时，需要进行地址映射，如图 2 所示，代码的低 1KB 地址被映射到 EMB1，高于 1KB 的地址被映射到 EMB2*（当然这只是一个示例，设计者可以根据自己的需求设计相应的地址映射逻辑）*。和使用 1 块 EMB 类似，回到 8051 的 mempsack 信号需要锁存。此外需要说明的是，在使用 2 块 EMB 作为 8051 代码存储器时，设计者需要自己建立这个互连，Primace 是不会自动产生这种模式的代码的*（基于如下原因我们暂不推荐使用 2 块 EMB 作为 8051 代码存储器：Primace 尚未支持 8051 与 2 块 EMB 的内部互联，这样用户在初始化 EMB 时需要将超过 1KB 的 8051 代码分拆成单行的.dat 文件，这样既麻烦又不利于 Debug，所以 2 块 EMB 做 8051 代码存储器的情况仅推荐在代码比较固定的如 Bootloader 等应用场合使用，下面的部分将介绍基于 1EMB 作为 8051 代码存储器这种模式）*。

8051 存储器映射

当使用 EMB 作为 8051 的代码存储器时，它的开始地址是 0x0000，容量是 1KB/2KB*（需要注意的是，EMB 是 SRAM 型存储器，它并不能掉电保存数据，FP 配置数据及 8051 代码在 Download 时是写入 Astro 片上 SPI Flash 存储器的，在上电后系统会自动读取 SPI Flash 中的数据配置 FP 并将 8051 代码装入 EMB 中开始运行）*，Astro 系列 CSoC 同时包含 16KB 的片上 SRAM，可以作为数据存储器使用，如果 16KB 还不够，设计者可以通过 FP 逻辑来扩展数据存储器，如果这样的话，如图 3 所示，16KB 的片上 SRAM 将作为低 16KB 使用，通过 FP 逻辑扩展的数据存储器将使用更高的地址（默认配置下 16KB 作为低 16KB 数据存储器被使用）

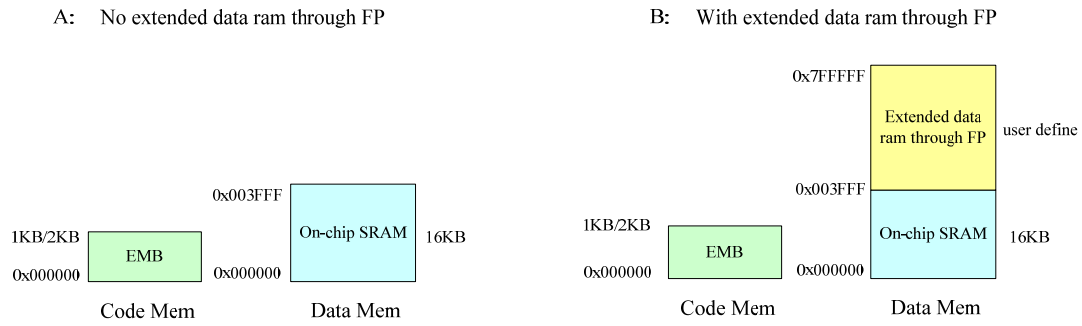


图 3：8051 存储器映射

设计流程

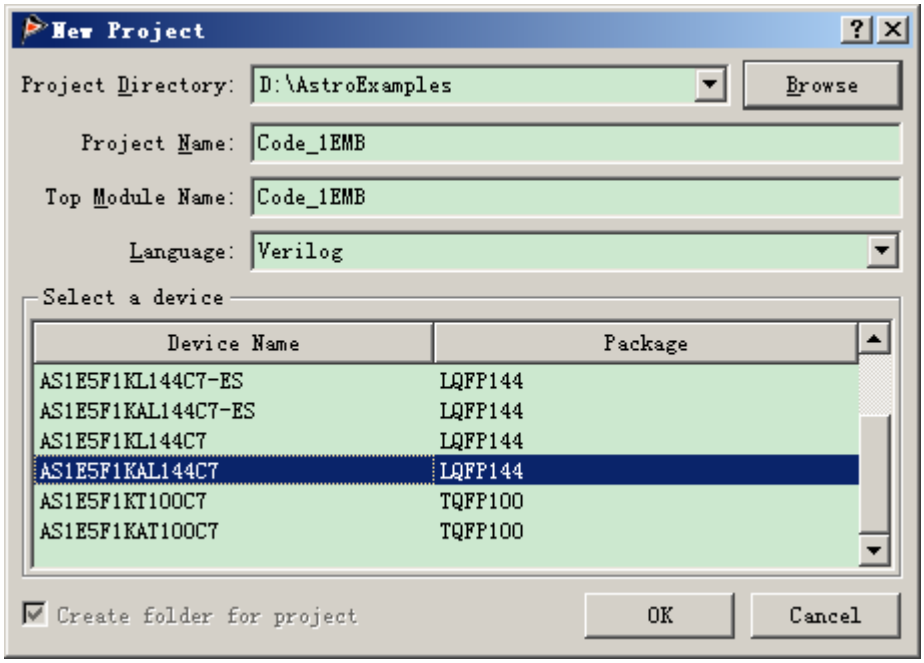
Astro 系列在使用 EMB 作为 8051 代码存储器时比较简单，设计流程可以分为 2 个部分：FP 逻辑设计和 8051 固件设计，下面就介绍此 2 部分的设计过程。

硬件及软件环境的准备

a. CME Astro-EVB-L144 开发板；
b. CME JTAG 调试器及其相应驱动；
c. CME Primace3.0（或其以上版本）EDA 开发环境；
d. Keil uVision2（或其以上版本）；
e. 所有以上硬件的获得及软件环境的安装及使用方法请参考相应的文档，在此不再赘述。

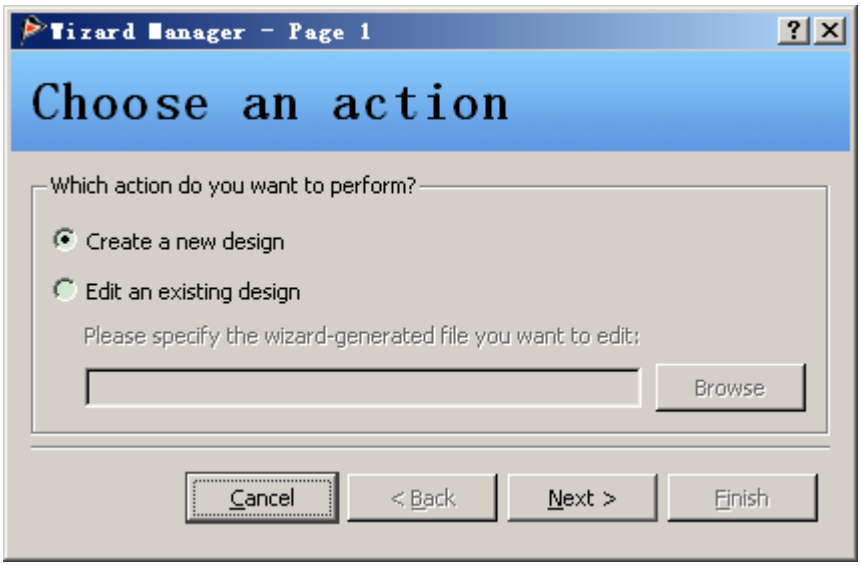
FP逻辑设计流程

FP 部分的实现包括 2 个文件：
a. mySystem.v：Astro 系统配置文件，由 Primace 模板管理器产生；
b. Code_1EMB.v：用户文件，由用户手动产生；
下面分别介绍这 2 个文件的实现：
1） 新建一个工程：
①. 在 Primace 软件【Project】菜单中选择【New Project】新建工程如下图：

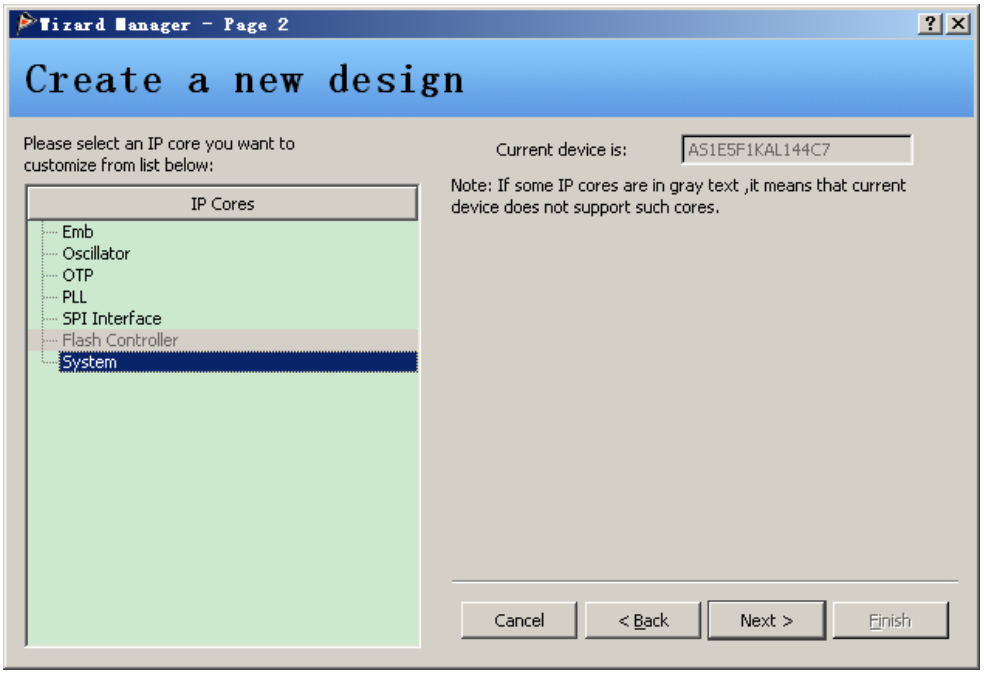


2） mySystem.v 文件的实现：

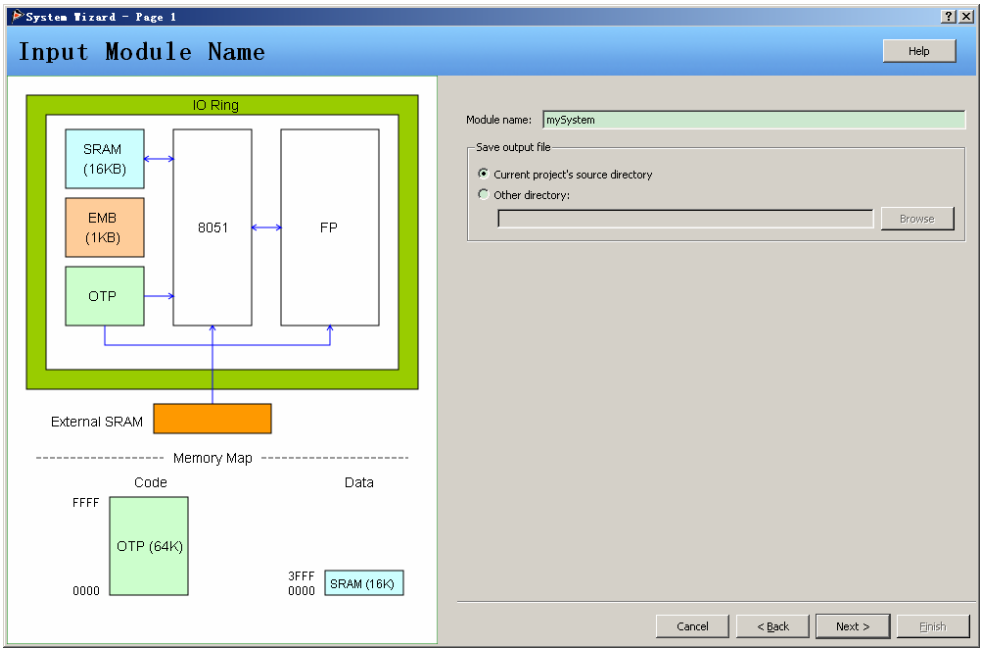
①．在此新工程中（或已有工程），【Tools】菜单下选择【Wizard Manager】打开模板管理器，如下图：



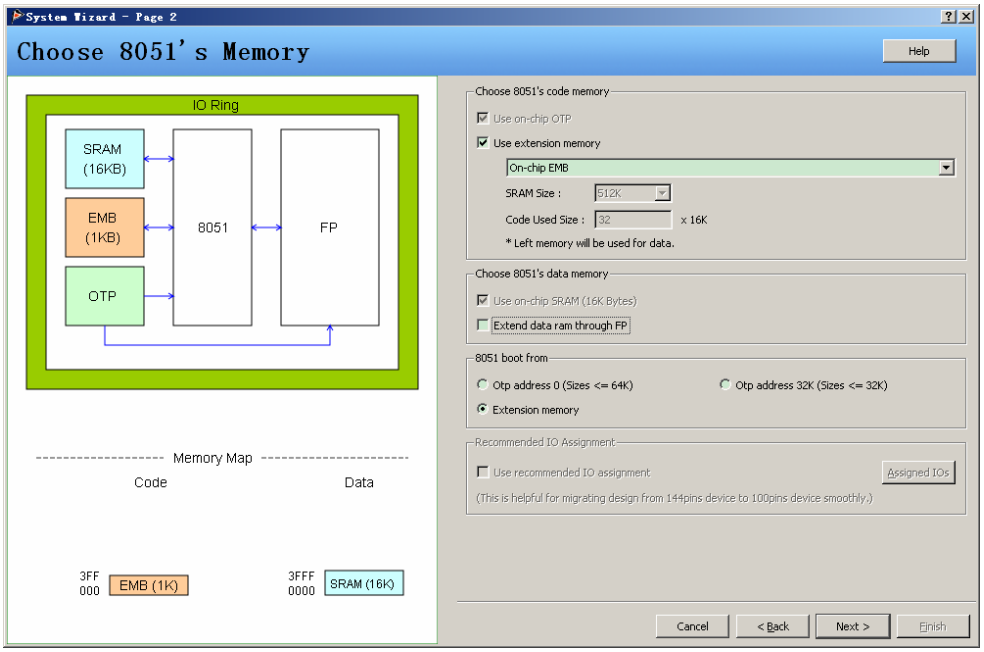
②．选【Next】，如下图：



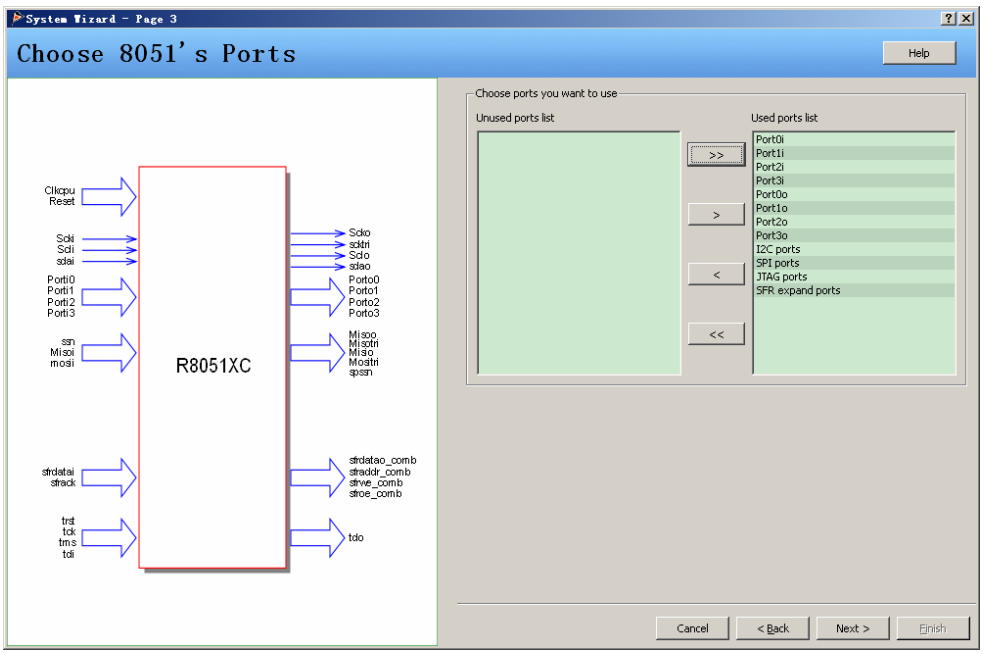
③．IP Cores 中选择 System，选【Next】，如下图：



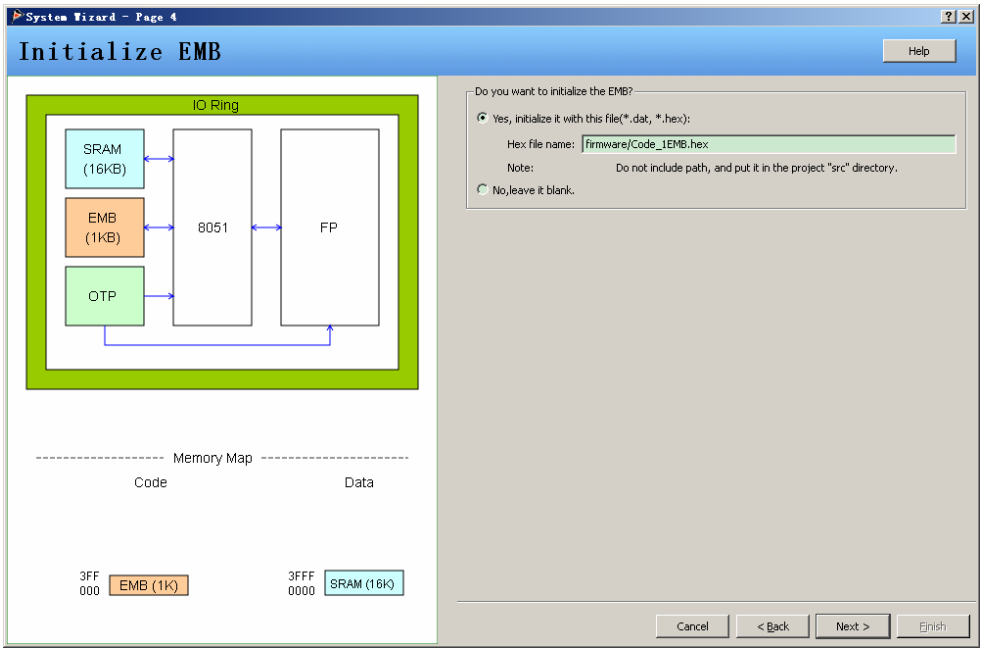
④．模组名称填 mySystem（或其它），选【Next】，如下图：



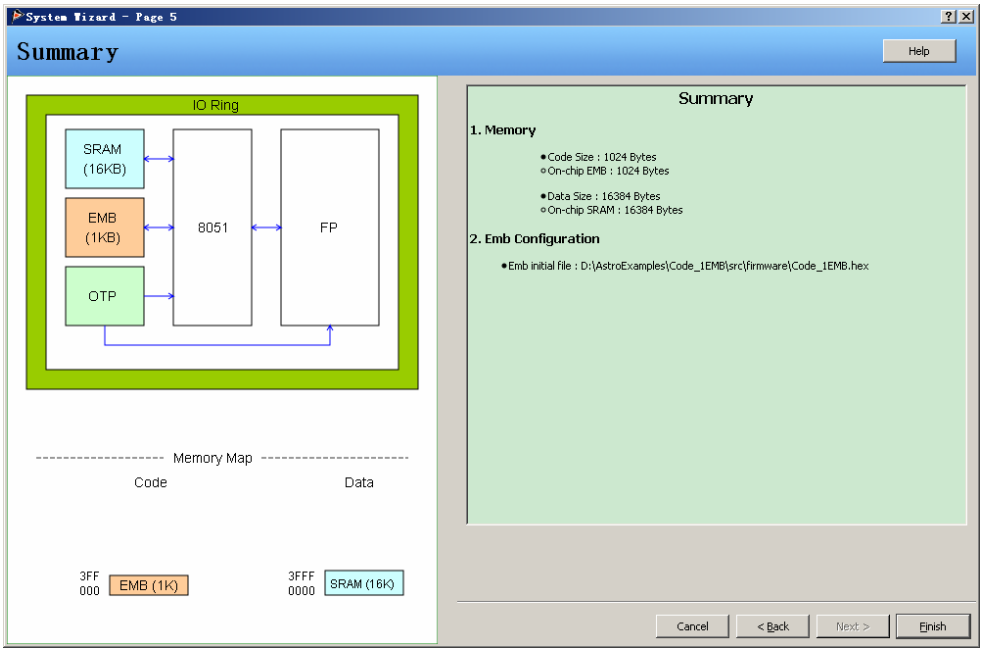
⑤. 勾选 Use extension memory 后，列表框中选 On-chip EMB，8051 boot from 选择 Extension memory，选【Next】，如下图：



⑥. 选择要使用的 8051 功能端口*（注意此处的 JTAG Ports 不是 FP JTAG Ports，而是 8051 的调试接口，如果要在 Keil 中使用 Capital Microelectronics AGDI 在线调试 8051，那么此处 JTAG Ports 必须选择）*，选【Next】，如下图：



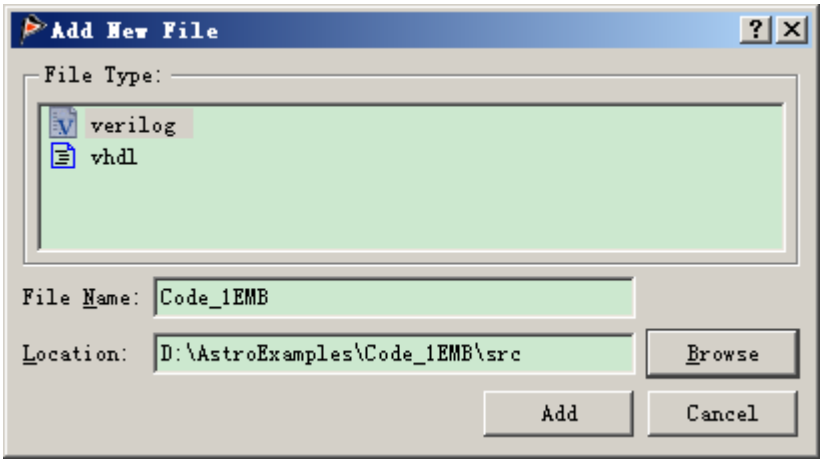
⑦. 填入 Keil 中产生的.hex 文件名称（*注意.hex 文件默认放入工程 src 目录*）***为便于调试，建议将 8051 工程文件夹直接放入 src 目录，并使用相对路径***），*如果这样选择后，系统上电 FP 完成配置后将直接从 EMB 中引导 8051 代码开始运行；如果不确定.hex 文件，可选择 No,leave it black 选项空着.hex，这样系统上电 FP 完成配置后将等待在 Keil 环境中用 AGDI 装入 8051 代码运行*），选【Next】，如下图：



⑧. 择【Finish】完成系统文件的配置，产生的文件将直接加入工程中。

3） Code_1EMB.v 文件的实现：

①. 在工程管理器中右键单击并选择【Add New File …】出现新建文件窗口，如下图：



②．选【Add】，此新的空文件即加入了工程中，具体 Code_1EMB.v 的设计参照具体的应用，本例仅仅是示范性的介绍了怎样用 FP 直接控制顶板 D1 LED 和用 8051 顶板 D2 LED，具体详见 src 文件中的 Code_1EMB.v 文件，唯一需要注意的是此文件中的顶层模块名称应和建立工程时设定的顶层模块名称对应。

③．完成 FP 文件后，分别选【Flow】菜单的【Run Synthesis】和【Run Mapper】将工程进行综合和映射，找出语法错误。

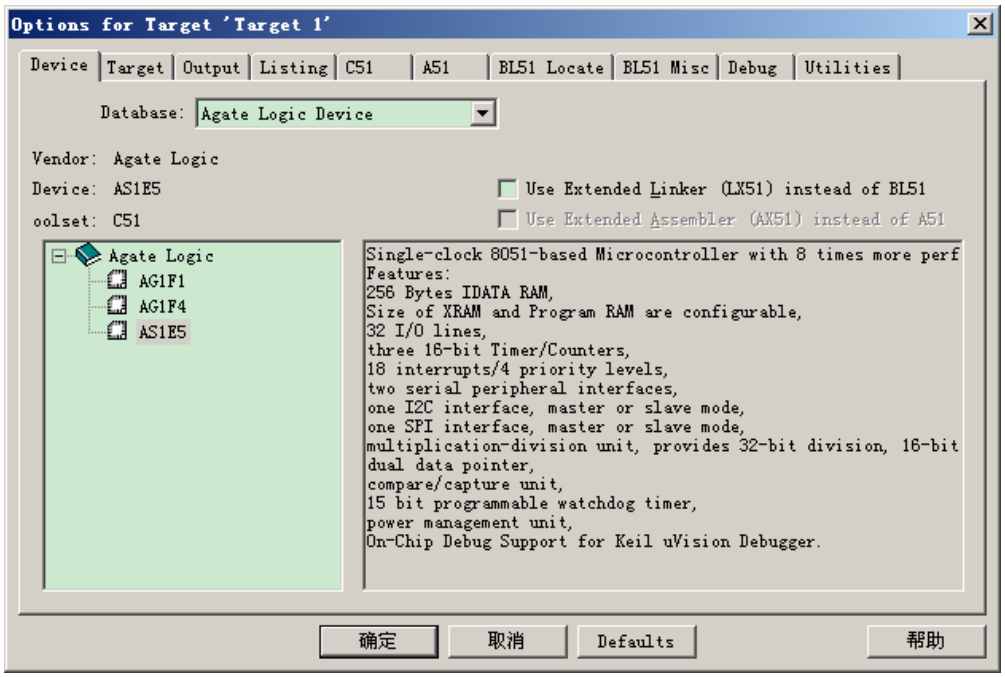
8051 固件设计流程

8051 固件的设计比较简单，分如下两种方式：

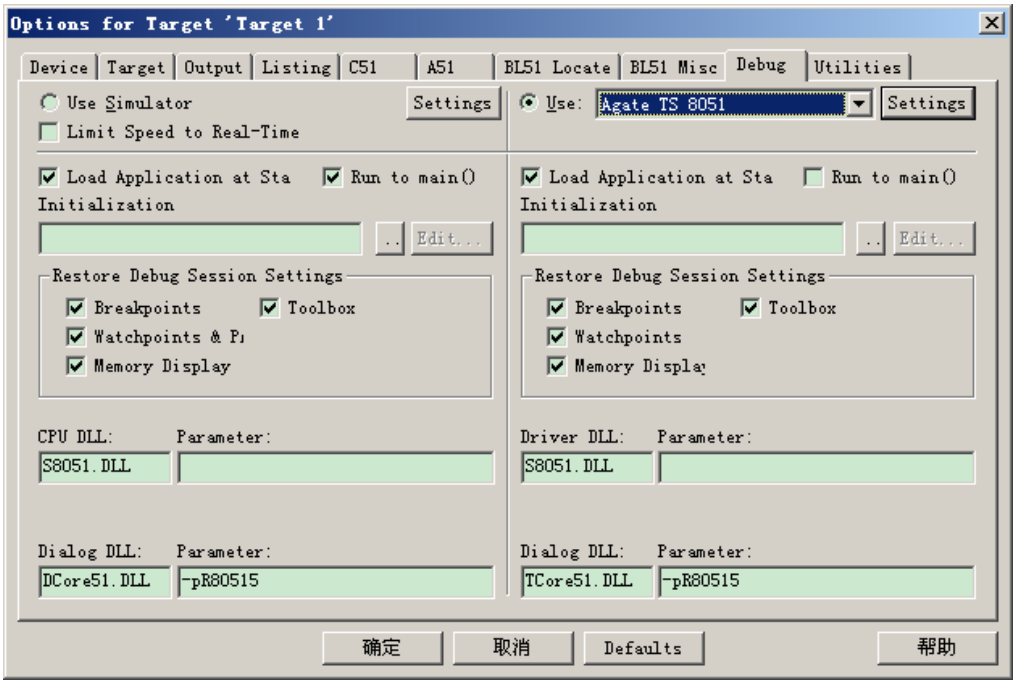
a. 不使用 AGDI 调试器，直接生成.hex 文件，此种方式比较简单，和常规的 8051 设计一样，只需将 Astro 内嵌 8051 头文件 AGR51.h 包含到源文件中即可，根据应用编译产生.hex 文件备用；

b. 使用 AGDI 调试器，在 Keil 中编译完成后，在 Debug 模式下下载并调试，具体实现方法如下：

①．在 Keil 中新建工程，Database 选择 Agate Logic Device（*在 Primace 软件安装完成后会出现这个选项*），元件选 AS1E5，如下图：



②．建立工程后，如果已安装Capital Microelectronics AGDI软件（*未安装的话请参考Keil environment setting for Astro CSoC debug文档安装*），则在工程选项板中就会出现Agate TS 8051 调试器项，选择它，如下图：



③．编译整个工程，准备进入 Debug 模式在线调试 Astro 内嵌 8051。

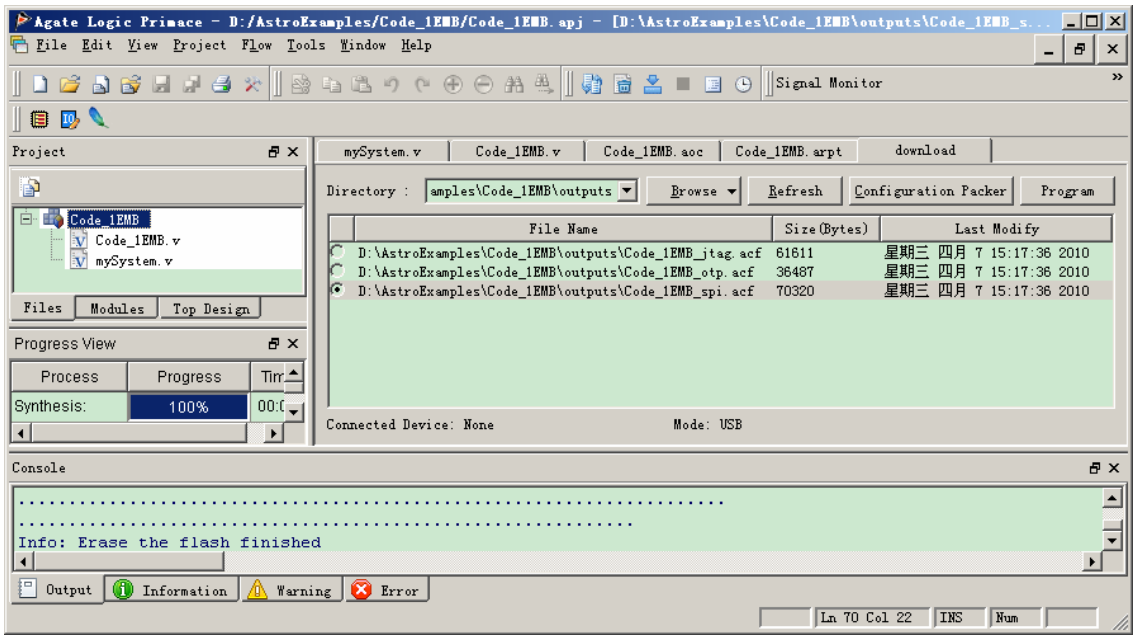
设计综合及下载

通过上面两个步骤，主要流程已经解决，再进行以下几步即可完成整个设计：

a. 调用【Tools】菜单中的【IO Editor】，将输入输出信号根据真实板子对芯片的 IO 进行分配；

b. 调用【Flow】菜单中的【Rerun Project】重新对整个工程进行编译（*当然也可以使用【Flow】菜单中的各个分步功能对工程进行分步编译调试*），成功完成后将获得最后用于下载的流文件（*注意在整个编译过程中没有 error 但有 warning 产生时，应仔细查看每个 warning 的内容看是否会影响实际结果*）。

c. 调用【Flow】菜单中的【Download】或直接点工具栏上的下载图标进入下载界面如下图：



d. 如果只是要观看一下结果，选择 Code_1EMB_jtag.acf 在连接好 JTAG 下载线到 FPGA-JTAG 口后点击【Program】下载，此种方式 FP 配置信息存在于 Astro 内部配置 RAM 中，掉电不保存，仅用于调试；如果需要掉电不丢失，则选择 Code_1EMB_spi.acf 下载，完成后代码将保存在 Astro 片上 SPI Flash 中，掉电不丢失，下次上电会自动运行。

结束语

本文主要介绍 Astro 系列芯片在存储器模式为内部 SPI Flash+内部 EMB 状况下的开发过程，希望籍此能让读者对 Astro 系列芯片的开发有一个直观的了解。

Capital Microelectronics Astro系列CSoC芯片同时拥有 8051 内核和FPGA结构，因此可以有多种灵活和复杂的应用，更多信息请参阅Capital Microelectronics公司网站www.capital-micro.com.或直接咨询相关市场人员。

附录

附录列出了使用 2 块 EMB 做 8051 代码存储器的系统内核文件供参考。

mySystem.v *(仅包括核心部分)*

```
module mySystem (
    clkcpu,         // clock of 8051
    reset,          // reset of 8051, high active
    port0i,         // 8 bit input port
    port1i,         // 8 bit input port, combine with int2-7, ccu, t2, rxd1
    port2i,         // 8 bit input port
    port3i,         // 8 bit input port, combine with int0-1, rxd0i, t0, t1
    port0o,         // 8 bit output port
    port1o,         // 8 bit output port, combine with ccu, txd1
    port2o,         // 8 bit output port
    port3o,         // 8 bit output port, combine with txd0, rxd0o
`ifdef EXSFR
    sfrdatao,       // SFR data bus output
    sfraddr,        // SFR address
    sfrdatai,       // SFR data but input
    sfrwe,          // SFR write enable, high active
    sfroe,          // SFR output enable, high active
`endif
`ifdef EXFP
   imemaddr,      //Extended data memory (through FP) address
   imemdatao,     //Extended data memory (through FP) data output
   imemdatai,     //Extended data memory (through FP) data input
   imemwr,        //Extended data memory (through FP) write enable, high activ
   imemrd,        //Extended data memory (through FP) read enable, high active
`endif
    scki,           // Serial clock input
    scko,           // Serial clock output
    scktri,         // Serial clock tri-state enable, high active
    ssn,            // Slave select input, low active
    misoi,          // "Master input / slave output" input pin
    misoo,          // "Master input / slave output" output pin
    misotri,        // "Master input / slave output" tri-state enable, high active
    mosii,          // "Master output / slave input" input pin
    mosio,          // "Master output / slave input" output pin
    mositri,        // "Master output / slave input" tri-state enable, high active
    spssn,          // Slave select output
    scli,           // Serial clock input
    sdai,           // Serial data input
```

```
    sclo,           // Serial clock output
    sdao,           // Serial data output
    tck,            // 8051 Debug clock(IEEE1149.1 Test Clock)
    tms,            // 8051 Test Mode Select(IEEE 1149.1 Test Mode Select)
    tdi,            // 8051 Debug data input(IEEE 1149.1 Test Data Input)
    tdo         // 8051 Debug data output(IEEE 1149.1 Test Data Output)
    );
.............................
`ifdef EXFP
   reg           memack;
   wire         memwr_comb;
   wire         memrd_comb;
`endif

`ifdef EXFP
   always@(posedge  clkcpu or posedge reset)
   begin
      if(reset)   memack <= 0;
      else       memack <= memwr_comb | memrd_comb;
   end
`endif

    assign  psram0_ce = ~memaddr_comb[EMBMEM_W-1] & (mempswr_comb | mempsrd_comb);
    assign  psram1_ce = memaddr_comb[EMBMEM_W-1] & (mempswr_comb | mempsrd_comb);

   //ack to 8051
   always@(posedge  clkcpu or posedge reset)
   begin
      if(reset)   mempsack_0 <= 0;
      else       mempsack_0 <= psram0_ce;
   end
   always@(posedge  clkcpu or posedge reset)
   begin
      if(reset)   mempsack_1 <= 0;
      else       mempsack_1 <= psram1_ce;
   end

   //to 8051
    assign  mempsack = (mempsack_0 | mempsack_1);

   //read to 8051
`ifdef EXFP
   assign   memdatai = mempsack_0 ? mempsdata0 :
```

```
                    mempsack_1 ? mempsdata1 :
                    memack ? imemdatai : 8'b0;
`else
   assign   memdatai = mempsack_0 ? mempsdata0 :
                    mempsack_1 ? mempsdata1 : 8'b0;
`endif

  //instance for 8051 program store memory
   psram u0_psram(
    .clk    (clkcpu),              // Clock input.
    .a      (memaddr_comb[EMBMEM_W-2:0]),      // Address input.
    .d      (memdatao_comb),    // Data input.
    .ce     (psram0_ce),               // Enable signal. High active.
    .we     (mempswr_comb),     // Write Enable. High active.
    .q      (mempsdata0)           // Data output.
    );
   defparam u0_psram.emb_init_file = "init0.dat";

   psram u1_psram(
    .clk    (clkcpu),              // Clock input.
    .a      (memaddr_comb[EMBMEM_W-2:0]),      // Address input.
    .d      (memdatao_comb),    // Data input.
    .ce     (psram1_ce),               // Enable signal. High active.
    .we     (mempswr_comb),     // Write Enable. High active.
    .q      (mempsdata1)           // Data output.
    );
   defparam u1_psram.emb_init_file = "init1.dat";

`ifdef EXFP
   assign imemaddr = memaddr_comb;
   assign imemdatao = memdatao_comb;
   assign imemwr = memwr_comb;
   assign imemrd = memrd_comb;
`endif

`ifdef EXSFR
   assign sfraddr = sfraddr_comb;
   assign sfrdatao = sfrdatao_comb;
   assign sfrwe = sfrwe_comb;
   assign sfroe = sfroe_comb;
   assign sfrack = 1'b1;
`endif
endmodule
```

关于Capital Microelectronics

Capital Microelectronics 是全球 APGA 技术的首创者和领导者，致力于提供可编程成逻辑器件、可配置片上系统（CSoC）、集成 EDA 工具、IP 及 IC 设计服务，目标定位于通信设备、工业控制系统、消费类电子产品等多种应用类市场领域。

技术支持

电话：+86 10 82150100

E-Mail：support@capital-micro.com.cn

网址：www.capital-micro.com