```
//************************************************/
//*                  ALC XMS                     */
//*   Automatic Local-Clock XBM MEF Synthesis    */
//*      Entrada: arquivo .nounc                 */
//*      Saida  : arquivo .vhdl                  */
//*                                              */
//*  Autor: Felipe Tuyama                        */
//************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "Classes/GenKiss2/GenKiss2.h"
#include "Classes/GenDG/GenDG.h"
#include "Classes/GenFunc/GenFunc.h"
#include "Classes/GenVHDL/GenVHDL.h"
#include "Classes/tools/tools.h"
int Nminstates, Ndependencias;
char fileName[MAX], name[MAX];
bool showDG = false, useDDC = false, showStamina = false;

void getFileName(char* name)
{
   int i = -1;
   while(name[++i] != '.')
      fileName[i] = name[i];
   fileName[i] = '\0';
}

void fileNameDot(char* path, char* extension)
{
   strcpy(name, path);
   strcat(name, fileName);
   strcat(name, extension);
}

int main(int arc, char** argv)
{
   printf(KBLU);
   printf("****************************************\n");
   printf("*             ALC XMS                  *\n");
   printf("*    Local-Clock XBM MEF Generator     *\n");
   printf("*                                      *\n");
   printf("****************************************\n");
   printf("*   By Felipe Tuyama                   *\n");
   printf("****************************************\n");
   printf(KYEL);

   //**************************************/
   //*  Interpretação do Comando de ALC XMS  */
   //**************************************/
   if (arc > 1) getFileName(argv[1]);
   else getFileName("scsi-init-send-1.nounc");
   if (arc > 2 && argv[2][0] == '1') useDDC = true;
   if (arc > 3 && argv[3][0] == '1') showDG = true;
   printf ("%sFileName[%s]%s\n", KRED, fileName, KYEL);
   system ("mkdir -p ALC_XMS");
   system ("mkdir -p ALC_XMS/kiss2");
   system ("mkdir -p ALC_XMS/log");
   system ("mkdir -p ALC_XMS/blif");
   system ("mkdir -p ALC_XMS/vhdl");

   //**************************************/
   //* Converte especificação XBM -> Kiss2   */
   //**************************************/
   // Realiza a conversão da especificação
   // Input: Nounc @ Output: Kiss2
   printf("%s$ Conversão XBM - KISS2.%s\n", KYEL, KWHT);
   fileNameDot("", ".nounc");
   GenKiss2(name, "ALC_XMS/kiss2/arquivo.kiss2", useDDC);

   //**************************************/
   //*        Minimização de estados        */
```

```
   //****************************************/
   // Uso da ferramenta Stamina para minimizar estados
   // Input: Kiss2 @ Output: Kiss2

   printf("%s$ Minimização de Estados.%s\n", KYEL, KWHT);
   if (showStamina == true)
      system("stamina -v 1 -o ALC_XMS/kiss2/arquivo_min.kiss2 ALC_XMS/kiss2/arquivo.kiss2");
   else system("stamina -o ALC_XMS/kiss2/arquivo_min.kiss2 ALC_XMS/kiss2/arquivo.kiss2");
   Nminstates = analyseStatesKiss("ALC_XMS/kiss2/arquivo_min.kiss2");
   printf ("> Nº Estados minimizados: %d\n", Nminstates);

   //****************************************/
   //*   Análise do Grafo de Dependência     */
   //****************************************/
   // Lê o arquivo Kiss2 minimizado
   // Faz análise do Grafo de Dependências
   printf("%s$ Análise do Grafo de Dependência.%s\n", KYEL, KWHT);
   Ndependencias = analyseGD("ALC_XMS/kiss2/arquivo_min.kiss2", showDG);
   printf("> Dependências detectadas: %d%s\n", Ndependencias, KMAG);

   //****************************************/
   //*        Assinalamento de estados       */
   //****************************************/
   // Uso da ferramenta Jedi / Assinalamento One-Hot
   // Conforme os resultados da análise realizada.
   // Input: Kiss @ Output: Blif
   printf("%s$ Assinalamento de Estados.%s\n", KYEL, KWHT);
   // Codificação One-Hot
   if (Ndependencias != 0)
   {
      // system("jedi -e h arquivo_min.kiss2 >arquivo.blif");
      system("jedi -p ALC_XMS/kiss2/arquivo_min.kiss2 >ALC_XMS/blif/arquivo.blif");
      GenFunc("ALC_XMS/blif/arquivo.blif", "ALC_XMS/blif/FGC.blif", "ALC_XMS/blif/OUT.blif", "ALC_XMS/
blif/NSTATE.blif");
   }
   // Assinalamento convencional usando JEDI
   else
   {
      system("jedi -p ALC_XMS/kiss2/arquivo_min.kiss2 >ALC_XMS/blif/arquivo.blif");
      GenFunc("ALC_XMS/blif/arquivo.blif", "ALC_XMS/blif/FGC.blif", "ALC_XMS/blif/OUT.blif", "ALC_XMS/
blif/NSTATE.blif");
   }

   //****************************************/
   //*          Minimização Lógica           */
   //****************************************/
   // Uso da ferramenta Espresso
   // Input: Kiss @ Output: Eq. Booleanas
   printf("%s$ Minimização Lógica.%s\n", KYEL, KWHT);
   system("espresso -o fr ALC_XMS/blif/arquivo.blif >ALC_XMS/blif/arquivo_min.blif");
   system("espresso -o fr ALC_XMS/blif/FGC.blif >ALC_XMS/blif/FGC_min.blif");
   system("espresso -o fr ALC_XMS/blif/OUT.blif >ALC_XMS/blif/OUT_min.blif");
   system("espresso -o fr ALC_XMS/blif/NSTATE.blif >ALC_XMS/blif/NSTATE_min.blif");

   //****************************************/
   //*          Conversão para VHDL          */
   //****************************************/
   // Conversão das Eq. Booleanas para VHDL
   // Input: Eq. Booleanas @ Output: Códigos VHDL
   printf("%s$ Conversão para VHDL.%s", KYEL, KWHT);
   GenVHDL("ALC_XMS/blif/FGC.blif"   , "ALC_XMS/vhdl/FGC_Block.vhdl");
   GenVHDL("ALC_XMS/blif/OUT.blif"   , "ALC_XMS/vhdl/OUT_Block.vhdl");
   GenVHDL("ALC_XMS/blif/NSTATE.blif", "ALC_XMS/vhdl/NSTATE_Block.vhdl");
   GenDLatchVHDL("ALC_XMS/vhdl/D_Latch.vhdl");

   fileNameDot("ALC_XMS/vhdl/", ".vhdl");
   assembleVHDL("ALC_XMS/kiss2/arquivo_min.kiss2", "ALC_XMS/blif/arquivo.blif", name);

   writeLog("ALC_XMS/log/log.txt", Nminstates, Ndependencias);
   return 0;
}
```