ПРИЛОЖЕНИЕ П1: ПРОЕКТ I-T001

Проектът *I-T001* съдържа реализациите на генераторите, разгледани в т. 2.1: *LFSR* конфигурация на Фибоначи, *LFSR* конфигурация на Галоа, *Алгоритъм-М* на Кнут.

```
/*
 * I-T001.xc
 *
 *  Създаден на: 16.05.2012
 *   Автор: Луканчевски
 *   Генератори:
 *   - LFSR конфигурация на Фибоначи;
 *   - LFSR конфигурация на Галоа;
 *   - Алгоритъм-М (Кнут).
 *
 *   Полином: x^32 + x^7 + x^5 + x^3 + x^2 + x + 1
 *   Hex код: 0x80000057
 *   Нотация: Филип Купман (Philip Koopman) - в Hex кода се
пропуска мл. бит
 *
 *   За извеждане на генерираната последователност се използва
линията:
 *   XD0[P1A0] >>> СЛУЧАЙНА ПОСЛЕДОВАТЕЛНОСТ
 *
 *   За управление на процеса taskL се използва линията:
 *   XD1[P1B0] >>> СТРОБ СИНХРО >>> XD13[P1F0]
 *
 */

#include <xs1.h>
#include <stdlib.h>
//#include <string.h>
//#include <stdio.h>

#define RNG_TYPE  3
// 1 - LFSR конфигурация на Фибоначи
// 2 - LFSR конфигурация на Галоа
// 3 - Algorithm-M

#define LFSR_F_VER  2
// 1 - RNG_LFSR_F() работи с фиксиран полином
// 2 - RNG_LFSR_F() работи с полином, задаван като параметър

typedef enum {FALSE=0, TRUE} BOOL;
typedef unsigned int UINT;
typedef unsigned char BYTE;

in port iportButStart = XS1_PORT_1K;  // бутон стартиране на ге-
нерацията
in port iportButStop  = XS1_PORT_1L;  // бутон спиране генерация-
та на неограничената последователност

out port oportRngBit = XS1_PORT_1A;  // порт за извеждане на ге-
нерираната последователност
out port oportSync   = XS1_PORT_1B;  // изх. строб за генериране
на последователността
in port  iportSync   = XS1_PORT_1F;  // вх. строб за генериране
```

```
на последователността

out port oportLed    = XS1_PORT_4F;  // порт на светодиодната
индикация
int intLed = 0;

#define NN  1024  // размерност на масива с извадката
const int M = 32;  // дължина на регистъра
const int N = NN;  // размер на извадката в words

#define NN_AM 8192  // размерност на масива в Алгоритъм-М
UINT uintDelay[NN_AM];  // работен масив в Алгоритъм-М

const int LED_PERIOD = 12500000;  // период на вкл/изкл на свето-
диода: (10^8 1/sec) * (0.125 sec)

UINT uintShiftReg = 1;  // LFSR - глобално състояние на
генератора
UINT uintRandomArr[NN];  // масив за генерирана последователност

void taskP(chanend chanRight);
void taskQ(chanend chanLeft);
void taskL(void);

void Randomize(void);
UINT RNG_LFSR_F(UINT uintSeed, UINT uintPoly);
UINT RNG_LFSR_G(UINT uintSeed, UINT uintPoly);
UINT RNG_ALG_M(UINT uintSeed, UINT uintPoly);

int main(void)
{
  chan chanPC;

  oportSync <: 0;

  Randomize();

  par
  {
    taskP(chanPC);
    taskQ(chanPC);
    taskL();
  }

  oportLed <: 0;

  return 0;
}

void taskP(chanend chanRight)
{
  UINT uintMsg;
```

```
  int intButStop;

  // изчакване на старт
  iportButStart when pinseq(0) :> void;

  // Генериране на крайна последователност
  for(int i = 0; i < N*M; i++)
  {
#if (RNG_TYPE == 1)
    uintMsg = RNG_LFSR_F(0, 0x80000057);

    chanRight <: uintMsg;
    oportSync <: 1;
#elif (RNG_TYPE == 2)
    uintMsg = RNG_LFSR_G(0, 0x80000057);

    chanRight <: uintMsg;
    oportSync <: 1;
#elif (RNG_TYPE == 3)
    uintMsg = RNG_ALG_M(0, 0x80000057);

    // разпакетиране
    for(int i = 0; i < M; i++)
    {
      chanRight <: (uintMsg & 0x1);
      uintMsg >>= 1;

      oportSync <: 1;
    }
#else
  #error INVALID RNG_TYPE
#endif
  }

  // Генериране на неограничена последователност
  while(TRUE)
  {
    iportButStop :> intButStop;
    if(intButStop == 0)
    {
      oportSync <: 0;

      // изчакване на повторен старт
      iportButStart when pinseq(0) :> void;
    }

#if (RNG_TYPE == 1)
    uintMsg = RNG_LFSR_F(0, 0x80000057);
    chanRight <: uintMsg;
    oportSync <: 1;
#elif (RNG_TYPE == 2)
    uintMsg = RNG_LFSR_G(0, 0x80000057);
```

```
    chanRight <: uintMsg;
    oportSync <: 1;
#elif (RNG_TYPE == 3)
    uintMsg = RNG_ALG_M(0, 0x80000057);

    // разпакетиране
    for(int i = 0; i < M; i++)
    {
      chanRight <: (uintMsg & 0x1);
      uintMsg >>= 1;

      oportSync <: 1;
    }
#else
  #error INVALID RNG_TYPE
#endif
  }
}

void taskQ(chanend chanLeft)
{
  int i, j;
  UINT uintMsg;

  for (i = 0; i < N; i++)
  {
    // пакетиране
    for (j = 0; j < M; j++)
    {
      chanLeft :> uintMsg;
      oportRngBit <: uintMsg;

      uintMsg = uintMsg << 31;
      uintRandomArr[i] = uintMsg | (uintRandomArr[i] >> 1);
    }
  }

  while(TRUE)
  {
    chanLeft :> uintMsg;
    oportRngBit <: uintMsg;
  }
}

void taskL(void)
{
  timer timerT;
  int intT;

  timerT :> intT;
  intT += LED_PERIOD;
```

```
  intLed = 0;
  oportLed <: intLed;

  while(TRUE)
  {
    // изчакване на вх. строб
    iportSync when pinseq(1) :> void;

    intLed = ~intLed;
    oportLed <: (intLed & 0x1);

    timerT :> intT;
    intT += LED_PERIOD;
    timerT when timerafter(intT) :> void;
  }
}

void Randomize(void)
{
  timer timerSeed;
  UINT uintSeed;
  timerSeed :> uintSeed;

  // инициализация на LFSR
  RNG_LFSR_G(uintSeed, 0x80000057);

  // инициализация на Алгоритъм-М
  srand(uintSeed);
  for(int i = 0; i < NN_AM; i++)
  {
    uintDelay[i] = rand();
  }
}

UINT RNG_LFSR_F(UINT uintSeed, UINT uintPoly)
{
  UINT uintMSB;
  UINT uintShiftRegCopy = uintShiftReg;

  if (uintSeed > 0)
    uintShiftReg = uintSeed;

#if (LFSR_F_VER == 1)
  uintMSB = (uintShiftReg >> 31) ^ (uintShiftReg >> 6) ^ (uint-
ShiftReg >> 4)
      ^ (uintShiftReg >> 2) ^ (uintShiftReg >> 1) ^ uintShiftReg;
#elif (LFSR_F_VER == 2)
  uintMSB = 0;

  for(int i = 0; i < (M - 1); i++)
  {
    if((uintPoly & 0x00000001) == 1)
```

```
      uintMSB ^= uintShiftRegCopy;

    par
    {
      uintShiftRegCopy >>= 1;
      uintPoly >>= 1;
    }
  }
#else
  #error INVALID LFSR_F_VER
#endif

  uintMSB = (uintMSB & 0x00000001) << 31;
  uintShiftReg = uintMSB | (uintShiftReg >> 1);

  return uintShiftReg & 0x00000001;
}

UINT RNG_LFSR_G(UINT uintSeed, UINT uintPoly)
{
  UINT uintResult;

  if (uintSeed > 0)
    uintShiftReg = uintSeed;

  if(uintShiftReg & 0x00000001)
  {
    uintShiftReg = 0x80000000 | ((uintShiftReg ^ uintPoly) >> 1);
    uintResult = 1;
  }
  else
  {
    uintShiftReg >>= 1;
    uintResult = 0;
  }

  return uintResult;
}

UINT RNG_ALG_M(UINT uintSeed, UINT uintPoly)
{
  UINT uintInx, uintRandomBit, uintResult;

  // пакетиране
  uintInx = 0;
  for(int i = 0; i < M; i++)
  {
    uintRandomBit = RNG_LFSR_F(0, 0x80000057);
    uintRandomBit = uintRandomBit << 31;
    uintInx = uintRandomBit | (uintInx >> 1);
  }
```

```
  uintInx = uintInx % NN_AM;
  uintResult = uintDelay[uintInx];
  uintDelay[uintInx] = rand();

  return uintResult;
}
```