Wireless Communications Systems
Take Home Problem

Cody Schafer

November 11, 2011

1 Problem 1

1.1 a

SR output if $x_0 = 0$ will continually be 0 as the XOR of 0 and 0 is 0.

1.2 b

The state of SR after 31 shifts given an $x_0 = 7ffffff$ is $0x38000004$. This is verified via `./p1 0x7ffffff 31 31`

1.3 c

It is periodic. The period is 0x7fffffe, meaning that given a time t, $x_{t+0x7fffffe} = xt$.

1.4 d

This is verified via `./p1 0x7ffffff -1 -1`. Note that this takes longer to run than it should. The output is `7fffffe:  7ffffff -> 1 match`

```
#include <stdio.h>
#include <stdbool.h>
#include <stdint.h>
#include <stdlib.h>
#define __unused __attribute__((__unused__))

#define ARRAY_SIZE(A)  (sizeof(A) / sizeof((A)[0]))
#define BIT_N(A, N)  (((A) & (1 << (N))) >> (N))
#define MASK_N(N)     ((1ull << ((unsigned long long)(N)))
    - 1)

static uint32_t update_pn(uint32_t cur)
{
        return cur << 1 |
                (BIT_N(cur, 27) ^ BIT_N(cur, 30));
}

static unsigned long long arg_ull(char *arg)
{
        unsigned long long n = 0;
        if (arg[0] == '0' && arg[1] == 'x') {
                sscanf(arg, "%llx", &n);
        } else {
                sscanf(arg, "%llu", &n);
        }
        return n;
}

#define CHECK_ARGS(name, args) \
        check_args(argc, argv, name, ARRAY_SIZE(args),
            args)
static void check_args(int argc, char **argv, char *name,
                size_t arg_ct, char **args)
{
        if ((size_t)argc < (arg_ct + 1)) {
                fprintf(stderr, "usage: %s", argc?argv
                    [0]:name);

                size_t i;
                for(i = 0; i < arg_ct; i++)
                        fprintf(stderr, " <%s>", args[i])
                            ;
                fputc('\n', stderr);

                exit(EXIT_FAILURE);
        }
}

int main(int argc, char **argv)
{
        unsigned long long iter = 0, show_every = 0;
        uint32_t pn_sr = 0, pn_sr_orig = 0;

        CHECK_ARGS("./p1",
                ((char *[]){"initial", "iterations", "
                    show_every"}) );
        pn_sr = arg_ull(argv[1]);
        iter  = arg_ull(argv[2]);
        show_every = arg_ull(argv[3]);
        pn_sr_orig = pn_sr;
        unsigned long long i;
        for(i = 0; i < iter; i++) {
                pn_sr = update_pn(pn_sr);
                bool out = BIT_N(pn_sr, 31);
                pn_sr = pn_sr & MASK_N(31);

                bool match = pn_sr == pn_sr_orig;
                bool show  = ((i + 1) % show_every) == 0;

                if (show || match) {
                        printf("%llx: %lx -> %x\n",
                                       i,
                                       (unsigned long)
                                           pn_sr,
                                       out);
                }

                if (match) {
                        printf("match\n");
                        return 0;
                }
        }

        return 0;
}
```

2 Problem 2

2.1 a

2.2 b

2.3 c

2.3.1 Code listings impliment a cdma decoder

Listing 1: config.h

```
/* R: "Finally, your messages will be coded in 7-bit
   ASCII format (MSB first, left to right)." */
#define ACSII_CHAR_BITS 7
#define ACSII_RECV_MSB_FIRST 0
```

```
/* R: "That is, using seed Sm , the first 256
   bits (ordered LSB to MSB) output by the shift register
        will comprise sm1 ,
   the next sm2 and so on."
 *
 * What does it mean for the bits to be orderd "[from]
    LSB to MSB"?
 * Is the LSB the first emitted? Or the MSB?
 */
#define SHIFT_REG_LSB_EMIT_FIRST 0

/* This should do the same thing as changing the above */
#define FLIP_RK 0

/* R: During bit interval k, user m has codeword smk
   composed of  1s.
 *
 * binary = {1, 0} instead. */
#define CODEWORD_IS_BINARY 0
```

Listing 2: p2.c

```
#include "config.h"

#include <stdio.h>
#include <stdbool.h>
#include <stdint.h>
#include <stdlib.h>

#define __unused __attribute__((__unused__))

#define ARRAY_SIZE(A) (sizeof(A) / sizeof((A)[0]))
#define BIT_N(A, N) (((A) & (1ull << (N))) >> (N))
#define MASK_N(N)    ((1ull << ((unsigned long long)(N)))
   - 1)

static uint32_t update_pn(uint32_t cur)
{
        return ((cur << 1) |
                (BIT_N(cur, 27) ^ BIT_N(cur, 30))) &
                   MASK_N(31);
}

struct sr {
        uint32_t pn;
        uint8_t  codeword[256/8];
};
```

```
static bool shift_in_bit(uint8_t *ar, size_t len, bool in
    )
{
        size_t i = len;
        if (!len)
                return false;

        bool ret = BIT_N(ar[len - 1], 7);

        for(; i > 1; i--) {
                ar[i - 1] = ar[i - 1] << 1 | (ar[i - 2]
                    >> 7);
        }

        ar[0] = ar[0] << 1 | in;

        return ret;
}

static void update_sr(struct sr *s)
{
        uint32_t new_pn = update_pn(s->pn);
        bool     out_bit = BIT_N(s->pn, 30);

        shift_in_bit(s->codeword, ARRAY_SIZE(s->codeword)
            , out_bit);

        s->pn = new_pn;
}

static void next_codeword(struct sr *s)
{
        size_t i;
        for(i = 0; i < 256; i++) {
                update_sr(s);
        }
}

static unsigned long long arg_ull(char *arg)
{
        unsigned long long n = 0;
        if (arg[0] == '0' && arg[1] == 'x') {
                sscanf(arg, "%llx", &n);
        } else {
                sscanf(arg, "%llu", &n);
        }
        return n;
}

#define CHECK_ARGS(name, args) check_args(argc, argv,
    name, ARRAY_SIZE(args), args)
static void check_args(int argc, char **argv, char *name,
     size_t arg_ct, char **args)
{
        if ((size_t)argc < (arg_ct + 1)) {
                fprintf(stderr, "usage: %s", argc?argv
                    [0]:name);

                size_t i;
                for(i = 0; i < arg_ct; i++)
                        fprintf(stderr, " <%s>", args[i])
                            ;
                fputc('\n', stderr);

                exit(EXIT_FAILURE);
        }
}

static long long next_rk(FILE *f)
{
        long long ret;
        if (fscanf(f, "%lld ", &ret) != 1) {
                fprintf(stderr, "your input file is
                    broken\n");
                exit(EXIT_FAILURE);
        }
        return ret;
}



static bool bit_in_array(uint8_t *ar, size_t bit)
{
        return BIT_N(ar[bit / 8], bit % 8);
}

#define type_ptr(t) ((t *) NULL)
#define sizeof_field(s, f) sizeof(((s *)(NULL))->f)
```

```
static long long array_mult_by_bit(long long *a, uint8_t
    *b, size_t elems)
{
        /* for a: elems = number of long longs.
         * for b: elems = number of bits.
         */
        long long sum = 0;

        size_t i;
        for(i = 0; i < elems; i++) {
#if SHIFT_REG_LSB_EMIT_FIRST
                size_t bit_ix = i;
#else
                size_t bit_ix = elems - i - 1;
#endif
                bool bit = bit_in_array(b, bit_ix);

                long long s_m_k = bit ? 1 : -1;

                long long r_k   = a[i];

                sum += s_m_k * r_k;
        }

        return sum;
}

#define print_array(a, elems, fmt, out) do {      \
        size_t i;                         \
        for (i = 0; i < elems; i++) {     \
                fprintf(out, fmt, a[i]);          \
        }                                 \
        fputc('\n', out);                 \
} while(0)

int main(int argc, char **argv)
{
        CHECK_ARGS("./p2", ((char *[]){"data_file", "
            initial"}) );

        FILE *in = fopen(argv[1], "r");

        unsigned long long pi = arg_ull(argv[2]);

        struct sr sr = { pi , {0} };
        long long r[256] = {};

        char b = 0;

        size_t i = 0;
        size_t j = 0;

        while(!feof(in)) {
                /* r_k = sum(m=1, M, a_m_k * b_m_k *
                   s_m_k)
                 * b_m_k = 1 | r_k s_m_k > 0, otherwise
                    -1
                 */
                long long r_k = next_rk(in);
#if RK_FLIP
                r[256 - j - 1] = r_k;
#else
                r[j] = r_k;
#endif

                j++;
                if ((j % 256) == 0) {
                        j = 0;
                        next_codeword(&sr);
                        //print_array(sr.codeword,
                            ARRAY_SIZE(sr.codeword), "%x",
                             stderr);

                        long long b_m_k =
                            array_mult_by_bit(r, sr.
                            codeword, 256);

#if ACSII_RECV_MSB_FIRST
                        size_t ix = ACSII_CHAR_BITS - i -
                             1;
#else
                        size_t ix = i;
#endif

                        b |= (b_m_k > 0) << ix;

                        i ++;

                        if ((i % ACSII_CHAR_BITS) == 0) {
                                printf("%c", b);
                                b = 0;
                        i = 0;
                    }
                }
            }

            return 0;
}
```

2.3.2 The obtained output

```
"Cody: As early as 1936, SoLzhenitsyn was developing the characters and
concepts for a planned epic work on the First World War and the Russian
Sevolution. This eventually led to the novel August 1914 b some of
the chapters he grote then still survive.[citation needed] Solzhenitsyn
studied mathematics at Rostov State Universit{. At the same time he
took correspondence courses from the Moscow Institute of Philosophy,
Literature and History, at this time heavily ideological in sco‘e. As
he himself makes clear, he did not question the state ideology or the
superiority of the Soviet Uniol untiL he spent time an the camps.... "
```