Dongle: Sistem Otentikasi Software Secara Hardware

Agust Isa Martinus
Prodi Teknik Informatika, Universitas Muhammadiyah Cirebon
agust.isa@umc.ac.id

Abstrak

Hak atas Kekayaan Intelektual (HKI atau HaKI) dewasa ini semakin digandrungi dan digalakkan oleh banyak kalangan. Perlindungan atas hasil karya yang berupa perangkat lunak atau program komputer dapat dilakukan dengan banyak cara, seperti menggunakan nomor seri baik berdiri sendiri maupun melalui internet, pembatasan regional penggunaan program, sampai penggunaan perangkat keras khusus untuk proteksi dan otentikasi perangkat lunak yang syah. Perangkat keras khusus untuk proteksi dan otentikasi perangkat lunak tersebut di kalangan penggunanya disebut Dongle atau sering juga digunakan terminologi hardware key, hardware token, atau security device.

Dalam tulisan ini, dibahas dan dirancang Dongle yang menggunakan metode enkripsi data. Dongle menerima sejumlah data, melakukan enkripsi, kemudian mengirimkan hasil enkripsi tersebut. Algoritma enkripsi yang digunakan adalah yang relatif mudah dan murah untuk diterapkan menggunakan komponen perangkat keras PLD (Programmable Logic Device) yang berkapasitas tidak terlalu besar (FPGA atau CPLD dengan kapasitas kecil), yaitu XOR dengan dua kunci dan transposisi bit untuk masukan 16-bit data polos (plain text) dan keluaran 16-bit data terenkripsi (ciphered text). Hasil rancangan-rancangan yang dibuat, diimplementasi menggunakan Very HSIC Description Language (VHDL).

Hasil rancangan yang sudah diimplementasikan, yaitu Sistem Penerima Data dan Sistem Pengirim Data, telah berhasil berdasarkan simulasi yang dilakukan, menggunakan ModelSim SE 6.0. Rancangan Dongle yang dibuat di sini, transposisi bit yang digunakan bersifat statis. Untuk lebih meningkatkan keamanan atau kerumitan enkripsi sederhana yang digunakan, dapat diterapkan transposisi bit dinamis, yaitu transposisi bit berubah tergantung pada keadaan saat itu dan sebelumnya atau pada fungsi yang digunakan, atau yang lainnya.

Kata kunci: *dongle, enkripsi, chipered text, otentikasi, security*.

PENDAHULUAN

Hak atas Kekayaan Intelektual (HKI atau HaKI) dewasa ini semakin digandrungi dan digalakkan oleh banyak kalangan. Meskipun tidak pernah tuntas perseteruan antara kubu yang pro dan kontra, tetapi fakta yang sudah terjadi adalah Pemerintah Indonesia melalui Departemen Hukum dan Hak Asasi Manusia sudah menyetujui dengan mengeluarkan undang-undang mengenai HaKI, Perlindungan atas hasil karya yang berupa perangkat lunak atau program komputer termasuk ke dalam pengaturan **UNDANG-UNDANG REPUBLIK INDONESIA NOMOR 19 TAHUN 2002, TENTANG HAK CIPTA**.

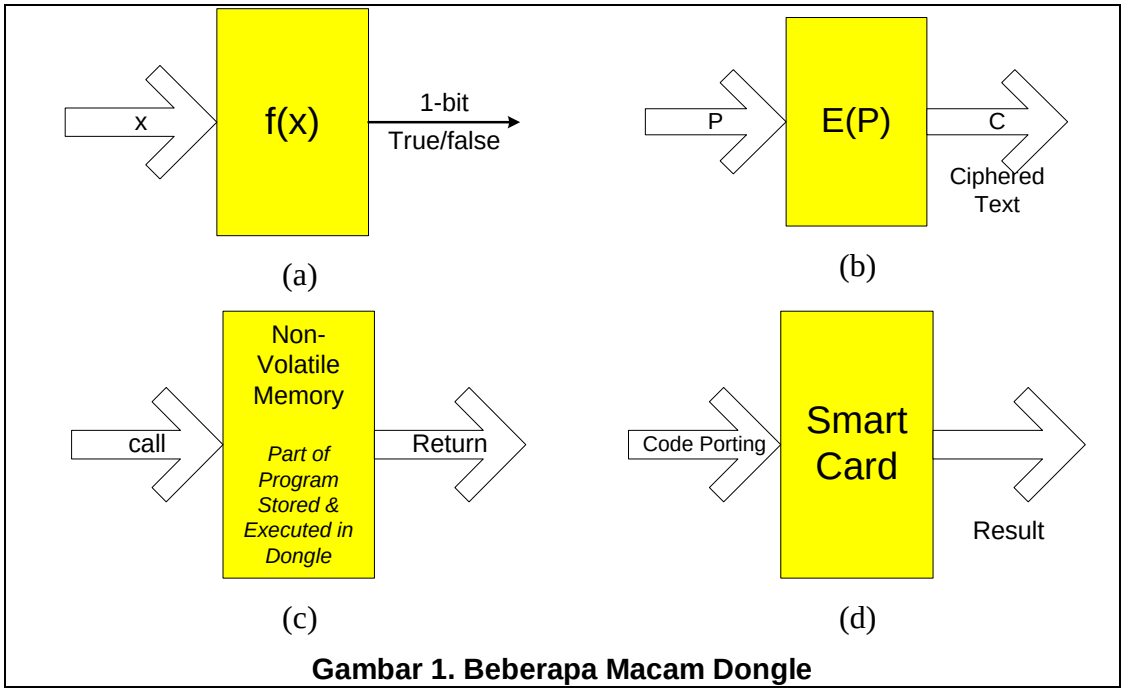
Perlindungan atas hasil karya yang berupa perangkat lunak atau program komputer dapat dilakukan dengan banyak cara, seperti menggunakan nomor seri baik berdiri sendiri maupun melalui internet, pembatasan regional penggunaan program, sampai penggunaan perangkat keras khusus untuk proteksi dan otentikasi perangkat lunak yang syah. Perangkat keras khusus untuk proteksi dan otentikasi perangkat lunak tersebut di kalangan penggunanya disebut Dongle. Dalam kesempatan ini, penulis akan mencoba merancang satu jenis Dongle dengan metode yang sudah dikenal.

Cara-cara perlindungan perangkat lunak atau program komputer tersebut ada yang relatif mudah sulit dipecahkan sampai dengan yang sulit atau membutuhkan waktu lama untuk memecahkannya. Sudah banyak orang berhasil membuat pemecah kode atau pembangkit nomor seri untuk perangkat lunak tertentu yang dilindungi. Hal tersebut didukung dengan (lebih) mudahnya menggandakan perangkat lunak dibandingkan menggandakan perangkat keras. Di sinilah Dongle, sebagai sistem perlindungan secara perangkat keras untuk otentikasi perangkat lunak atau program komputer, diharapkan dapat lebih berperan.

DONGLE

Penyebutan dengan nama ‘Dongle’ lebih terkenal dari pada terminologi atau nama lain yang digunakan oleh pembuat atau penyedia alat tersebut. Para pembuat atau penyedia alat proteksi untuk otentikasi perangkat lunak, menyebut alat ini dengan terminologi *hardware key*, *hardware token*, atau *security device*.

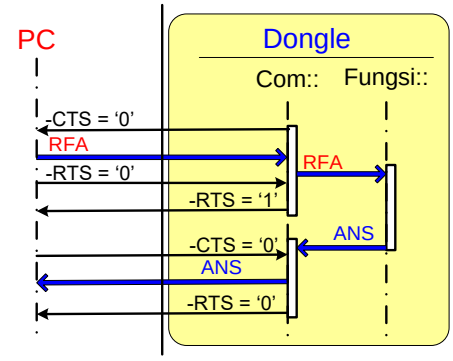
Dongle sudah mengalami beberapa evolusi, dari awal mula yang hanya rangkaian pasif dalam konektor khusus sampai kini yang berupa *smart card* yang dapat menyimpan bagian



Gambar 1. Beberapa Macam Dongle

program dan memproses algoritma enkripsi yang canggih. Beberapa jenis umum Dongle yang sudah dikenal dapat dilihat pada Gambar 1.

Dalam tulisan ini, akan dibahas dan dirancang Dongle seperti pada Gambar 1.b, yaitu menggunakan metode enkripsi data. Dongle menerima sejumlah data, melakukan enkripsi, kemudian mengirimkan hasil enkripsi tersebut. Sedangkan Gambar 1.a adalah implementasi Dongle yang paling naif, menerima data masukan, mengolahnya ke dalam suatu fungsi f(x), kemudian mengeluarkan satu bit yang menyatakan "betul" atau "salah" hasil pengolahan data yang diterimanya tersebut. Gambar 1.c adalah Dongle yang memiliki *non-volatile memory*



Gambar 2. Proses Transfer Data Dongle–PC

(tetap menyimpan data walaupun tanpa sumber daya listrik) yang dapat diisi (di pabrik/pembuat program) dan melaksanakan bagian dari program komputer, kemudian mengeluarkan hasilnya untuk bagian program yang berjalan di dalam komputer. Gambar 1.d adalah Dongle yang menggunakan *smart card* yang dapat diisi (*ported*) dan melaksanakan bagian dari program komputer, juga melaksanakan algoritma enkripsi yang canggih, kemudian mengeluarkan hasilnya untuk bagian program yang berjalan di dalam komputer.

PERANCANGAN

Dongle, yang akan dibuat, dirancang untuk dipasang pada Komputer Pribadi (PC, *Personnel Computer*) pada *port* serial RS232 level dengan komunikasi asinkron. Dongle mengenkripsi data yang diterima dari luar, kemudian hasil enkripsi tersebut dikirimkan ke luar. Algoritma enkripsi yang digunakan adalah yang relatif mudah dan murah untuk diterapkan menggunakan komponen perangkat keras PLD (*Programmable Logic Device*) yang berkapasitas tidak terlalu besar (FPGA atau CPLD dengan kapasitas kecil), yaitu XOR dengan dua kunci dan transposisi bit untuk masukan 16-bit data polos (*plain text*) dan keluaran 16-bit data terenkripsi (*ciphered text*). Selain itu, dengan metoda enkripsi ini didapat kecepatan yang relatif tinggi, hanya dibatasi oleh kecepatan PLD dan komunikasi asinkron yang digunakan, kecil sekali pengaruh algoritma. Dengan metoda transposisi bit untuk 16-bit data, didapatkan kemungkinan hasil atau ciphered text sebanyak, $2^{16} = 65536$ kemungkinan. Tingkat kerahasiaan atau keamanan semakin bertambah dengan digunakannya dua kunci untuk XOR data masukan.

Proses komunikasi data antara PC dengan Dongle yang dirancang, sebagai berikut:

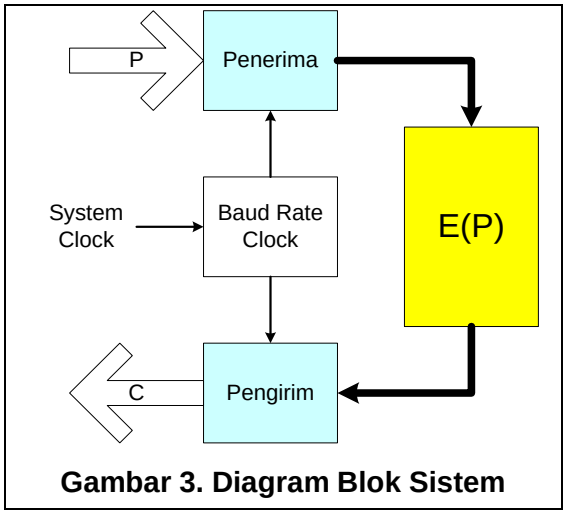
A. Permintaan otentikasi dari PC ke Dongle

i. PC melihat status –CTS (*Clear To Send*, aktif '0'), tunggu sampai –CTS = '0' yang menandakan Dongle siap menerima data.
ii. PC mengirim dua byte (B1, B0) data serial asinkron sebagai RFA (*Request For Authentication*).
iii. Reset –RTS, siap menerima data balasan.
iv. Menerima dua byte ANS dari Dongle sebagai balasan.

v. Mengeksekusi fungsi dekripsi; menverifikasi dua byte ANS tersebut.
vi. Kembali ke langkah A.i.

B. Jawaban dari Dongle ke PC

i. Setelah menerima dua byte data dari PC, kemudian Dongle menset –RTS (Request To Send, aktif '0') menjadi '1' yang menandakan Dongle sedang sibuk dan sementara tidak bisa melayani pengiriman data.
ii. Mengeksekusi fungsi enkripsi.
iii. Mencek –CTS, tunggu sampai –CTS = '0' yang menandakan PC siap menerima data.
iv. Kirim balasan berupa dua byte (Q1, Q0) ANS (Answer) sebagai jawaban.

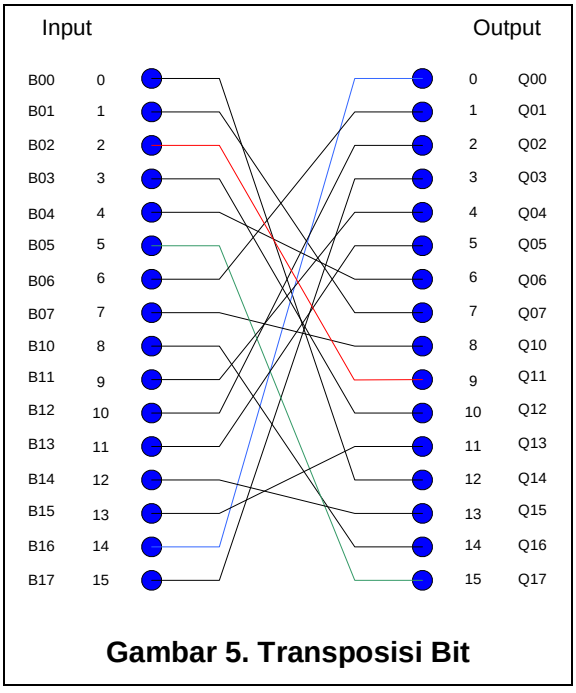


Gambar 3. Diagram Blok Sistem

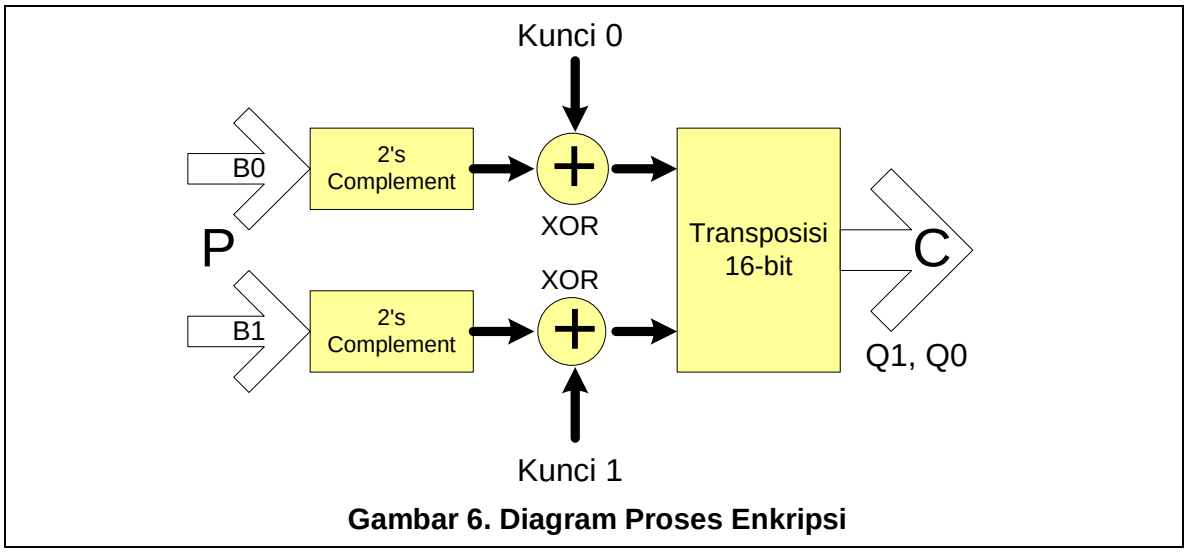
v. Reset –RTS, siap untuk menerima dua byte RFA berikutnya.
vi. Kembali ke langkah B.i.

Setting komunikasi serial asinkron yang digunakan adalah data 8N1 dan kecepatan 8192 bps (*bit per second*) menggunakan *handshaking* RTS–CTS dengan frekuensi *clock* atau kristal yang digunakan 32768 kHz.

Dongle yang dibangun terdiri dari empat bagian, yaitu: *Enkripsi*, *Penerima*, *Pengirim*, dan *Baud Rate Clock Generator*. Bagian Enkripsi bertugas menyandikan data-data masukan. Bagian Penerima dan Pengirim melakukan komunikasi data dengan Komputer PC. Dan, bagian Baud Rate Clock Generator menghasilkan *clock* (dari *system clock*) yang digunakan oleh bagian Penerima dan Pengirim dalam proses komunikasi data dengan Komputer PC. Blok Diagram Sistem ditunjukkan pada Gambar 3.



Gambar 5. Transposisi Bit



Gambar 6. Diagram Proses Enkripsi

Enkripsi

Dua byte ANS jawaban dari Dongle merupakan hasil dari pengolahan dua byte masukan (RFA yang terdiri dari B1 dan B0) pada satu fungsi enkripsi sebagai berikut:

$$Q1, Q0 = E(B1, B0)$$

Dalam bagian Enkripsi ini, pada byte masukan B1 dan B0 tersebut pertama dilakukan 2's complement secara individu, kemudian masing-masing di-XOR dengan kunci yang berbeda, dan terakhir dilakukan transposisi 16-bit. Walaupun pada tahap pertama dan kedua, proses 2's complement dan XOR dilakukan untuk masing-masing byte, tetapi pada tahap akhir dilakukan transposisi 16-bit sekaligus untuk kedua byte tersebut sehingga urutan byte sangat berpengaruh. Urutan byte berbeda akan memberikan hasil keluaran yang sangat berbeda dan relati sulit untuk diterka. Seseorang harus mengetahui kunci yang digunakan dan susunan transposisi bit yang diterapkan.

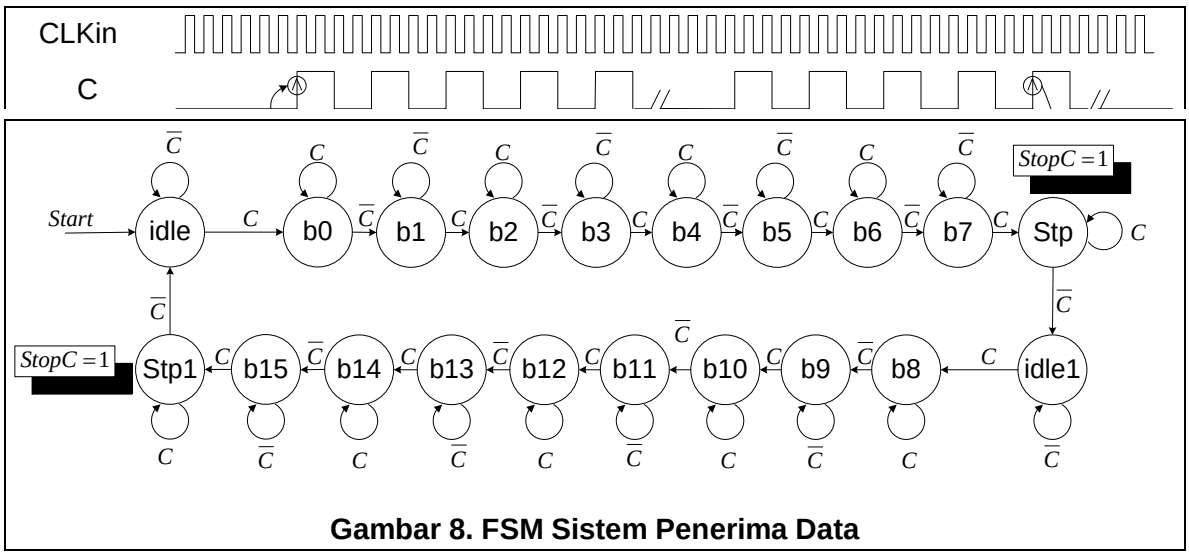
Dalam persamaan lebih rinci, transposisi tiap bit seperti ditunjukkan pada Gambar 5 adalah:

Q00 = B16,	Q10 = B07,
Q01 = B06,	Q11 = B02,
Q02 = B12,	Q12 = B03,
Q03 = B17,	Q13 = B15,
Q04 = B11,	Q14 = B00,
Q05 = B13,	Q15 = B14,
Q06 = B04,	Q16 = B10,
Q07 = B01	Q17 = B05

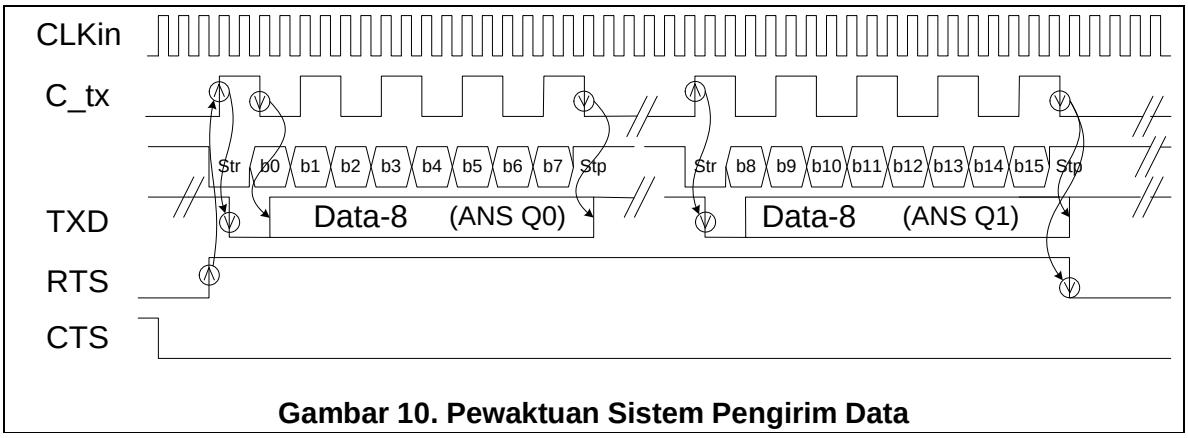
Keterangan:

B00 – B07 adalah B0 bit-0 – bit-7.
B10 – B17 adalah B1 bit-0 – bit-7.
Q00 – Q07 adalah Q0 bit-0 – bit-7.
Q10 – Q17 adalah Q1 bit-0 – bit-7.

Dengan metoda transposisi bit 16-bit, didapatkan kemungkinan transposisi sebanyak, $2^{16} = 65536$ kemungkinan. Gambar 5 adalah transposisi 16-bit yang digunakan pada



Gambar 8. FSM Sistem Penerima Data



Gambar 10. Pewaktuan Sistem Pengirim Data

perancangan Dongle dalam tulisan ini yang merupakan salah satu kemungkinan transposisi 16-bit tersebut. Perancang lain dapat memilih transposisi yang lain atau membuat transposisi yang dinamis.

Penerima 8-bit Data UART

Sistem Penerima Data dirancang untuk berkomunikasi serial asinkron dengan kecepatan 8192 bps menggunakan handshaking RTS–CTS. Satu text-frame, mulai dari start-bit, data, sampai stop-bit dan tanpa parity-bit berjumlah 10-bit dengan seting data 8N1 (8-data, No Parity, dan 1-bit stop).

Awal kerja Sistem Penerima Data ialah mendeteksi ‘start-bit’ pada jalur sinyal RXD. Jika menerima ‘start-bit’, pembangkit clock untuk Sistem Penerima Data mulai aktif. Berdasarkan clock tersebut, Sistem Penerima Data akan membaca data yang dikirim dari PC. Karena fungsi enkripsi digabungkan dalam sistem ini, maka bersamaan dengan diterimanya data, sekaligus juga dilakukan transposisi data. Jika data yang diterima dan dienkripsi sudah lengkap, Sistem Penerima Data akan menset sinyal RTS menjadi ‘1’ yang menandakan sistem sibuk dan tidak siap menerima data, kemudian beralih ke Sistem Pengiriman Data. Diagram pewaktuan Sistem Penerima Data dan Enkripsi ditunjukkan pada **Error! Reference source not found.**.

Rancangan Sistem Penerima Data dan Enkripsi yang dibuat berdasarkan diagram pewaktuan **Error! Reference source not found.** dan ditulis di sini, dapat di lihat pada Gambar 8. Pada rancangan tersebut dapat diamati bahwa sistem mulai aktif jika mendapatkan sinyal clock C untuk penerimaan. Sinyal clock C ini dibangkitkan oleh sistem pada Gambar 11.

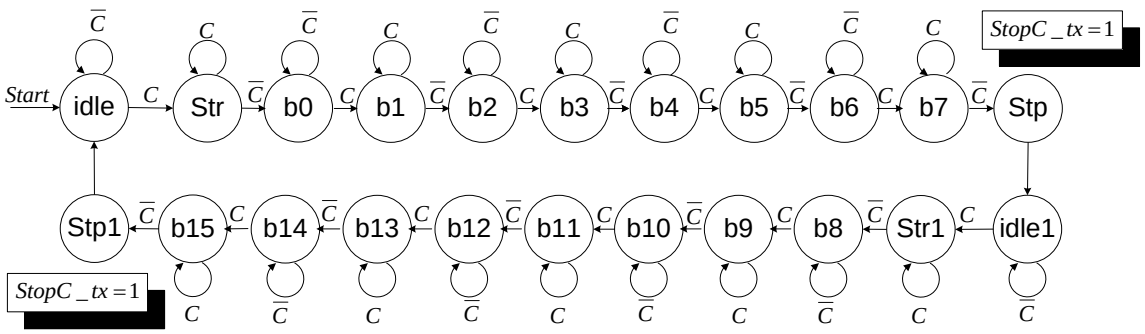
Pengirim 8-bit Data UART

Rancangan Sistem Pengirim Data ditunjukkan pada Gambar 10. Sistem mulai aktif jika dipicu oleh sinyal *clock* C_tx. Pertama sistem mengirimkan ‘start-bit’=’0’ yang menandakan awal dari text-frame yang akan dikirim. Selanjutnya data yang ada di register internal akan dikirim, bit-per-bit dan diakhiri satu ‘stop-bit’ untuk text-frame tersebut. Jika semua data, 2-byte ANS sudah terkirim, kemudian sistem menurunkan sinyal RTS menjadi ‘0’ yang menandakan pengiriman selesai dan siap menerima data berikutnya dari PC. Pewaktuan untuk Sistem Pengirim Data ditunjukkan pada Gambar 10.

Rancangan untuk Sistem Pengirim Data berdasarkan diagram pewaktuan pada Gambar 10, ditunjukkan pada diagram keadaan terbatas Gambar 9.

Baud Rate Clock Generator

Kecepatan transfer data sistem pengirim dan penerima, keduanya sama, yaitu 8192 bps. Frekuensi clock yang digunakan sebagai *clock-in hardware* CPLD atau FPGA adalah



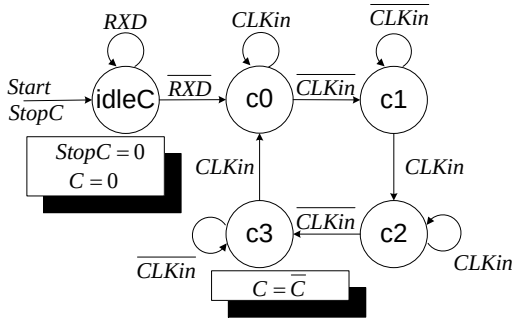
Gambar 9. FSM Sistem Pengirim Data

32768 kHz. Untuk mendapatkan kecepatan data 8192 bps menggunakan *system clock* 32768 kHz, dirancang pembagi 4 sebagai pembangkit *clock* untuk *baud rate generator*. Rancangan pembangkit clock untuk Sistem Penerima Data ditunjukkan pada Gambar 11. Sistem Pembangkit Clock untuk Penerima mulai aktif jika mendapatkan 'start-bit' pada RXD.

Rancangan Sistem Pembangkit Clock untuk Sistem Pengirim Data sama dengan pembangkit *clock* untuk penerima. Perbedaannya adalah sinyal pemicu masing-masing, Pada pembangkit *clock* untuk pengirim data, pemicunya adalah sinyal RTS dari sistem penerima di dalam Dongle sendiri. Sinyal RTS yang menjadi '1' menandakan Dongle akan mengirim data yang terenkripsi sebagai jawaban ke PC. Rancangan Sistem Pembangkit Clock untuk Sistem Pengirim Data ditunjukkan pada Gambar 12.

IMPLEMENTASI

Hasil rancangan-rancangan yang sudah dibuat dan ditunjukkan pada bagian-bagian sebelumnya, diimplementasi menggunakan *Very HSIC Description Language* (VHDL). Implementasi menggunakan VHDL tersebut ditunjukkan pada penggalan-penggalan program berikut dalam bagian tulisan ini.



Gambar 11. FSM Pembangkit Clock untuk Sistem Penerima Data

Perpindahan Keadaan

Semua perpindahan keadaan (*state transition*) dalam rancangan yang dibuat, selaras dengan *clock system*. Sinkronisasi tersebut didapat menggunakan implementasi program yang ditunjukkan pada Program 1.

```
PROCESS(Clock)
BEGIN
  IF Clock'event AND (Clock='1') THEN   -- Synchronous FSM
    pres_crx <= next_crx; --
    pres_ctx <= next_ctx; --
    pres_rx <= next_rx;   --
    pres_tx <= next_tx;   --
  END IF;
END PROCESS;
```

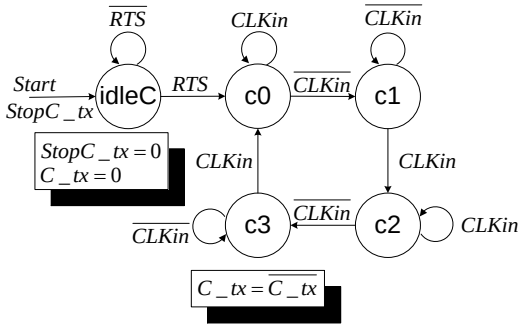
Program 1. Transfer State Tersinkronisasi

Pembangkit Clock untuk Penerima dan Pengirim

Pembangkit *clock* 8192 Hz untuk Sistem Penerima Data diimplementasi pada Program 2.

```
PROCESS(Clock)
BEGIN
```

```
  IF stopC='1' THEN
    next_crx<=idleC;
  END IF;
```



Gambar 12. FSM Pembangkit Clock untuk Sistem Pengirim Data

```
  CASE pres_ crx IS
    WHEN idleC =>
      C <= '0';
      stopC <= '0';
      IF rxd='0' AND cts='1' THEN next_crx<=c0;
      ELSE next_crx<=idleC;
      END IF;
    WHEN c0 =>
      IF Clock='0' THEN                          next_crx<=c1;
      ELSE next_crx<=c0;
      END IF;
    WHEN c1 =>
      IF Clock='1' THEN                          next_crx<=c2;
      ELSE next_crx<=c1;
      END IF;
    WHEN c2 =>
      IF Clock='0' THEN                          next_crx<=c3;
      ELSE next_crx<=c2;
      END IF;
    WHEN c3 =>
      C <= NOT C;   --
      IF Clock='1' THEN                          next_crx<=c0;
      ELSE next_crx<=c3;
      END IF;
  END CASE;
END PROCESS;
```

Program 2. Pembangkit Clock untuk Sistem Penerima Data

Pembangkit clock 8192 Hz untuk Sistem Pengirim Data diimplementasi pada Program 3.

```
PROCESS(Clock)
BEGIN
  IF stopC_tx='1' THEN                           next_ctx<=idleC;
  END IF;

  CASE pres_ ctx IS
    WHEN idleC =>
      C <= '0';
```

```
        stopC_tx <= '0';
        IF cts='0' AND rts='1'                          THEN
next_ctx<=c0;
        ELSE next_ctx<=idleC;
        END IF;
      WHEN c0 =>
        IF Clock='0' THEN
 next_ctx<=c1;
        ELSE next_ctx<=c0;
        END IF;
      WHEN c1 =>
        IF Clock='1' THEN
 next_ctx<=c2;
        ELSE next_ctx<=c1;
        END IF;
      WHEN c2 =>
        IF Clock='0' THEN
 next_ctx<=c3;
        ELSE next_ctx<=c2;
        END IF;
      WHEN c3 =>
        C_tx <= NOT C_tx;    --
        IF Clock='1' THEN
 next_ctx<=c0;
        ELSE next_ctx<=c3;
        END IF;
   END CASE;
 END PROCESS;
```

Program 3. Pembangkit Clock untuk Sistem Pengirim Data

Sistem Penerima Data dan Enkripsi

Sistem Penerima Data diimplementasikan bersama dengan Enkripsi data yang ditunjukkan pada Program 4.

```
 stateRX_proc:
 PROCESS(pres_rx, C, cts)
 BEGIN
   CASE pres_rx IS
     WHEN idle =>
        rts <= '0';          --
        IF C='1' THEN   `                               next_rx <=
b0;
        ELSE next_rx <= idle;
        END IF;
      WHEN b0 =>
        Sreg1(4) <= rxd;     --
        IF C='0' THEN                                   next_rx <=
b1;
        ELSE next_rx <= b0;
        END IF;
...
... dari satate b0 s.d. state b7
... dilakukan transposisi untuk
... bit-0 s.d. 7
...
      WHEN stp =>
        stopC <= '1';
```

```
        IF C='0' THEN                                          next_rx <=
idle1;
        ELSE next_rx <= stp;
        END IF;
      WHEN idle1 =>
        IF C='1' THEN                                          next_rx <=
b8;
        ELSE next_rx <= idle1;
        END IF;
      WHEN b8 =>
        Sreg1(6) <= rxd;     --
        IF C='0' THEN                                          next_rx <=
b9;
        ELSE next_rx <= b8;
        END IF;
...
... dari satate b8 s.d. state b15
... dilakukan transposisi untuk
... bit-8 s.d. 15
...
 `    WHEN stp1 =>
        rts <= '1';  --
        stopC <= '1';
        IF C='0' THEN                                          next_rx <=
idle;
        ELSE next_rx <= stp1;
        END IF;
   END CASE;
 END PROCESS;
```

Program 4. Sistem Pengirim Data dan Enkripsi

Sistem Pengirim Data

Sistem Pengirim Data diimplementasikan pada Program 5.

```
 stateTX_proc:   -- kirim data ANS
 PROCESS(pres_tx, C_tx, cts, rts)
 BEGIN
   CASE pres_tx IS
     WHEN idle =>
       txd <= '1';          --
       IF C_tx='1' THEN                                        next_tx <=
str;
       ELSE next_tx <= idle;
       END IF;
     WHEN str =>
       txd <= '0';     --
       IF C_tx='0' THEN                                        next_tx <=
b0;
 `     ELSE next_tx <= str;
       END IF;
     WHEN b0 =>
       txd <= Sreg0(0);     --
       IF C_tx='1' THEN                                        next_tx <=
b1;
       ELSE next_tx <= b0;
       END IF;
...
```

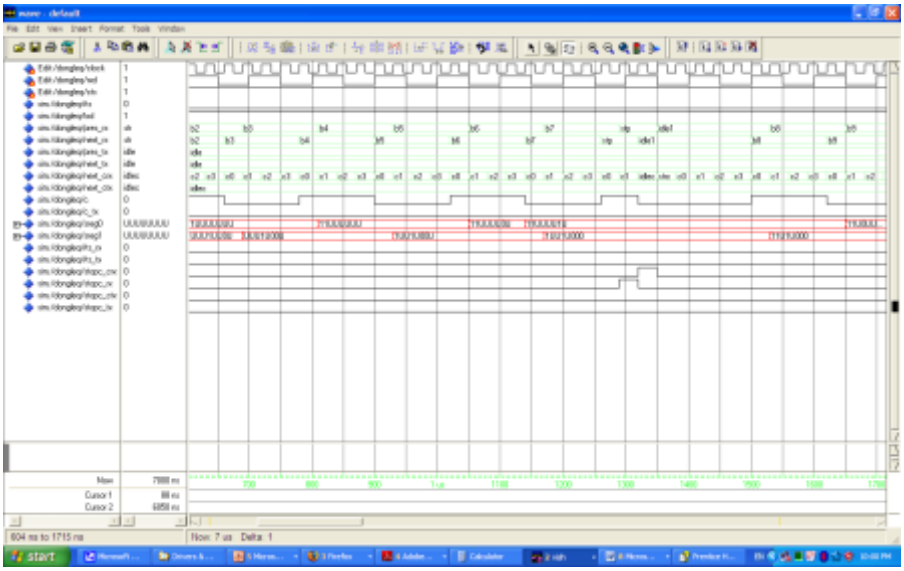
```
... dari satate b0 s.d. state b7
...
      WHEN stp =>
        stopC_tx <= '1';
        next_tx <= idle1;
      WHEN idle1 =>
        IF C_tx='1' THEN                                    next_tx <=
str1;
        ELSE next_tx <= idle1;
        END IF;
      WHEN str1 =>
        txd <= '0';     --
        IF C_tx='0' THEN                                    next_tx <=
b8;
        ELSE next_tx <= str1;
        END IF;
      WHEN b8 =>

        txd <= Sreg1(0);     --
        IF C_tx='1' THEN                                    next_tx <=
b9;
        ELSE next_tx <= b8;
        END IF;
...
... dari satate b8 s.d. state b15
...
      WHEN stp1 =>
        rts <= '0';  --
        stopC_tx <= '1';
        next_tx <= idle;
    END CASE;
 END PROCESS;
```

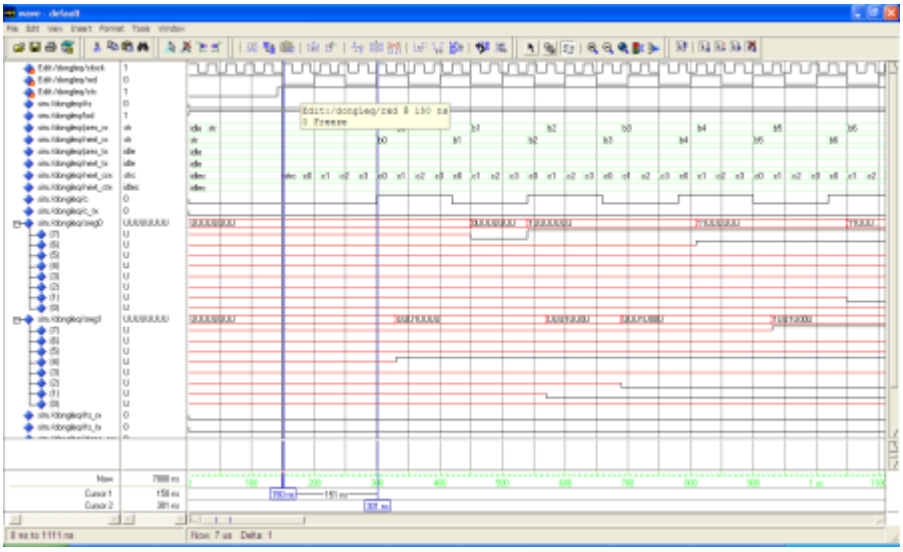
Program 5. Sistem Pengirim Data

SIMULASI

Hasil rancangan yang sudah diimplementasikan, kemudian disimulasikan untuk mengamati dan menguji beberapa kemungkinan yang sudah terduga, juga untuk mendapatkan beberapa kemungkinan yang belum terdeteksi. Dari hasil simulasi, jika didapati beberapa kelemahan, akan dilakukan perbaikan dan pengujian ulang sampai didapatkan hasil yang memadai. **Error! Reference source not found.** dan **Error! Reference source not found.** adalah gambar-gambar sinyal hasil simulasi terakhir dari Sistem Penerima Data dan Sistem Pengirim Data menggunakan ModelSim SE 6.0.



Gambar 14. Simulasi Sistem Penerima Data



Gambar 13. Simulasi Sistem Pengirim Data

PENUTUP

Rancangan Dongle yang dibuat dalam tulisan ini, transposisi bit yang digunakan bersifat statis. Untuk lebih meningkatkan keamanan atau kerumitan enkripsi sederhana yang digunakan, dapat diterapkan transposisi bit dinamis, yaitu transposisi bit berubah tergantung pada keadaan saat itu dan sebelumnya atau pada fungsi yang digunakan.

Jika kita sedang tidak ingin berbagi sesuatu, text file atau program komputer, kita dapat menggunakan metode enkripsi untuk melindunginya. Sesederhana apapun metode atau algoritma enkripsi yang kita gunakan, tetaplah diperlukan usaha lebih untuk memecahkannya dibandingkan tidak menggunakan sama sekali. Hanya dengan usaha yang sesuai, maka enkripsi tersebut dapat dipecahkan dan sudah selayaknya mereka yang berusaha ini mendapatkan hasilnya. Sekali lagi, sesederhana apapun algoritma enkripsi yang digunakan masih lebih baik dibandingkan tidak ada algoritma sama sekali.

PUSTAKA ACUAN

[1] Andrew S. Tanenbaum, "Network Security" dalam *Computer Networks*, New Jersey: Prentice Hall PTR, 2003.

[2] Hamilton Nickels, *Secret of Making and Breaking Codes*. New York: Carol Publishing Group, 1994.

[3] Michael Purser, "Security Management" dan "Algorithms" dalam *Secure Data Networking*, Massachusetts: Artech House, Inc., 1993.

[4] Stephen Brown dan Zvonco Vrenesic, *Fundamentals of Digital Logic with VHDL Design*, 2ed., New York: McGraw Hill, 2005.