PERCOBAAN 10.

PARITY GENERATOR DAN CHECKER

TUJUAN:

Setelah menyelesaikan percobaan ini mahasiswa diharapkan mampu

- Memahami prinsip kerja rangkaian *Parity Generator* dan *Parity Checker*
- Mendisain rangkaian *Parity Generator* dan *Checker* untuk fungsi Pengacakan data *(Data Scrambling)*

PERALATAN:

1. Logic Circuit Trainer ITF-02 / DL-02
2. Oscilloscope

TEORI:

1. PARITY GENERATOR DAN CHECKER

Dalam sistim transmisi digital, dimana urutan data biner dikirimkan dari pengirim ke penerima, sangat dimungkinkan terjadinya kesalahan (*error*) pada data yang diterima. Kesalahan ini biasanya disebabkan karena *external noise* (misalkan sinyal listrik atau suara yang ikut dalam data). Sebagai contoh, dikirimkan sinyal data BCD 5 (0101), kemudian pada proses transmisi, ada *noise* yang masuk sehingga mengubah nilai LSB ‘0’ menjadi ‘1’. Akibatnya di sisi terima, sinyal data yang masuk tadi dibaca sebagai 4 (0100). Data tersebut tentu salah, karena tidak sesuai dengan yang dikirimkan.

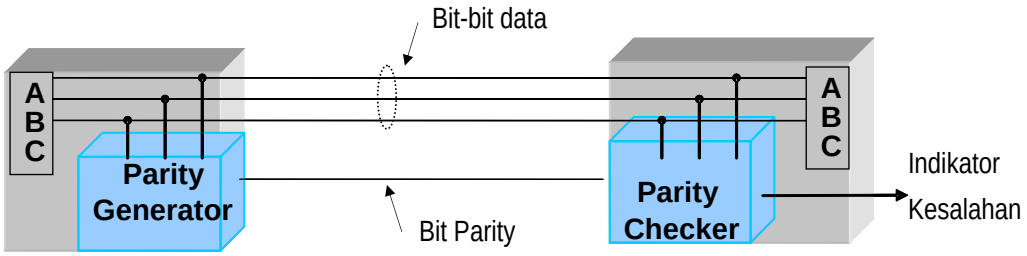
Untuk menghindari kesalahan data saat pengiriman, diberikan bit tambahan pada urutan data yang akan ditransmisikan. Bit tambahan ini dinamakan *bit parity*. Penambahan bit parity dilakukan di sisi kirim (*Transmitter*). Rangkaian pembangkit *bit parity* dinamakan *Parity Generator*. Jumlah *bit parity* bisa satu bit atau lebih. Berdasarkan jumlah bit biner ‘1’ dalam setiap kelompok, *bit parity* dibedakan menjadi 2 jenis : *Odd Parity Bit* dan *Even Parity Bit. Odd Parity bit* adalah bit tambahan yang diberikan untuk membuat jumlah bit ‘1’ pada urutan data yang disertainya menjadi ganjil, sedangkan *Even Parity Bit* adalah bit tambahan yang diberikan untuk membuat jumlah bit ‘1’ pada urutan data yang disertainya menjadi genap. Diberikan contoh sebagai berikut :

Urutan data : 1 0 1 1 0 1 1

Urutan data + Odd Parity Bit : 1 0 1 1 0 1 1 **0** ← Bit Parity

Urutan data + Even Parity Bit : 1 0 1 1 0 1 1 **1** ← Bit Parity

Parity Checker adalah rangkaian penge-cek nilai bit parity yang menyertai urutan data yang diterima. Rangkaian *Parity Checker* berada di sisi terima (*Receiver)*. Jenis bit parity yang di-cek harus sesuai dengan jenis bit parity di sisi kirim, bisa *Odd* atau *Even Parity*. Jika nilai cek setiap urutan data dan bit parity yang menyertainya adalah '0', maka urutan data dan bit parity tersebut benar. Jika bernilai '1' berarti ada kesalahan. Blok diagram *Parity Generator* dan *Parity Checker* ditunjukkan pada gambar 10-1.



Gambar 10-1. Blok Diagram Parity Generator dan Checker

Untuk mendisain rangkaian *Parity Generator*, perlu ditentukan lebih dulu jumlah data dalam setiap urutannya dan jenis bit parity yang akan digunakan. Sebagai contoh, akan dibuat urutan data 3 bit biner, yang disertai 1 *Even Parity Bit*. Tabel Kebenaran dari rangkaian yang akan dibuat ditunjukkan pada Tabel 10-1.

Tabel 10-1. Tabel Kebenaran urutan 3 bit data dan Output Even Parity Generator

INPUT			OUTPUT
A	B	C	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Dari tabel di atas, selanjutnya didapatkan persamaan sebagai berikut :

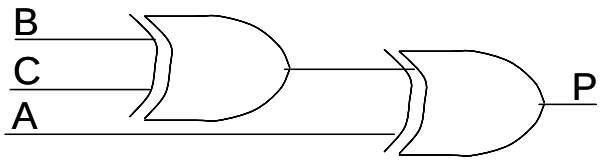
$$P = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$$

$$= \overline{A}(\overline{B}C + B\overline{C}) + A(\overline{B}\overline{C} + BC)$$

$$= \overline{A}(B \oplus C) + A(\overline{B \oplus C})$$

$$= A \oplus (B \oplus C)$$

Rangkaiannya seperti ditunjukkan pada gambar 10-2.



Gambar 10-2. Rangkaian *Even Parity Generator* 3 bit data

Untuk mendisain rangkaian *Parity Checker* perlu ditentukan lebih dulu jumlah data dalam setiap urutannya dan jenis bit parity yang akan dikirim. Selanjutnya output akan diberi nilai ‘0’ atau ‘1’ tergantung ada tidaknya kesalahan dalam satu urutan data. Tabel kebenaran *Parity Checker* ditunjukkan pada Tabel 10-2.

Tabel 10-2. Tabel Kebenaran *Even Parity Checker* 3 bit data

INPUT				OUTPUT
A	B	C	P	Ch
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Dari Tabel di atas, didapatkan persamaan sebagai berikut :

$$Ch = \overline{A}\overline{B}\overline{C}P + \overline{A}\overline{B}C\overline{P} + \overline{A}B\overline{C}\overline{P} + \overline{A}BCP + A\overline{B}\overline{C}\overline{P} + A\overline{B}CP + AB\overline{C}P + ABC\overline{P}$$

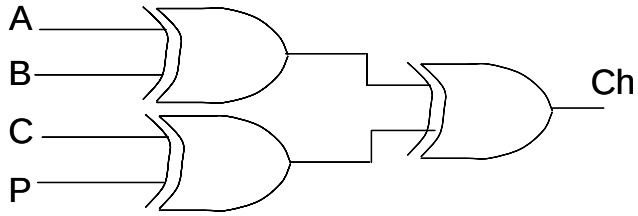
$$= \overline{A}\overline{B}(\overline{C}P + C\overline{P}) + \overline{A}B(\overline{C}\overline{P} + CP) + A\overline{B}(\overline{C}\overline{P} + CP) + AB(\overline{C}P + C\overline{P})$$

$$= (\overline{C}P + C\overline{P})(\overline{A}\overline{B} + AB) + (\overline{C}\overline{P} + CP)(\overline{A}B + A\overline{B})$$

$$= (C \oplus P)\overline{(A \oplus B)} + \overline{(C \oplus P)}(A \oplus B)$$

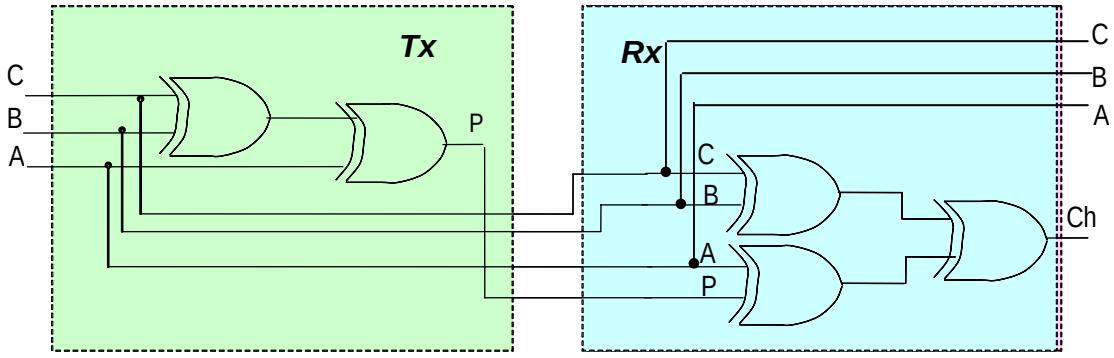
$$= (C \oplus P) \oplus (A \oplus B)$$

Rangkaiannya ditunjukkan pada gambar 10-3.



Gambar 10-3. Rangkaian Even Parity Checker 3 bit data

Rangkaian gabungan *Parity Generator* (di sisi kirim) dan *Parity Checker* (di sisi terima) untuk 3 bit data, ditunjukkan pada gambar 10-4



Gambar 10-4. Rangkaian Gabungan Parity Generator dan Checker 3 bit data

2. DATA SCRAMBLING

Data scrambling merupakan proses pengacakan data yang menggunakan aplikasi *Parity Generator* dan *Checker*. Prinsip kerja dari *data scrambling* ini sangat sederhana, yaitu meletakkan bit-bit parity di sela-sela urutan data informasi yang dikirim. Nilai bit parity adalah Ex-OR dari bit-bit data informasi pada posisi tertentu. Pada sisi terima, pengecekan dilakukan dengan meng-Ex-OR kan bit-bit parity dan bit data informasinya. Jika hasil pengecekan bernilai ‘0’ berarti urutan bit tersebut benar, jika ‘1’ berarti ada kesalahan di posisi tertentu.

Diberikan urutan data 3 bit ($D_2D_1D_0$), akan ditambahkan 2 bit parity di antara ketiga bit tersebut, yaitu X_1 dan X_0, sehingga urutan data yang dikirim menjadi $X_1D_2X_0D_1D_0$. Nilai X_1 adalah Ex-OR dari D_2 dan D_0, sedangkan nilai X_0 adalah Ex-OR dari D_1 dan D_0 (Aturan yang lebih detail dari nilai bit sisipan ini termuat pada pembahasan *Hamming Code* atau *Error Correction Code*). Persamaan dari X_1 dan X_0 dinyatakan sebagai berikut :

$$X_1 = D_2 \oplus D_0$$

$$X_0 = D_1 \oplus D_0$$

Tabel hasil *scrambling* diberikan pada Tabel 10-3.

Tabel 10-3. Tabel Hasil *Scrambling* 3 bit data dan 2 bit sisipan

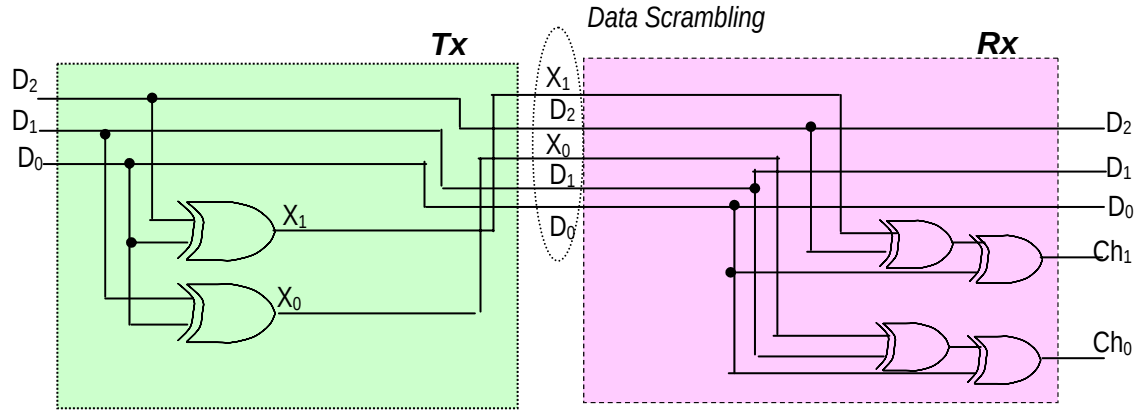
Data/bit	X_1	D_2	X_0	D_1	D_0
000	0	0	0	0	0
001	1	0	1	0	1
010	0	0	1	1	0
011	1	0	0	1	1
100	1	1	0	0	0
101	0	1	1	0	1
110	1	1	1	1	0
111	0	1	0	1	1

Pada sisi terima, nilai urutan data dan bit-bit sisipannya di Ex-OR kan untuk menge-cek apakah urutan data tersebut benar atau salah. Persamaan untuk mendapatkan nilai hasil penge-cek an adalah sebagai berikut :

$$Ch_1 = X_1 \oplus D_2 \oplus D_0$$

$$Ch_0 = X_0 \oplus D_1 \oplus D_0$$

Sehingga rangkaian lengkap Scrambler (di sisi kirim) dan Descrambler (di sisi terima) adalah seperti pada gambar 10-5.



Gambar 10-5. Rangkaian Lengkap *Scrambler* dan *Descrambler* 3 bit data

PROSEDUR:

1. Dengan menggunakan trainer ITF-02 atau DL-02, buat rangkaian *Odd Parity Generator* 2 bit data. Dapatkan Tabel Kebenarannya.
2. Masih dengan 2 bit data yang sama, tambahkan 1 input sebagai bit parity. Buat rangkaian *Odd Parity Checker*. Dapatkan Tabel Kebenarannya.

3. Sambungkan dua bagian tadi (bagian *Odd Parity Generator* dan *Odd Parity Checker*). Berikan output Parity bit dari *Parity Generator* sebagai bit input parity dari bagian *Parity Checker*. Perhatikan, apa yang terjadi pada output *Parity Checker* ? Buat Tabel Kebenarannya.
4. Dengan menggunakan trainer ITF-02 atau DL-02, buat rangkaian *scrambler* dan *descrambler* seperti gambar 10-5. Karena jumlah gerbang Ex-OR pada masing-masing trainer terbatas, lakukan untuk nilai X_1 dulu, selanjutnya baru nilai X_0. Buat Tabel Kebenarannya.

TUGAS:

1. Disain rangkaian *Odd Parity Generator* dan *Checker* untuk urutan 4 bit data. Lengkapi dengan Tabel Kebenaran dan persamaan untuk mendapatkan rangkaiannya.
2. Implementasikan metode *Hamming Code* (untuk urutan 4 bit data) dengan rangkaian *Scrambler* dan *Descrambler*. Lengkapi dengan Tabel Kebenaran dan persamaan untuk mendapatkan masing-masing bit sisipan dan kode pengecek kesalahan.