SISTEM DIGITAL (TIF-1106)

Pertemuan 2: Kode Digital (BCD 8421, Gray, ASCII), Deteksi Error Paritas, & Aritmatika BCD

Tim Dosen Teknik Informatika — S1 Teknik Informatika

Capaian Pembelajaran Pertemuan 2 (Sub-CPMK 1)

Sub-CPMK 1 (Lanjutan)

Mahasiswa mampu mengonversi data desimal dan karakter ke dalam kode **BCD 8421**, **Kode Gray**, dan **ASCII**, menganalisis operasi aritmatika BCD, serta merancang skema *Parity Bit* untuk deteksi kesalahan transmisi data.

Domain Pengetahuan Teoretis

- Konsep *Weighted* vs *Unweighted Codes*
- Keadaan Terlarang (*Invalid States*) BCD ($1010_2 - 1111_2$)
- Properti *Unit Distance* pada Kode Gray
- Standar Tabel Karakter ASCII 7-Bit / 8-Bit

Domain Keterampilan Praktis

- Algoritma Koreksi $+6_{10}$ (0110_2) Penjumlahan BCD
- Konversi Dua Arah Biner $\leftrightarrow$ Gray via Gerbang XOR
- Pembangkitan Even/Odd Parity Bit Generator
- Simulasi Dekoder BCD to 7-Segment di Logisim

Tugas Membaca Wajib Minggu 2 (*Weekly Reading*)

Tugas Literasi Sebelum Sesi Perkuliahan & Praktikum

Mahasiswa wajib membaca bab referensi berikut dari berkas PDF terbuka (`buku_referensi_pdf/`) sebelum mengikuti kelas teori dan praktikum:

- **[Ref 1] Kuphaldt (2024)**: *Lessons In Electric Circuits Vol. IV – Digital*
  - **Bab 3**: *Binary Codes & Data Representation* (BCD 8421, Gray Code, ASCII, Parity Bits)
- **[Ref 2] Coulston (2020)**: *Digital Design: A Datapath and Control*
  - **Bab 1**: *Data Encodings, Code Converters, & Error Detection*
- **[Ref 3] Coulston Labs (2020)**: *Digital Design Laboratory Manual*
  - **Lab 2**: *Code Converters, 7-Segment Displays, & Parity Checkers*

Pre-Test Evaluasi

Setiap sesi perkuliahan diawali kuis 5 menit berbasis materi bacaan wajib di atas.

Mengapa Kita Membutuhkan Kode Digital?

Jembatan Manusia & Mesin

Komputer bekerja secara optimal dengan **Biner Murni**, namun manusia berinteraksi dengan:

- Angka desimal ($0 - 9$) pada kalkulator/display.
- Karakter alfabet ($A - Z, a - z$) dan simbol teks.
- Posisi sudut mekanis pada sensor poros.

Klasifikasi Kode Digital

1. **Weighted Code**: Tiap posisi bit ada bobot fisis (Cth: BCD 8421).
2. **Unweighted Code**: Tanpa bobot (Cth: Gray Code).
3. **Alphanumeric Code**: Teks/angka (Cth: ASCII).
4. **Error-Detecting Code**: Deteksi error (Cth: Bit Paritas).

Kode BCD (Binary Coded Decimal 8421)

Definisi BCD 8421

Setiap digit desimal (0 s.d. 9) dikodekan secara individual menjadi **kelompok 4-bit biner** dengan pembobotan $2^3 = 8, 2^2 = 4, 2^1 = 2, 2^0 = 1$.

Desimal	0	1	2	3	4	5	6	7	8	9
BCD 8421	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

6 Kombinasi Terlarang (*Invalid / Illegal BCD Codes*)

Dari $2^4 = 16$ kombinasi 4-bit, hanya 10 kombinasi yang valid. Sebanyak **6 kombinasi** berikut adalah **TIDAK VALID**:

$$1010_2(10_{10}), \quad 1011_2(11_{10}), \quad 1100_2(12_{10}), \quad 1101_2(13_{10}), \quad 1110_2(14_{10}), \quad 1111_2(15_{10})$$

Perbandingan BCD 8421 vs Biner Murni

Studi Kasus Konversi: Nilai Desimal $(853)_{10}$

- **Konversi BCD 8421**:

$$8 \rightarrow \mathbf{1000}, \quad 5 \rightarrow \mathbf{0101}, \quad 3 \rightarrow \mathbf{0011} \implies (\mathbf{1000 \quad 0101 \quad 0011})_{\mathbf{BCD}} \quad \text{(Butuh 12 Bit)}$$

- **Konversi Biner Murni**:

$$(853)_{10} = (\mathbf{1101010101})_{\mathbf{2}} \quad \text{(Hanya Butuh 10 Bit)}$$

Parameter Evaluasi	Biner Murni	Kode BCD 8421
Efisiensi Penggunaan Memori	Sangat Tinggi	Lebih Rendah ($\approx$ 20% bit terbuang)
Kemudahan Antarmuka Display (7-Seg)	Memerlukan sirkuit konverter rumit	Sangat Mudah (Langsung per-digit)
Kecepatan Pemrosesan Aritmatika CPU	Sangat Cepat (Sirkuit Adder Sederhana)	Lebih Lambat (Butuh Sirkuit Koreksi +6)

Penjumlahan BCD & Algoritma Koreksi $+6_{10}$ (0110_2)

Saat dua digit BCD dijumlahkan menggunakan *Binary Full Adder*, hasil penjumlahan 4-bit dapat menghasilkan kesalahan karena selisih basis radiks ($16 - 10 = 6$).

Aturan Wajib Koreksi $+6$ (0110_2)

Koreksi dengan menambahkan **0110_2 ($+6_{10}$)** wajib dilakukan jika dan hanya jika:

1. Hasil penjumlahan 4-bit $> 9_{10}$ (yaitu bit menghasilkan pola 1010_2 s.d. 1111_2), **ATAU**
2. Menghasilkan *Carry-out* ($K = 1$) dari kelompok 4-bit tersebut.

Persamaan Logika Deteksi Koreksi (C_{korek})

Misalkan luaran adder 4-bit adalah Z_3, Z_2, Z_1, Z_0 dan carry out K:

$$C_{korek} = K + Z_3Z_2 + Z_3Z_1$$

Jika $C_{korek} = 1$, maka sirkuit otomatis mengaktifkan penambahan 0110_2.

Contoh Soal 1: Penjumlahan Multi-Digit BCD

Hitung Penjumlahan BCD: $(48)_{10} + (37)_{10} = (85)_{10}$

Digit Puluhan ($4 + 3$):

0100	(4_{BCD})
+0011	(3_{BCD})
0111	($= 7_{10}$)
+0001	(Carry dari Satuan)
1000	($= \mathbf{8_{BCD}} \leq 9 \rightarrow$ Valid!)

Digit Satuan ($8 + 7$):

1000	(8_{BCD})
+0111	(7_{BCD})
1111	($= 15_{10} > 9 \rightarrow$ ILLEGAL!)
+0110	(KOREKSI $+ 6$)
1 0101	(Carry $= 1$, Hasil $= \mathbf{5_{BCD}}$)

Hasil Akhir Penggabungan Digit: $(\mathbf{1000\ \ 0101})_{\mathbf{BCD}} = \mathbf{85_{10}}$ ✓

Pengurangan BCD menggunakan Komplemen 9 (*9's Complement*)

Bagaimana ALU melakukan pengurangan $(A - B)$ dalam BCD padahal sirkuit utamanya adalah adder? ALU memanfaatkan sistem **Komplemen 9**!

Definisi & Prosedur Komplemen 9

Komplemen 9 dari suatu digit desimal D adalah $(9 - D)$. Untuk menghitung $A - B$ dalam BCD:

1. Ubah pengurang (B) menjadi Komplemen 9-nya.
2. Jumlahkan $A +$ (Komplemen 9 dari B) menggunakan aturan penjumlahan BCD (koreksi $+6$ jika perlu).
3. Jika ada *End-Around Carry* (Carry Out dari MSB), tambahkan kembali 1 ke hasil akhir.

Contoh: $(8)_{10} - (3)_{10} = (5)_{10}$

Komplemen 9 dari 3 adalah $(9 - 3) = \mathbf{6}$ (0110_{BCD}).
Jumlahkan $8 + 6 \implies 1000 + 0110 = \mathbf{1110_2}$ $(14_{10}$, Illegal $\implies$ Koreksi $+6/0110_2)$.
$1110 + 0110 = \mathbf{1}\quad\mathbf{0100_2}$. Carry 1 ditambahkan (*End-Around*): $0100 + 0001 = \mathbf{0101}(5_{BCD})$. ✓

Kode Gray (*Reflected Binary / Unit Distance Code*)

Sifat Unit Distance

Antara dua nilai desimal yang berurutan, **hanya ada TEPAT 1 BIT yang berubah keadaan ($0 \leftrightarrow 1$)**.

Manfaat Fisis pada Shaft Encoder

Pada sensor sudut putar optik (*optical shaft encoder*), biner biasa dapat mengalami kesalahan pembacaan besar akibat ketidaksempurnaan mekanis (*switching glitch*).

- Transisi $3 \rightarrow 4$ Biner: $011 \rightarrow 100$ (3 bit berubah serentak $\rightarrow$ rawan error).
- Transisi $3 \rightarrow 4$ Gray: $010 \rightarrow 110$ (Hanya 1 bit berubah $\rightarrow$ bebas glitch!).

Desimal	Biner ($B_3B_2B_1B_0$)	Gray ($G_3G_2G_1G_0$)
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100

Visualisasi Masalah *Glitch* pada Sensor Linier/Rotari

Mengapa Biner rawan error? Jika sensor berada di perbatasan dua nilai, sensor mungkin membaca nilai acak.

Transisi Biner $3 \rightarrow 4$ **(**$011 \rightarrow 100$**):** Jika bit MSB telat membaca, sensor bisa sesaat membaca 000 (nilai 0).

Transisi Gray $3 \rightarrow 4$ **(**$010 \rightarrow 110$**):** Hanya 1 bit berubah. Posisi jarum yang "menggantung" hanya akan membaca nilai 3 atau 4, tanpa *glitch*.

Pola Linier Biner vs Gray (3-Bit)

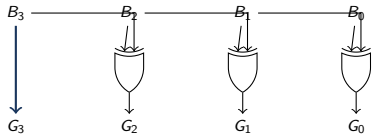
Algoritma Konversi Biner $\leftrightarrow$ Gray

Konversi Biner $\rightarrow$ Gray

1. Bit MSB Gray sama dengan MSB Biner:
$$G_{n-1} = B_{n-1}$$
2. Bit berikutnya adalah hasil XOR bit biner berurutan:
$$G_i = B_{i+1} \oplus B_i$$

Konversi Gray $\rightarrow$ Biner

1. Bit MSB Biner sama dengan MSB Gray:
$$B_{n-1} = G_{n-1}$$
2. Bit berikutnya adalah hasil XOR bit biner sebelumnya dengan bit Gray saat ini:
$$B_i = B_{i+1} \oplus G_i$$



Contoh Soal 2: Konversi Dua Arah Biner $\leftrightarrow$ Gray

Kasus A: Konversikan Biner $(10110110)_2 \rightarrow$ Kode Gray

- $G_7 = B_7 = \mathbf{1}$
- $G_6 = B_7 \oplus B_6 = 1 \oplus 0 = \mathbf{1}$
- $G_5 = B_6 \oplus B_5 = 0 \oplus 1 = \mathbf{1}$
- $G_4 = B_5 \oplus B_4 = 1 \oplus 1 = \mathbf{0}$
- $G_3 = B_4 \oplus B_3 = 1 \oplus 0 = \mathbf{1}$
- $G_2 = B_3 \oplus B_2 = 0 \oplus 1 = \mathbf{1}$
- $G_1 = B_2 \oplus B_1 = 1 \oplus 1 = \mathbf{0}$
- $G_0 = B_1 \oplus B_0 = 1 \oplus 0 = \mathbf{1}$

Hasil Gray: $(\mathbf{11101101})_{\text{Gray}}$ ✓

Kasus B: Konversikan Gray $(1101101)_{Gray} \rightarrow$ Biner Murni

$B_6 = G_6 = \mathbf{1}$; $B_5 = 1 \oplus 1 = \mathbf{0}$; $B_4 = 0 \oplus 0 = \mathbf{0}$; $B_3 = 0 \oplus 1 = \mathbf{1}$; $B_2 = 1 \oplus 1 = \mathbf{0}$; $B_1 = 0 \oplus 0 = \mathbf{0}$; $B_0 = 0 \oplus 1 = \mathbf{1}$
Hasil Biner: $(\mathbf{1001001})_2$ ✓

Kode Alfanumerik ASCII

Standar ASCII (American Standard Code for Information Interchange)

Mengodekan huruf alfabet, angka desimal, tanda baca, serta karakter kontrol komunikasi menggunakan **format 7-bit** ($2^7 = 128$ karakter unik).

Karakter	Hex 7-Bit	Biner 7-Bit ($b_6 \dots b_0$)	Keterangan
'A' s.d. **'Z'**	$41_{16} - 5A_{16}$	$1000001_2 - 1011010_2$	Huruf Kapital Latin
'a' s.d. **'z'**	$61_{16} - 7A_{16}$	$1100001_2 - 1111010_2$	Huruf Kecil (Bit $b_5 = 1$)
'0' s.d. **'9'**	$30_{16} - 39_{16}$	$0110000_2 - 0111001_2$	Angka Desimal (Bit $b_3..b_0$ = BCD)
NUL, CR, LF	$00_{16}, 0D_{16}, 0A_{16}$	$0000000_2, \dots$	Karakter Kontrol Transmisi

Trik Konversi Angka Desimal $\leftrightarrow$ ASCII

Untuk mengubah digit angka desimal ke ASCII, cukup tambahkan 011_2 di depan 4-bit BCD:

$$\text{Angka } 7_{10} = \underbrace{0110111_2}_{\text{ASCII}} = 37_{16}$$

Aplikasi Nyata: Transmisi Serial Asinkron (UART/RS-232)

Di dunia nyata, karakter ASCII dikirim secara serial.

Format Serial Frame

- **Start Bit (0)**: Menandakan awal transmisi.
- **Data Bits**: 7-bit ASCII (Lsb dikirim pertama).
- **Parity Bit**: Bit proteksi (Even/Odd).
- **Stop Bit (1)**: Menandakan akhir transmisi.

Timing Diagram Transmisi ASCII "A" (1000001_2) dengan Even Parity

Deteksi Kesalahan Transmisi dengan Bit Paritas

Bit paritas (*Parity Bit*) ditambahkan sebagai bit ke-8 (P) pada karakter ASCII 7-bit untuk mendeteksi *Single-Bit Error* selama transmisi:

Even Parity (Genap)

Nilai bit P dipilih sedemikian rupa sehingga **total jumlah bit 1 dalam frame data bernilai GENAP**:

$$P_{\text{even}} = b_6 \oplus \cdots \oplus b_0$$

Odd Parity (Ganjil)

Nilai bit P dipilih sedemikian rupa sehingga **total jumlah bit 1 dalam frame data bernilai GANJIL**:

$$P_{\text{odd}} = \overline{P_{\text{even}}} = \overline{b_6 \oplus \cdots \oplus b_0}$$

Keterbatasan Paritas & Solusi Industri Modern

Hanya mendeteksi jumlah error ganjil ($1, 3, 5$ bit). Jika terjadi **2 bit error serentak**, gagal terdeteksi! **Solusi Modern:** Sistem masa kini menggunakan **Hamming Code (ECC RAM)** untuk mengoreksi error, atau **CRC** pada jaringan/penyimpanan data.

Contoh Soal 3: Pembangkitan Paritas String Karakter

Tentukan Format Transmisi ASCII 8-Bit (Even Parity) untuk Kata "'DATA"'

Karakter	ASCII 7-Bit	Jumlah Bit 1	Even Parity (P)	Byte Transmisi (P + ASCII)
'D'	1000100_2	2 (Sudah Genap)	**0**	$\mathbf{0}1000100_2 = 44_{16}$
'A'	1000001_2	2 (Sudah Genap)	**0**	$\mathbf{0}1000001_2 = 41_{16}$
'T'	1010100_2	3 (Ganjil)	**1**	$\mathbf{1}1010100_2 = D4_{16}$
'A'	1000001_2	2 (Sudah Genap)	**0**	$\mathbf{0}1000001_2 = 41_{16}$

Pemeriksaan di Sisi Penerima (Receiver)

Jika penerima menerima byte $\mathbf{1}1000100_2$ saat membaca karakter "'D"', penerima menghitung total bit 1 = 3 (Ganjil). Karena sistem menggunakan *Even Parity*, penerima langsung membangkitkan **Parity Error Flag** dan meminta transmisi ulang!

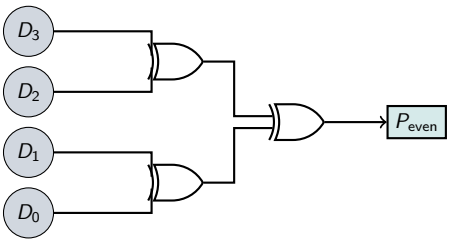
Bridging ke Praktikum: Even Parity Generator di Logisim Evolution

Bagaimana kita membuat *Even Parity Generator* untuk 4-bit data (D_3, D_2, D_1, D_0) di perangkat lunak simulasi?

Solusi: Gunakan kaskade gerbang **XOR** (Gerbang logika Eksklusif-OR). Output XOR akan bernilai 1 jika jumlah input bernilai ganjil.

Rangkaian ini akan kita rancang secara komprehensif pada **Sesi Praktikum Lab 2**.

Skematik Logisim (Even Parity 4-Bit)



Latihan Interaktif di Kelas: Uji Pemahaman Mandiri

Soal Latihan

1. Konversikan bilangan desimal $(96)_{10}$ ke dalam bentuk BCD 8421! 2. Jika suatu transmisi data menggunakan *Odd Parity*, tentukan bit paritas P untuk karakter ASCII ''C'' (1000011_2)! 3. Mengapa kode Gray sangat aman digunakan pada sensor posisi mekanis dibanding biner murni?

Jawaban & Pembahasan Singkat

1. $(96)_{10} \rightarrow 9 = 1001_2, 6 = 0110_2 \implies (\mathbf{1001}\quad\mathbf{0110})_{\text{BCD}}$ ✓
2. Karakter ''C'' (1000011_2) memiliki total 3 buah bit 1 (sudah ganjil). Maka bit *Odd Parity* $P = \mathbf{0} \implies \mathbf{0}1000011_2$. ✓
3. Karena kode Gray bersifat *Unit Distance* (hanya 1 bit berubah pada setiap pergantian posisi sudut), sehingga mencegah *transient glitch* saat transisi sensor. ✓

Rangkuman Pertemuan 2

- **BCD 8421** mengodekan setiap digit desimal menjadi 4-bit biner dan memerlukan **koreksi $+6$ (0110_2)** jika hasil penjumlahan > 9 atau menghasilkan carry out.
- **Kode Gray** adalah kode tak berbobot (*unweighted*) dengan sifat *Unit Distance* yang mencegah kesalahan pembacaan sensor putar optik (*Shaft Encoder*).
- Konversi Biner $\leftrightarrow$ Gray direalisasikan menggunakan jaringan gerbang logika **XOR**.
- **ASCII 7-Bit** adalah standar pengodean alfanumerik universal.
- **Bit Paritas (Even/Odd)** menambahkan 1 bit proteksi pada transmisi data untuk mendeteksi *Single-Bit Error*.