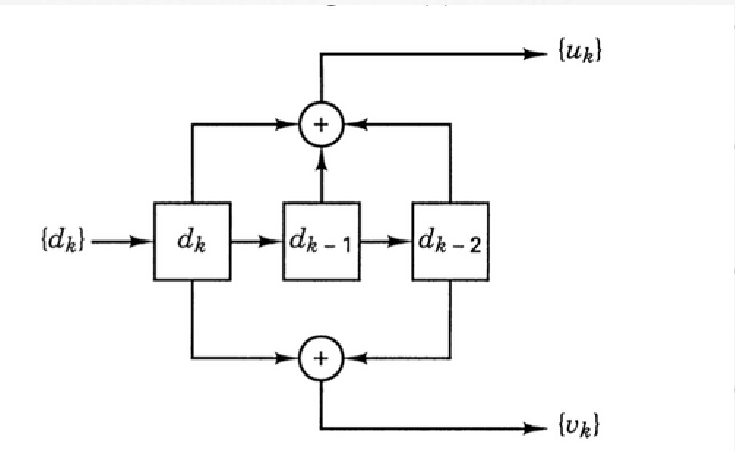
# TURBO ENCODER

## Concatenation of RSC codes

### Non Systematic code

$$u_k = \sum_{i=0}^{K-1} g_{1i}\, d_{k-i} \mod 2 \quad g_{1i} = 0,1$$

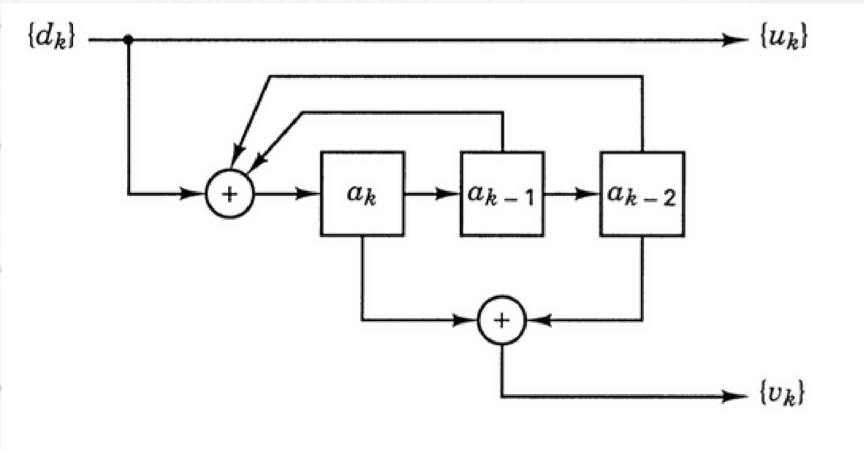
$$v_k = \sum_{i=0}^{K-1} g_{2i}\, d_{k-i} \mod 2 \quad g_{2i} = 0,1$$

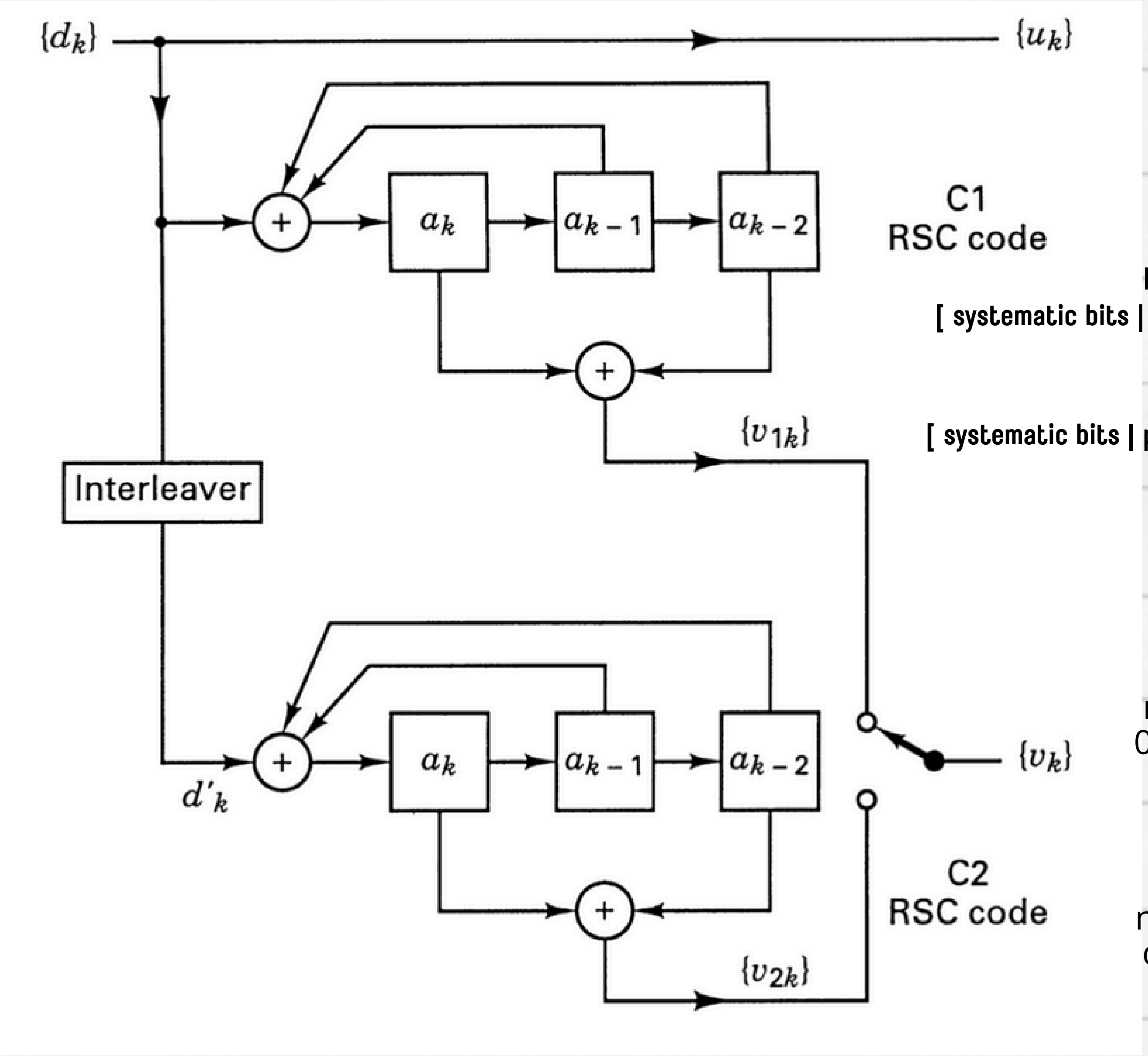



### Recursive Systematic code

$$a_k = d_k + \sum_{i=1}^{K-1} g'_i\, a_{k-i} \mod 2$$

qual to $g_{1i}$ if $u_k = d_k$, and to $g_{2i}$ if $v_k = d_k$.




We now parallely run two encoders as shown in figuer there is a switc yeilding v_k which provides puncturin thus overall code rate is still ½ instead c 1/3

**Final output if overall rate was 1/3**
**[ systematic bits | parity bits from encoder 1 | parity bits from encoder 2 ]**

**Final output if overall rate is 1/2**
**[ systematic bits | parity bits from encoder 1 or parity bits from encoder 2**

Why use interleaver- interleaver shuffles the output receiver from receiver(input for encoder).

If the component encoders are not recursive, the unit weight input sequence 0 0 … 0 0 1 0 0 … 0 0 will always generate a low-weight codeword at the input of a second encoder for any interleaver design. In other words, the interleaver would not influence the output-codeword weight distribution if the component codes were not recursive. However if the component codes are recursive, a weight-1 input sequence generates an infinite impulse response (infinite-weight output).

Constraint length k
memory k-1
shift register-state- like a conveyer belt
d_k- input to encoder at time t
g's- code generator
u_k, v_k- outputs
a_k- similar to d_k but are recursively calculated

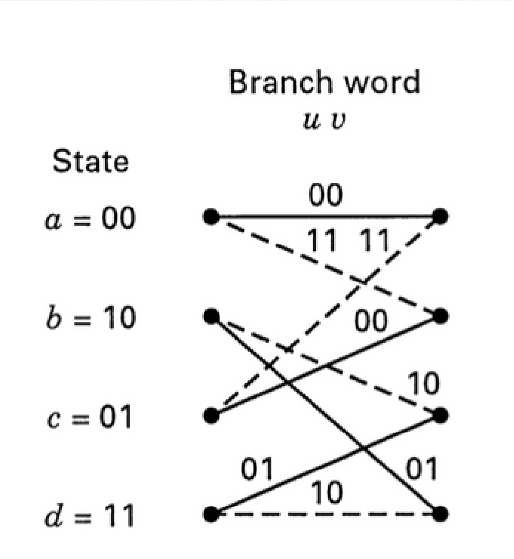
$\mathbf{G_1} = \{ g_{1i} \}$ and $\mathbf{G_2} = \{ g_{2i} \}$

**How G₁ and G₂ work:**

Each generator tells you which bits to XOR:

- For $G_1 = \{1\,1\,1\}$:
  `uₖ = dₖ XOR dₖ₋₁ XOR dₖ₋₂`
- For $G_2 = \{1\,0\,1\}$:
  `vₖ = dₖ XOR dₖ₋₂`

Notice $G_2$ skips $d_{k-1}$ because the middle bit is `0`.




Example- suppose k=3, and simple binary rate is 1/2

$\mathbf{G_1} = \{ 1\ 1\ 1 \}$ and $\mathbf{G_2} = \{ 1\ 0\ 1 \}$

| Time | $d_k$ | Past bits ($d_{k-1}, d_{k-2}$) | $u_k = d_k \odot d_{k-1} \odot d_{k-2}$ | $v_k = d_k \odot d_{k-2}$ |
|---|---|---|---|---|
| 0 | 1 | (0, 0) | $1\odot0\odot0$ = **1** | $1\odot0$ = **1** |
| 1 | 0 | (1, 0) | $0\odot1\odot0$ = **1** | $0\odot0$ = **0** |
| 2 | 1 | (0, 1) | $1\odot0\odot1$ = **0** | $1\odot1$ = **0** |
| 3 | 1 | (1, 0) | $1\odot1\odot0$ = **0** | $1\odot0$ = **1** |

**$[d_k]$ → Convolutional Encoder → $[u_k, v_k]$ → **Channel** → $[r_{1k}, r_{2k}]$ → Decoder → [estimated $d_k$]**

| Time ($k$) | Input $d_k = u_k$ | First Stage $a_k$ | State at Time $k$ $a_{k-1}$ | $a_{k-2}$ | Output $u_k v_k$ |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 1 1 |
| 2 | 1 | 0 | 1 | 0 | 1 0 |
| 3 | 1 | 0 | 0 | 1 | 1 1 |
| 4 | 0 | 0 | 0 | 0 | 0 0 |
| 5 | | | 0 | 0 | |

- Weight = number of 1s.
- Low-weight inputs are dangerous because they can create low-weight codewords.
- **Recursive encoders** fix weight-1 inputs by spreading them into high-weight outputs.
- **Interleavers** help by making sure bad patterns (weight-2 or weight-3 inputs) don't show up aligned in both encoders at the same time.
- **Turbo codes** work so well mainly because of recursion + interleaving.