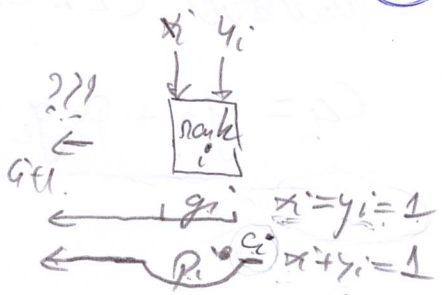
2.3. Parallel Data Computation

2.3.1. Full Carry Lookahead Adder

- outputs in parallel, each carry input



$$c_{i+1} = x_i \cdot y_i + x_i \cdot c_i + y_i \cdot c_i = x_i \cdot y_i + c_i (x_i + y_i)$$

$g_i = x_i \cdot y_i$: generate.

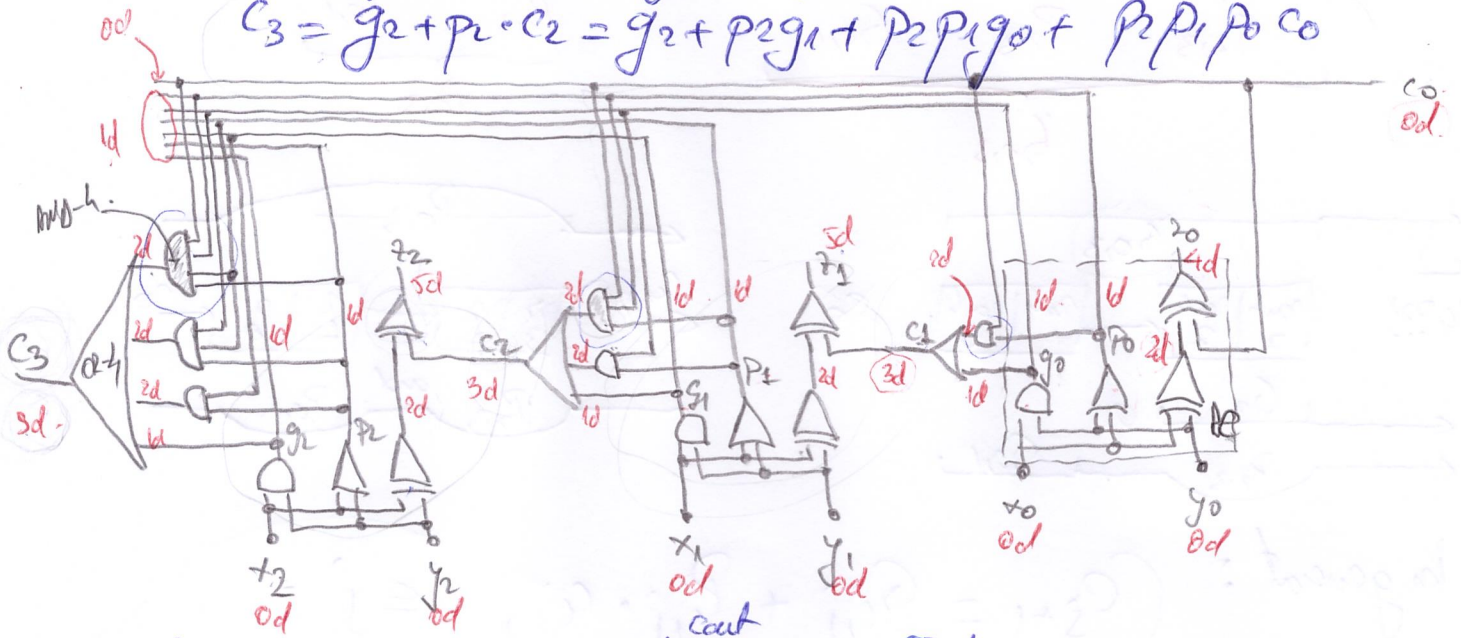
$p_i = x_i + y_i$: propagate.

$$c_{i+1} = g_i + p_i \cdot c_i$$

$$c_1 = g_0 + p_0 \cdot c_0$$

$$c_2 = g_1 + p_1 \cdot c_1 = g_1 + p_1 g_0 + p_1 p_0 c_0$$

$$c_3 = g_2 + p_2 \cdot c_2 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0$$



$$D^{2}_{F\text{-}CLA\text{-}3} = 5d \qquad D^{cout}_{F\text{-}CLA\text{-}3} = 3d$$

in general: $D^{?}_{F\text{-}CLA\text{-}n} = 5d \qquad D^{cout}_{F\text{-}CLA\text{-}n} = 3d \Rightarrow$ Constant latency.

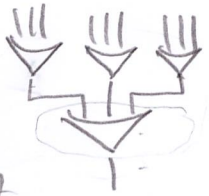
Theoretical design: F-CLA

- only implemented for small values of n. (2, 3, 4, at most 8)

Disadvantages:

- fan-out: c_0? for 3bit → 4 gates.
- fan-in: for 3bit → 4-input AND/OR

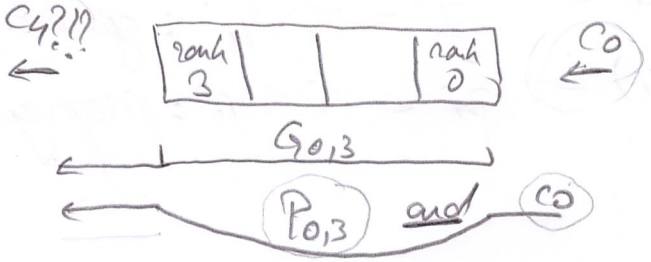
9-input OR gates!!!



tree of OR gates

Multilevel CLA: propagate and generate variables on blocks of cells

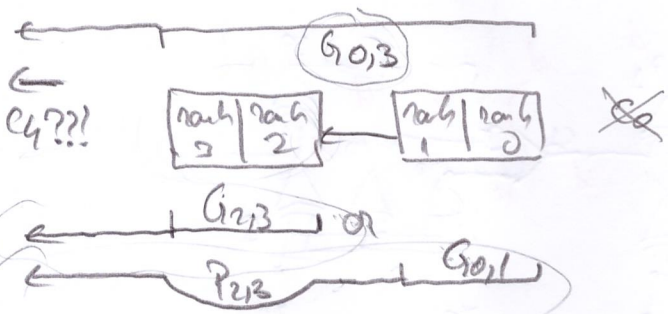
$$C_4 = \underbrace{g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0}_{G_{0,3}} + \underbrace{p_3 p_2 p_1 p_0}_{P_{0,3}} \cdot c_0$$

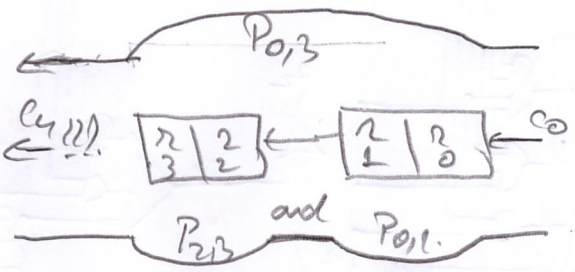


$$C_4 = G_{0,3} + P_{0,3} \cdot C_0$$

$$C_4 = \underbrace{g_3 + p_3 g_2}_{G_{2,3}} + \underbrace{p_3 p_2}_{P_{2,3}} \underbrace{(g_1 + p_1 g_0)}_{G_{0,1}} + \underbrace{p_3 p_2}_{P_{2,3}} \cdot \underbrace{p_1 p_0}_{P_{0,1}} \cdot c_0$$

$G_{0,3}$ $\quad$ $P_{0,3}$





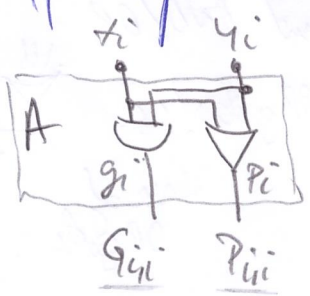
in general:

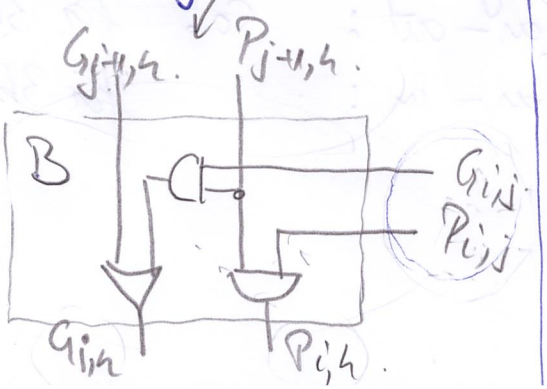
$$\begin{cases} C_{j+1} = G_{i,j} + P_{i,j} \cdot c_i, & i \le j \\ G_{i,k} = G_{j+1,k} + P_{j+1,k} \cdot G_{i,j}, & i \le j < k \\ P_{i,k} = P_{j+1,k} \cdot P_{i,j}, & i \le j < k \end{cases}$$

Notation:

$$G_{i,i} = g_i = x_i \cdot y_i$$
$$P_{i,i} = p_i = x_i + y_i$$

2 cells for implementing.

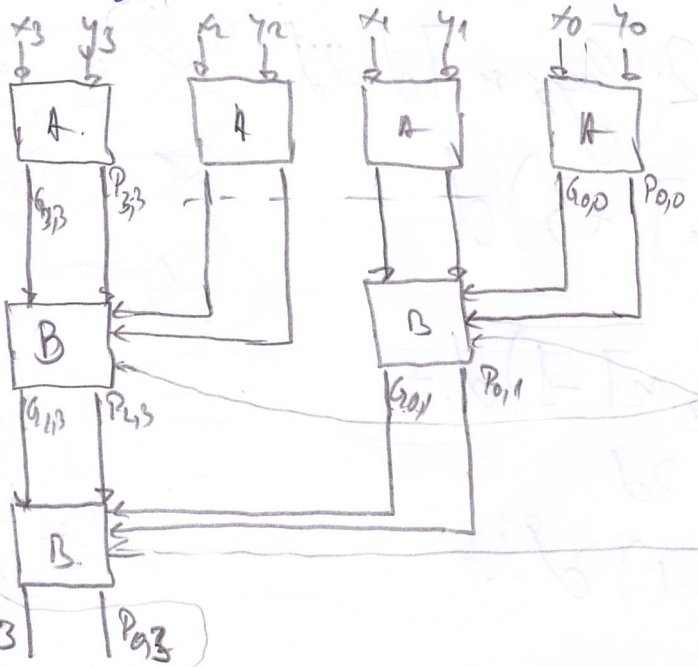


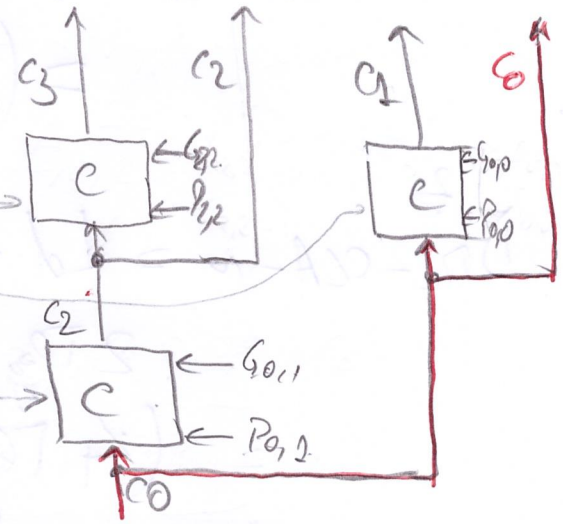


1 cell for C_{j+1}

Consider 2 4-bit operands X, Y.
- obtain $G_{0,3}$; $P_{0,3}$
- A and B cells.

compute c_1, c_2, c_3, c_0
↓ z_1 ↓ z_2 ↓ z_3 ↓ z_0

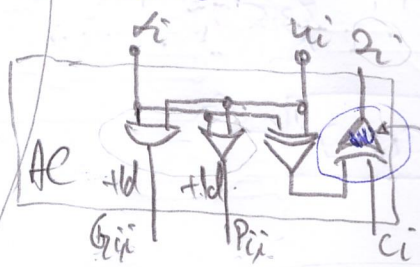


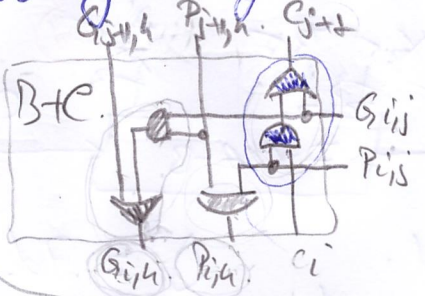


$c_4 = G_{0,3} + P_{0,3} \cdot c_0$

C cells overlapp the B cells ⇒ B+C cell

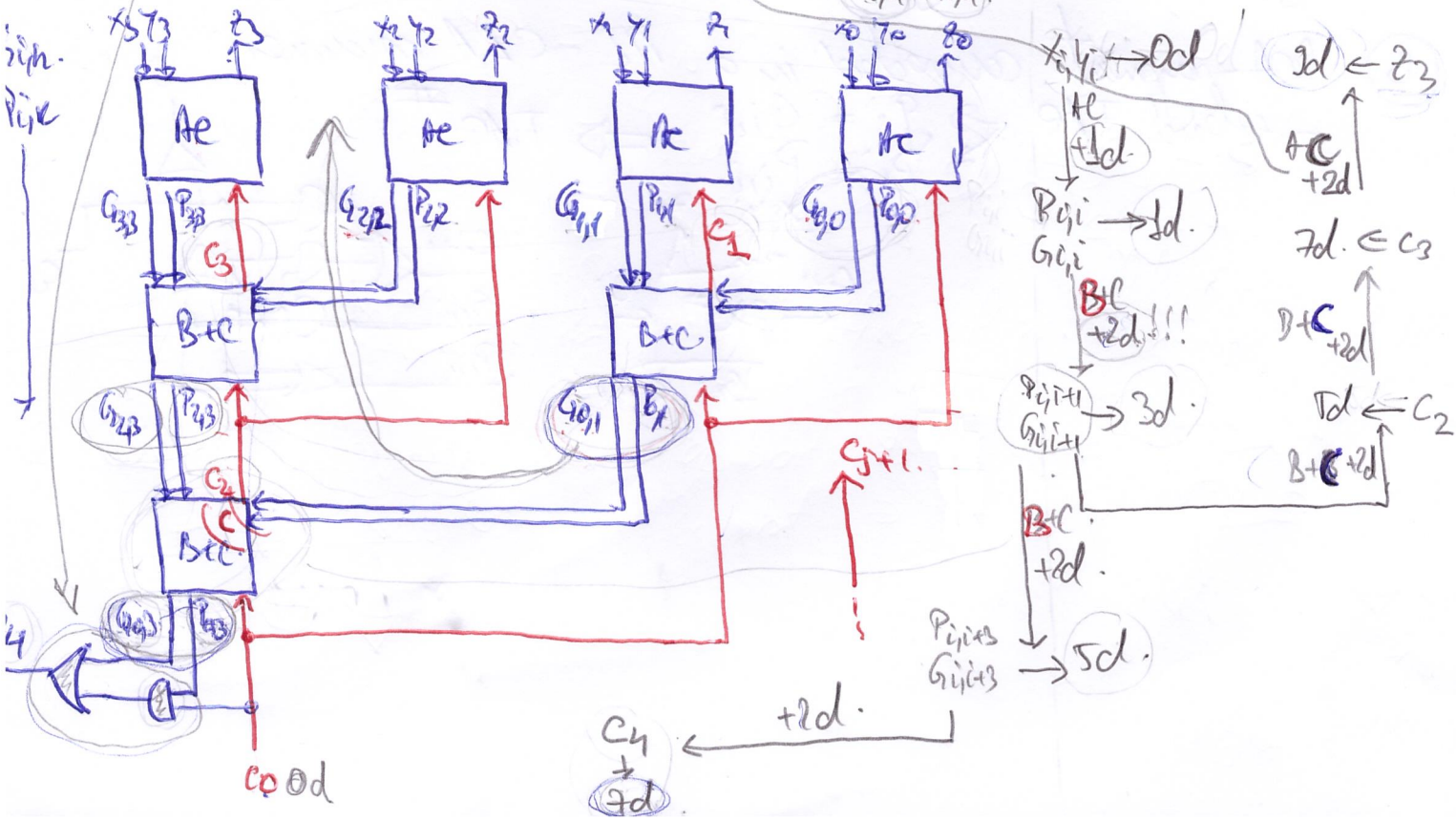
A cells become AC cells by adding z_i generation logic





In the TEST Board ! ! on 8 bits

Latencies



$D^{z}_{ML-CLA-4} = 9d$ $\quad \Delta^{cout}_{ML-CLA-4} = 7d$

$\lceil 2.01 \rceil = 3$

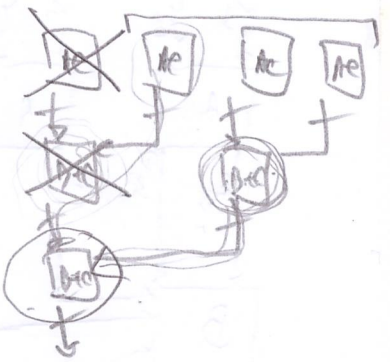
$\lceil 7 \rceil = 7$

$\lceil a \rceil -$

In general: G_{ii}, P_{ii}

$$\Delta^{cout}_{ML-CLA-n} = 1d + 2\lceil \log_2 n \rceil d + 2d$$

$$= (2\lceil \log_2 n \rceil + 3)d$$



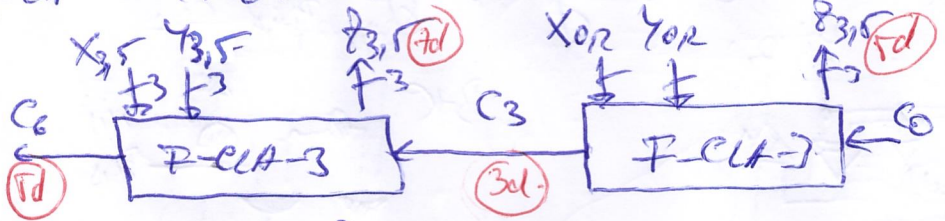
$$D^{z}_{ML-CLA-n} = 1d + 2(\lceil \log_2 n \rceil - 1)d + 2\lceil \log_2 n \rceil d + 2d$$

$$= (4\lceil \log_2 n \rceil + 1)d.$$

Hybrid CLA.

Consider operands on 6 bits.

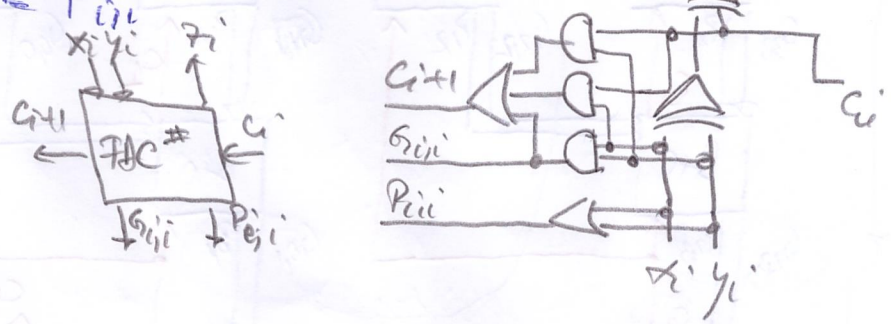
(A) F-CLA connected in RCA structure

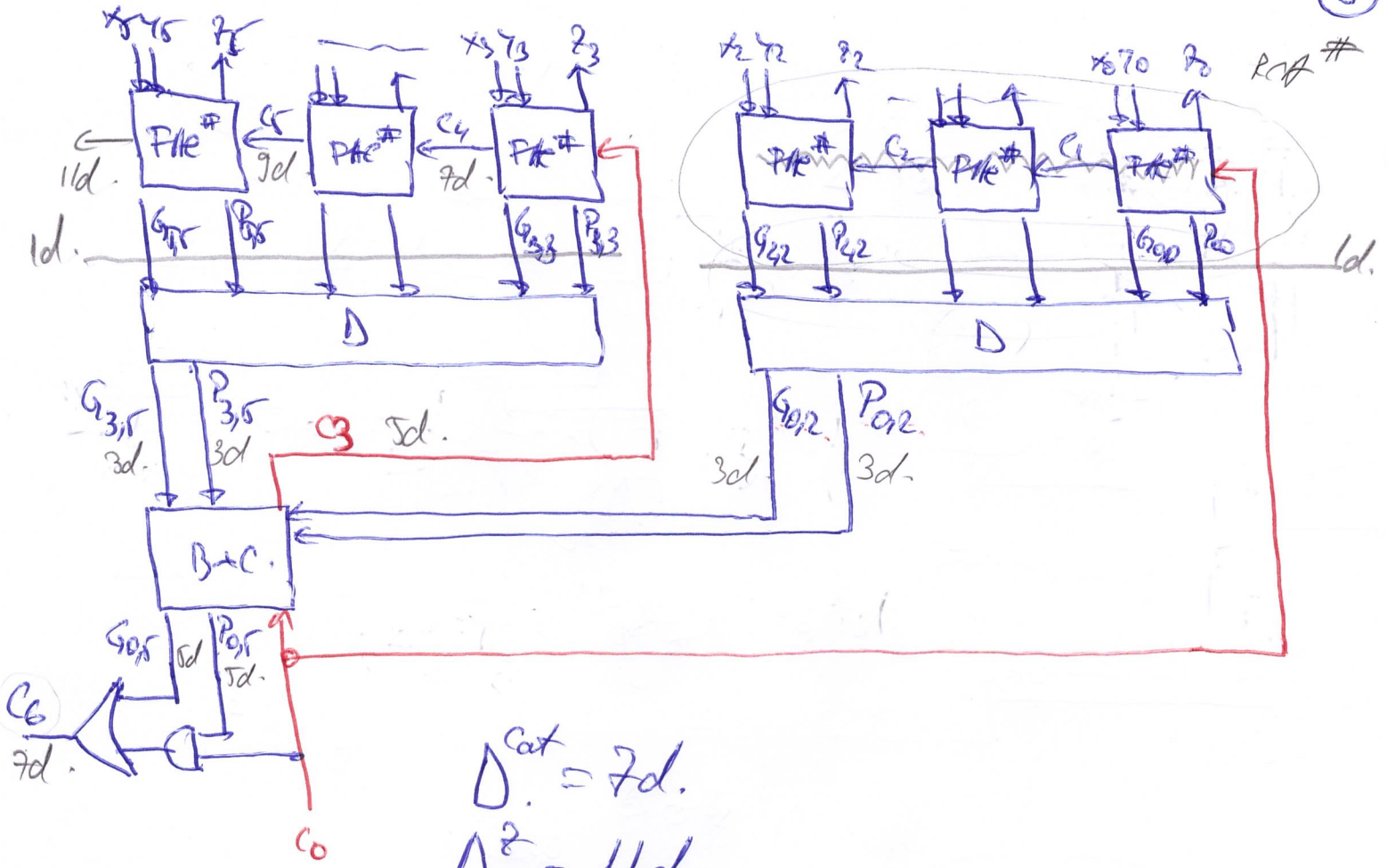


$\Delta^{cout} = 7d.$ $\quad \Delta^{s} = 7d.$

(B) RCA-segments connected in a ML-CLA manner

- extend FAC $\Rightarrow$ $g_i = G_{iii}$, $p_i = P_{iii}$ $\Rightarrow$ FAC#





$\Delta^{Cat} = 7d.$

$\Delta^{2} = 11d.$