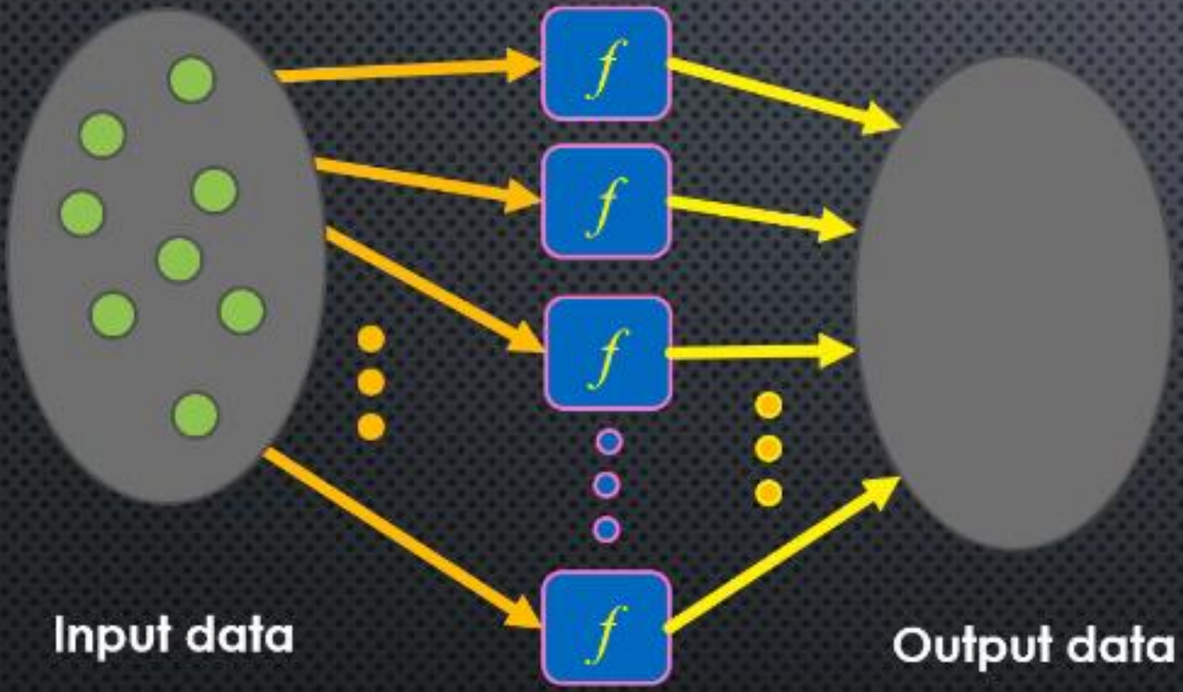
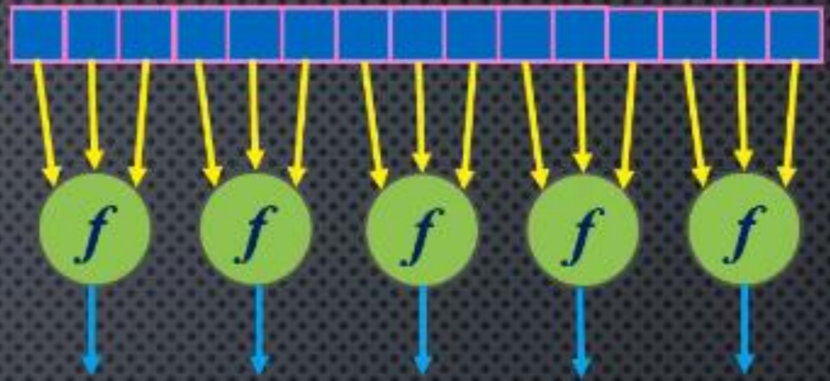
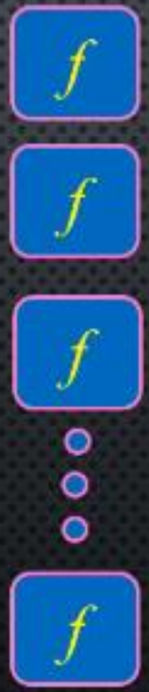
COMBINATIONAL LOOP

Lecture – 6

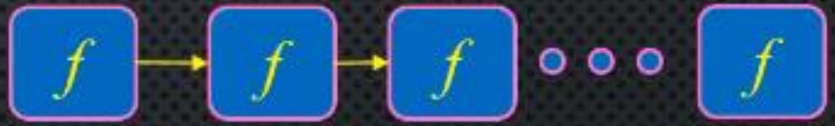
REPETITIVE PATTERN





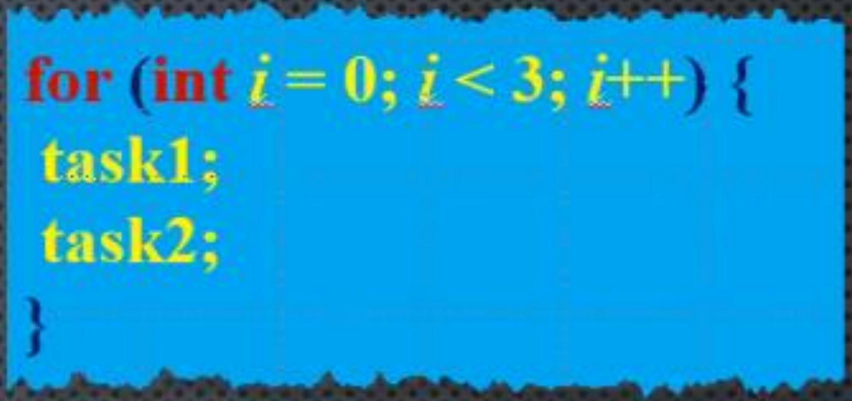


Parallel in Hardware

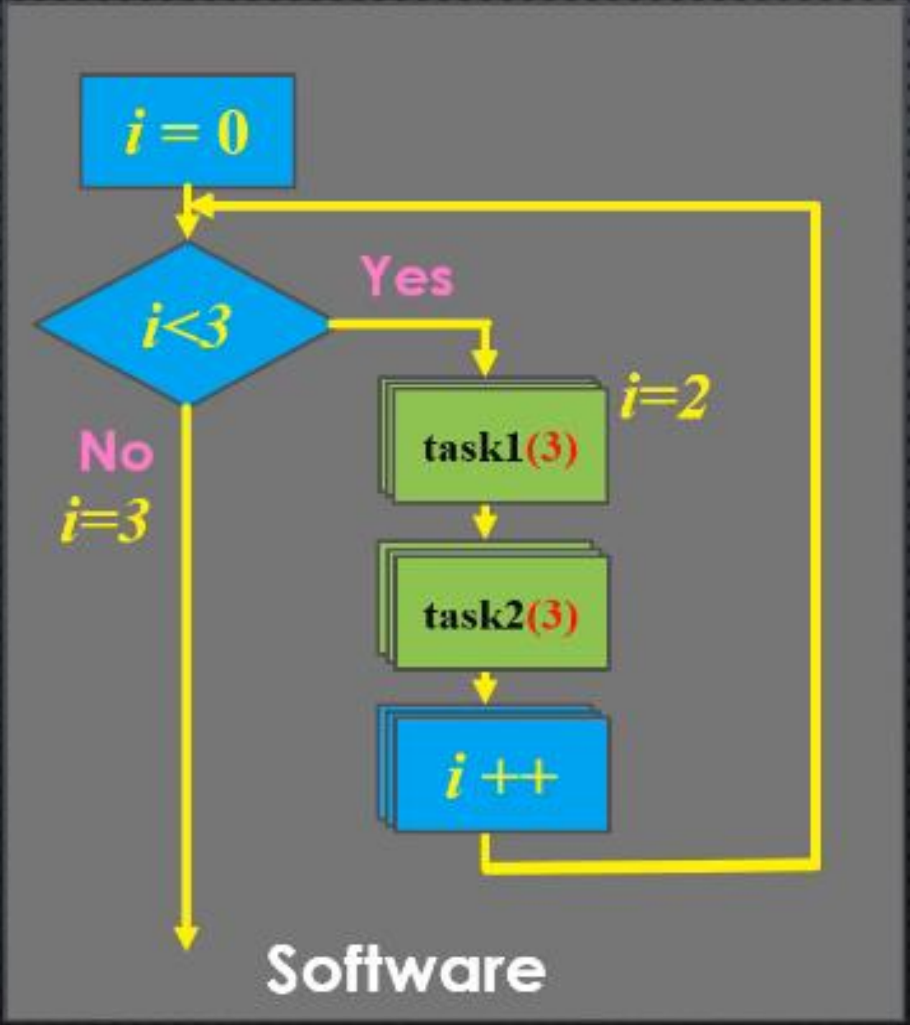


Sequential in Software

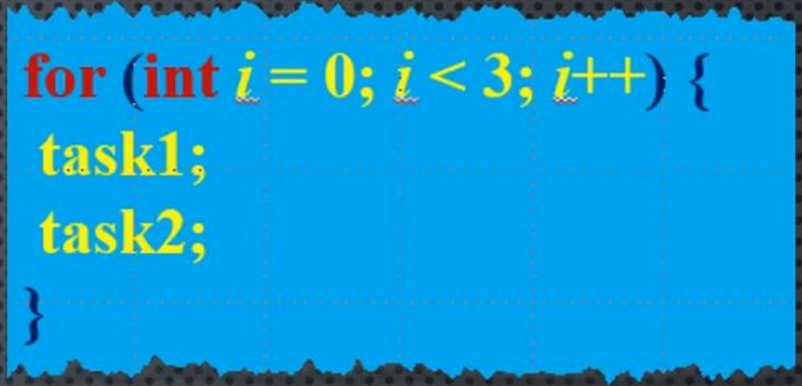
REPETITIVE PARALLEL PATTERN



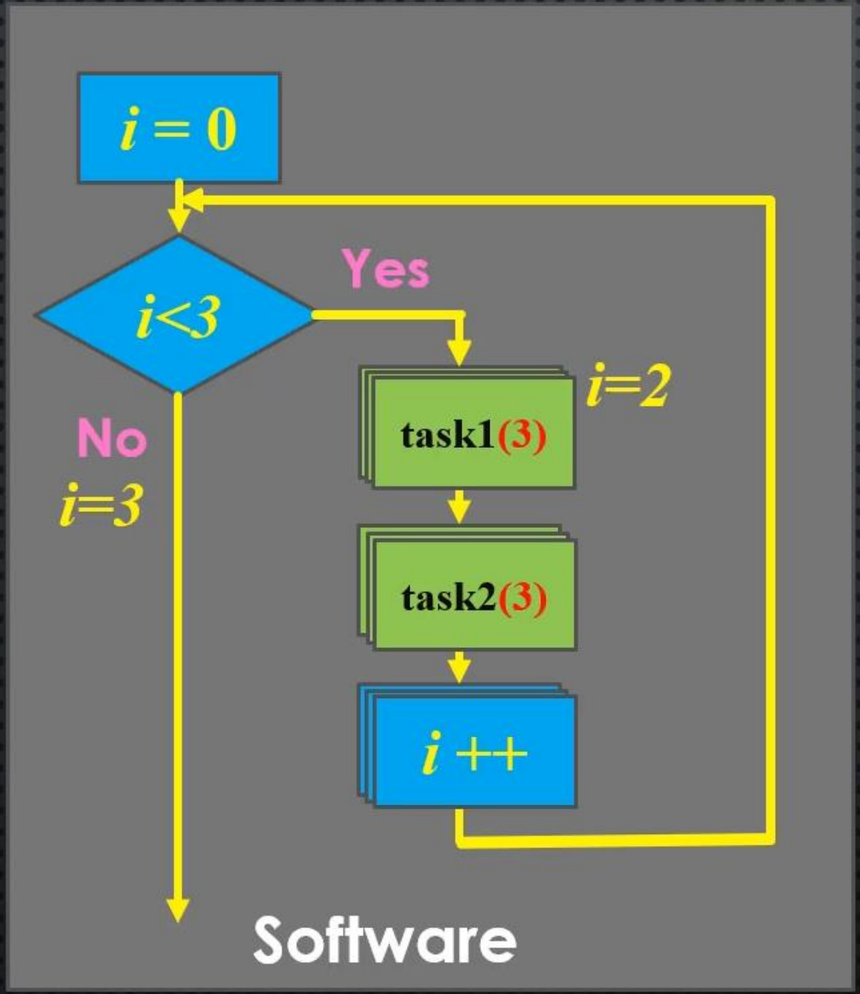
```
for (int i = 0; i < 3; i++) {
 task1;
 task2;
}
```

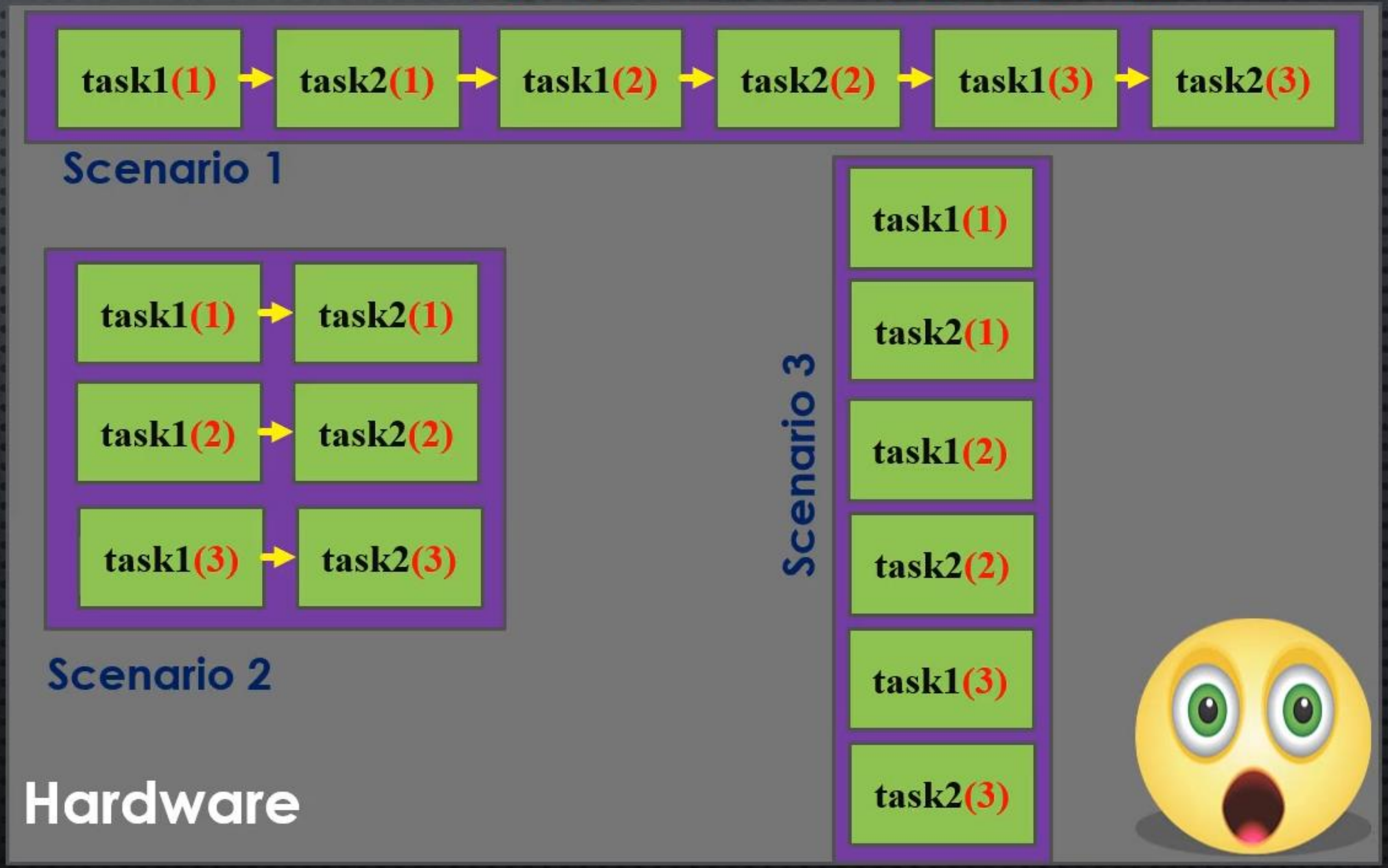


REPETITIVE PARALLEL PATTERN



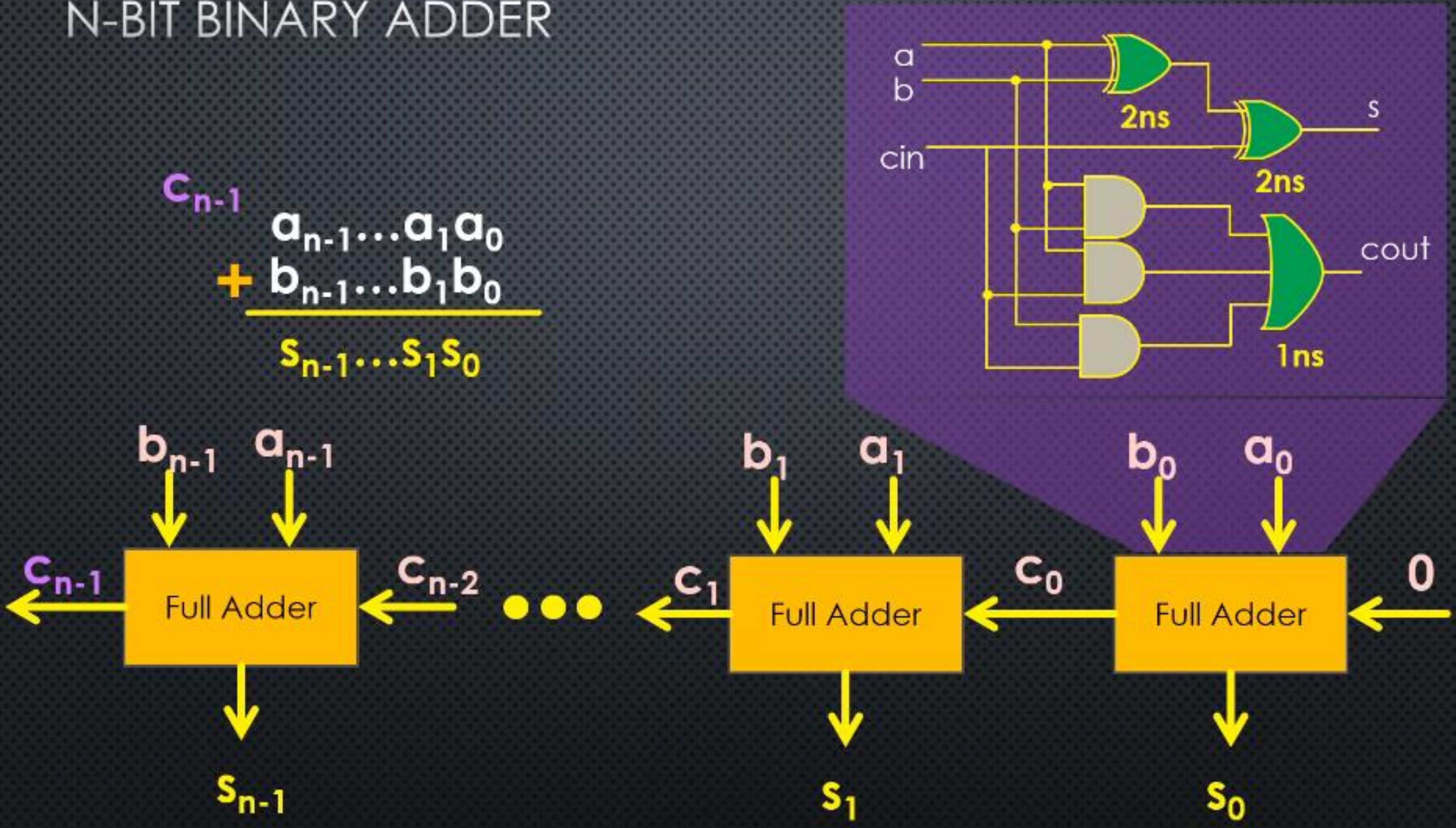
```
for (int i = 0; i < 3; i++) {
 task1;
 task2;
}
```



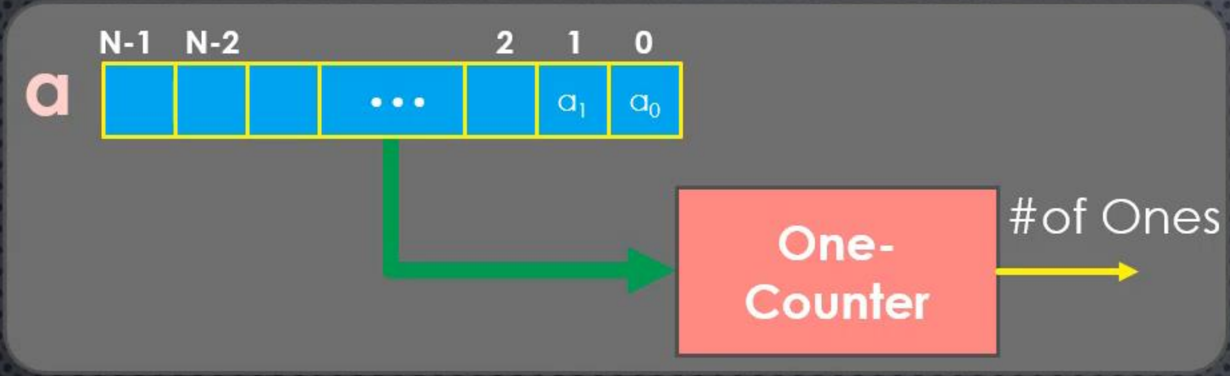


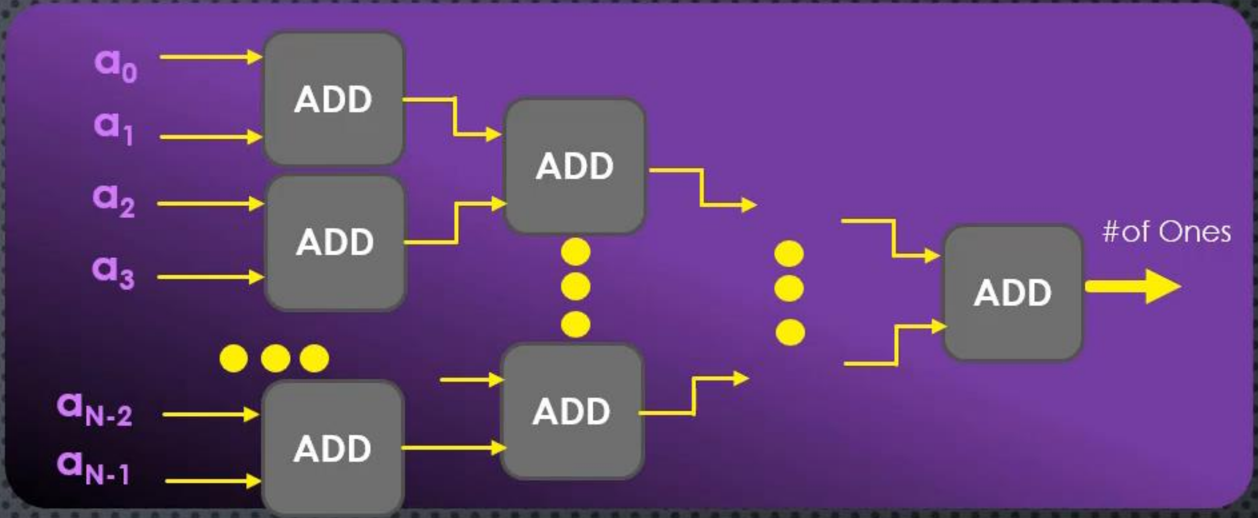
N-BIT BINARY ADDER

$$\begin{array}{lr} c_{n-1} & \\ & a_{n-1}\ldots a_1 a_0 \\ + & b_{n-1}\ldots b_1 b_0 \\ \hline & s_{n-1}\ldots s_1 s_0 \end{array}$$



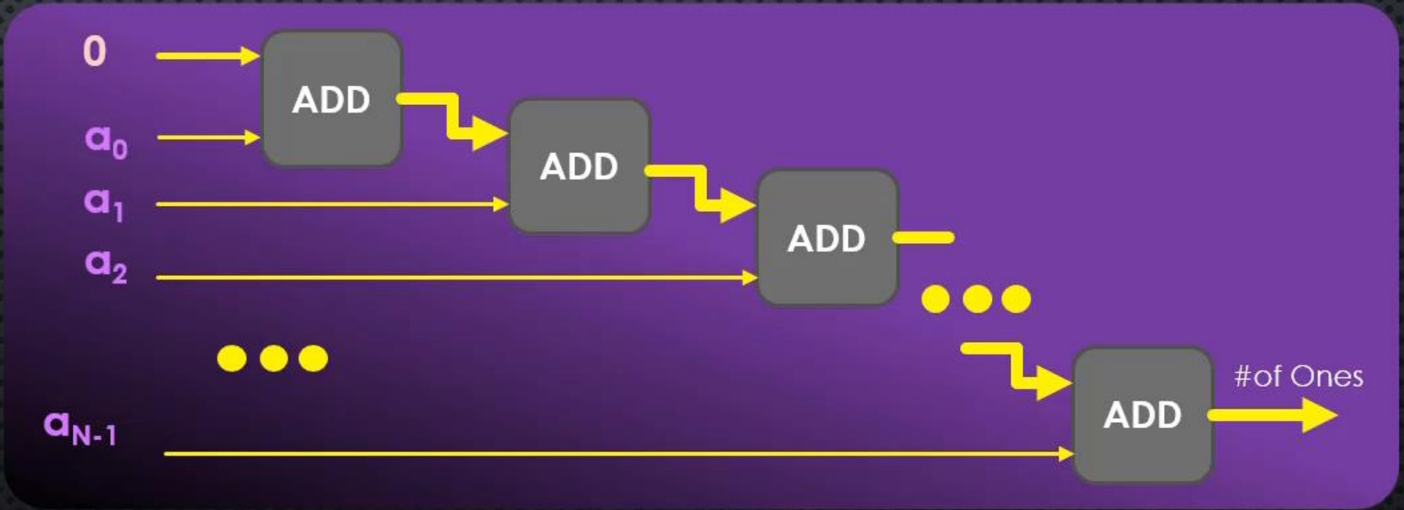
N-BIT ONE COUNTER





log*N* levels of ADDs

$$\#of\ Ones = \sum_{i=0}^{i<N} a_i$$

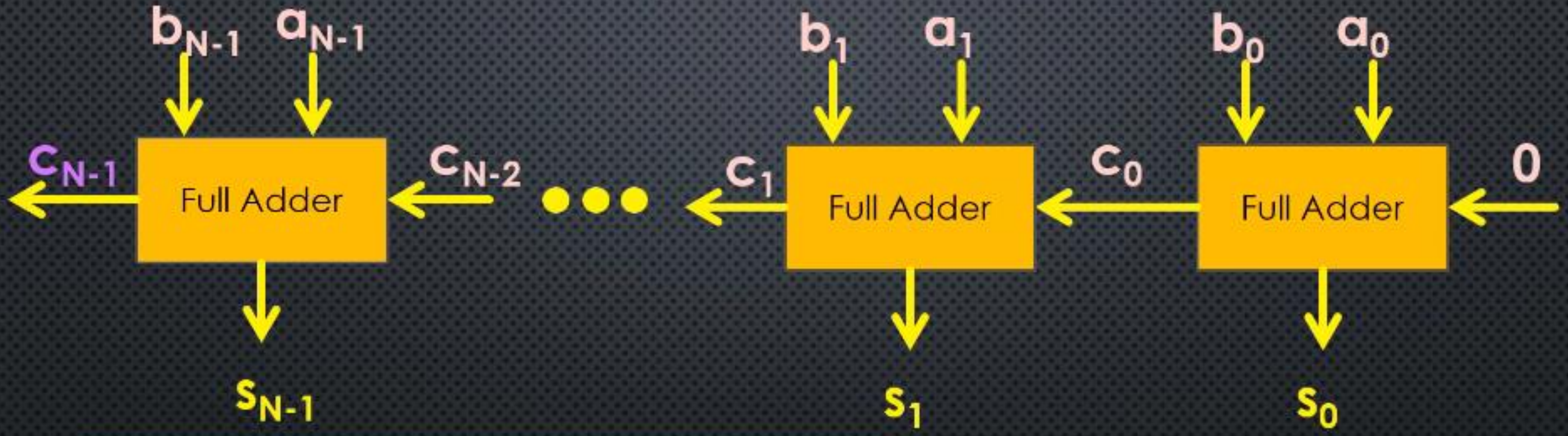


***N* levels of ADDs**

LOOP UNROLL

Lecture 6

N-BIT BINARY ADDER

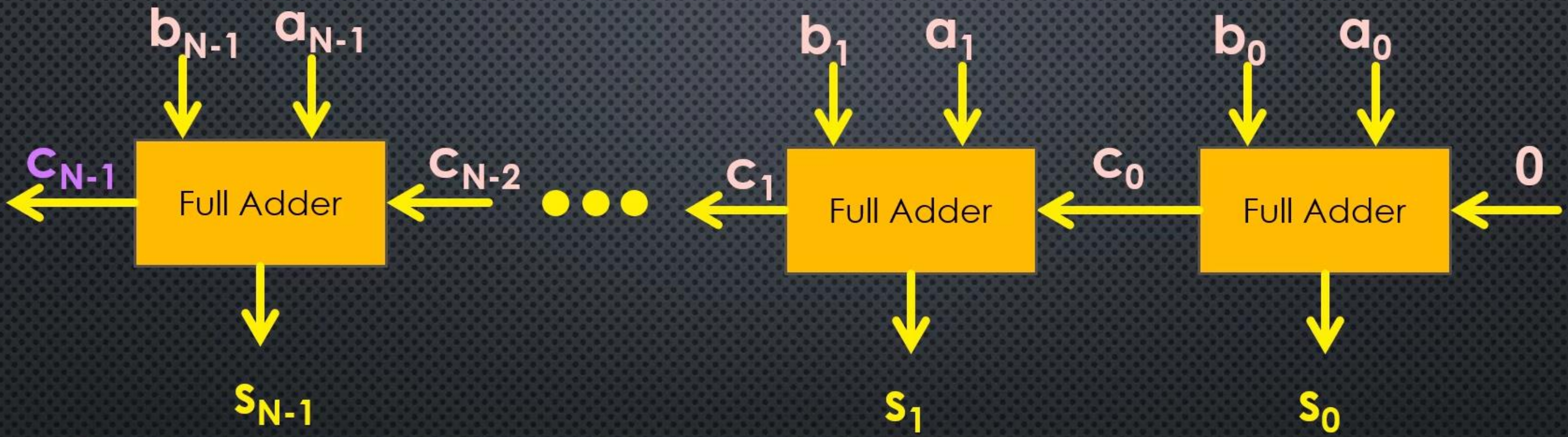


```
full_adder (a[0], b[0], 0,      c[0], s[0]);
full_adder (a[1], b[1], c[0], c[1], s[1]);
full_adder (a[2], b[2], c[1], c[2], s[2]);
full_adder (a[3], b[3], c[2], c[3], s[3]);
...   ...  ...
full_adder (a[N-1], b[N-1], c[N-2], c[N-1], s[N-1]);
```

Two issues with this code

- Not parametric
- Not generalizable

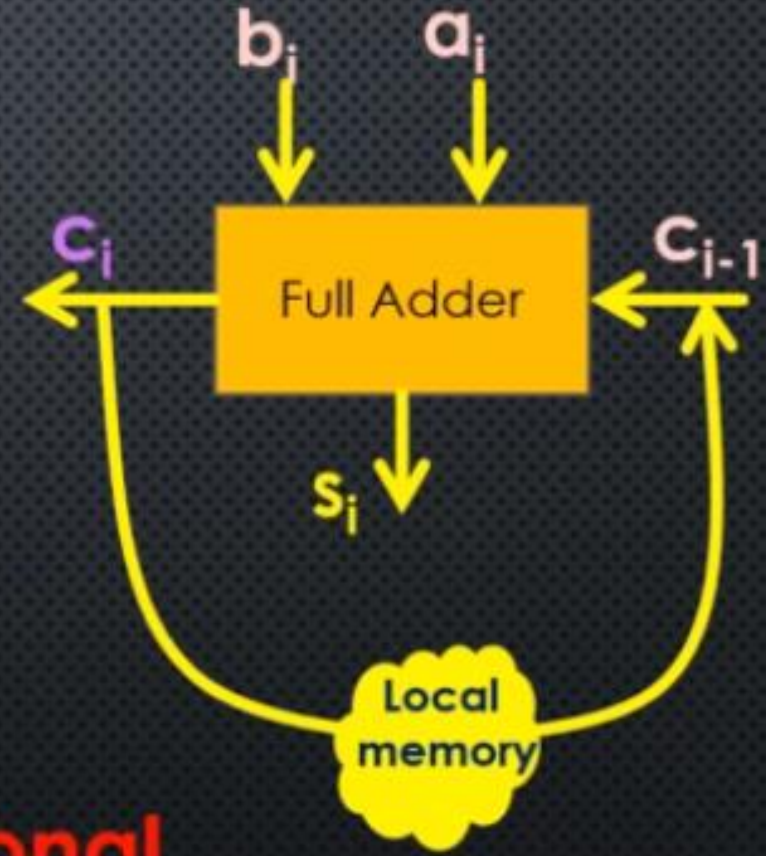
N-BIT BINARY ADDER WITH *FOR*-LOOP



```
full_adder(a[0], b[0], 0, c[0], s[0]);

for (int i = 1; i < N; i++) {
  full_adder(a[i], b[i], c[i-1], c[i], s[i]);
}
```

N-BIT BINARY ADDER WITH *FOR*-LOOP

```
full_adder(a[0], b[0], 0, c[0], s[0]);

for (int i = 1; i < N; i++) {
  full_adder(a[i], b[i], c[i-1], c[i], s[i]);
}
```

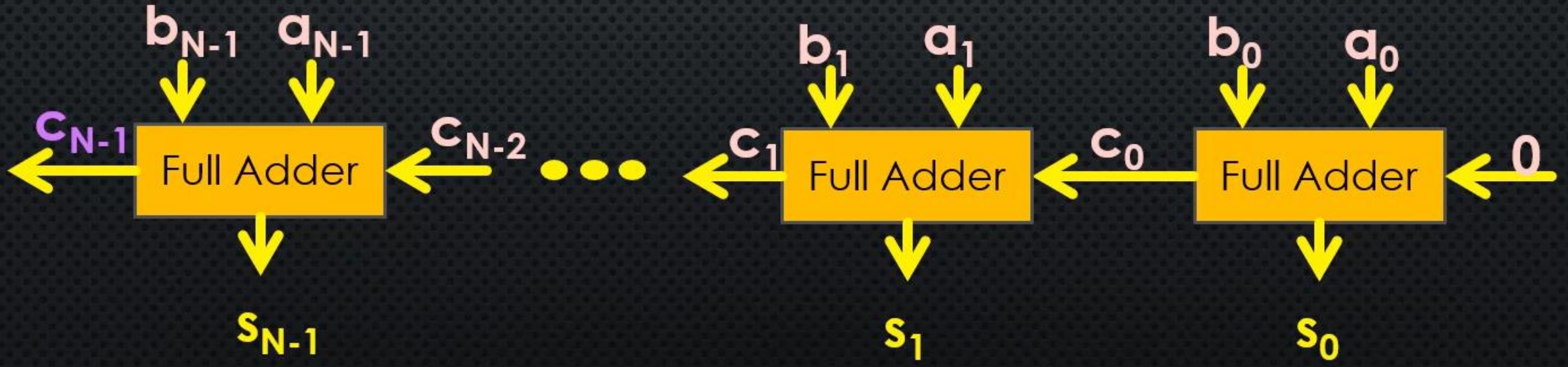


Not Combinational

FOR UNROLLING CONCEPT

```
full_adder(a[0], b[0], 0, c[0], s[0]);

for (int i = 1; i < N; i++) {
#pragma HLS UNROLL
  full_adder(a[i], b[i], c[i-1], c[i], s[i]);
}
```

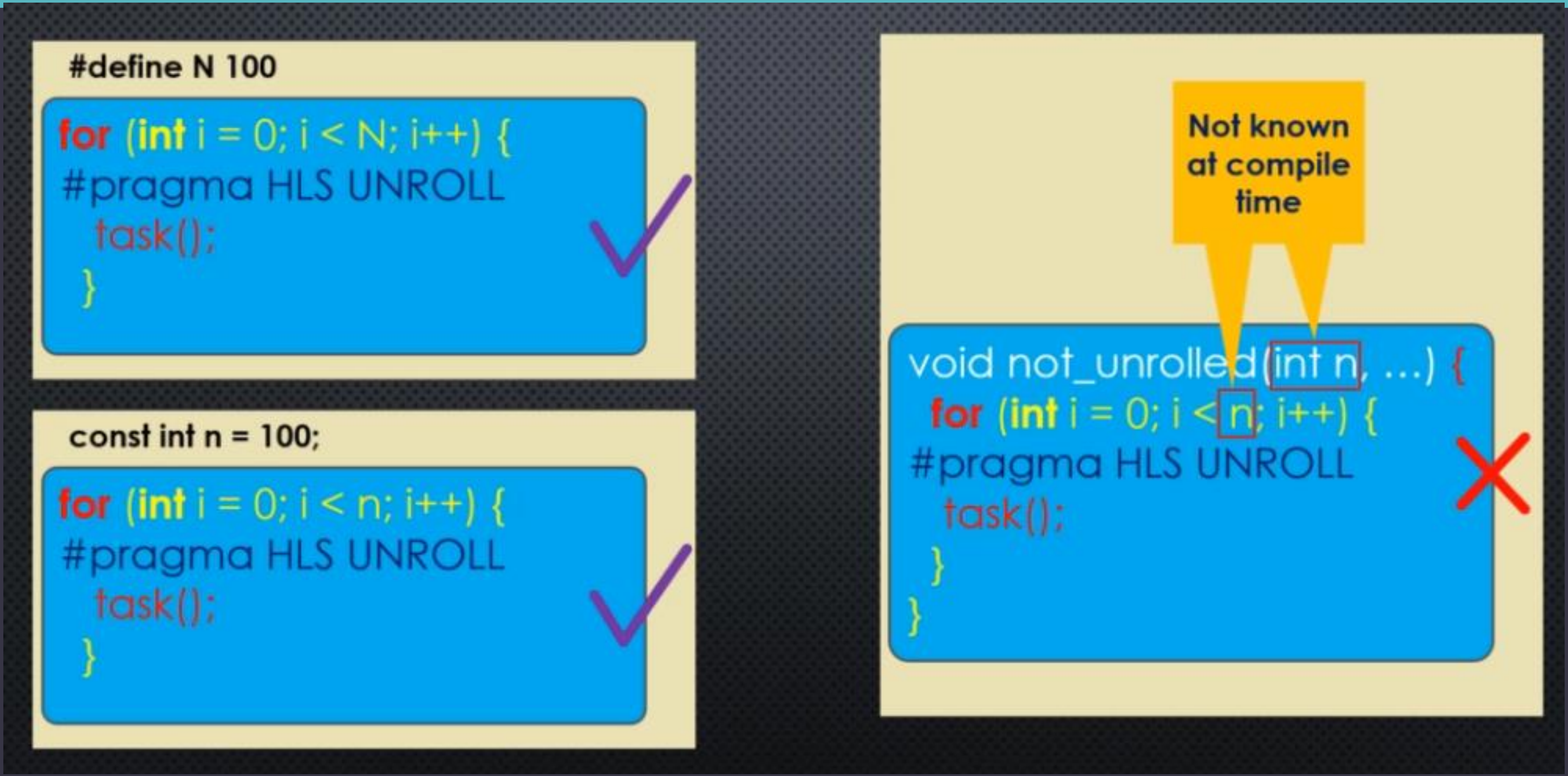
```
full_adder (a[0], b[0], 0,      c[0], s[0]);


full_adder (a[1], b[1], c[0], c[1], s[2]);
full_adder (a[2], b[2], c[1], c[2], s[3]);
full_adder (a[3], b[3], c[2], c[3], s[4]);
...    ...  ...
full_adder (a[N-1], b[N-1], c[N-2], c[N-1], s[N-1]);
```



LOOP UNROLL CONDITION

Lecture-6

STATIC FOR LOOP UNROLLING



```
#define N 100
for (int i = 0; i < N; i++) {
#pragma HLS UNROLL
   task();
 }
```

✓

```
const int n = 100;
for (int i = 0; i < n; i++) {
#pragma HLS UNROLL
   task();
 }
```

✓

Not known at compile time

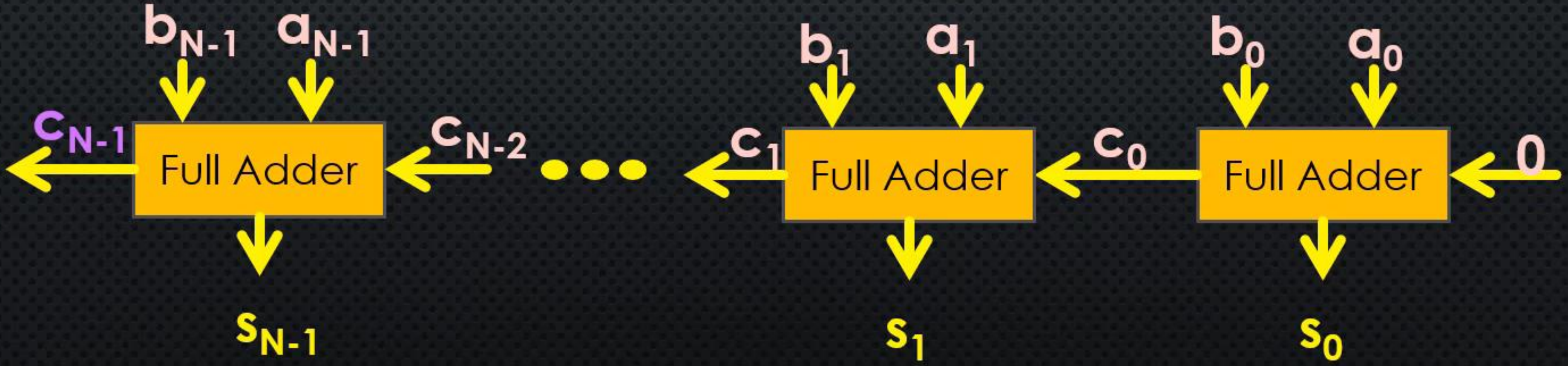
```
void not_unrolled(int n, ...) {
  for (int i = 0; i < n; i++) {
#pragma HLS UNROLL
   task();
  }
}
```

✗

FOR-LOOP UNROLLING

```
full_adder(a[0], b[0], 0, c[0], s[0]);

for (int i = 1; i < N; i++) {
#pragma HLS UNROLL
  full_adder(a[i], b[i], c[i-1], c[i], s[i]);
}
```

```
full_adder (a[0], b[0], 0,      c[0], s[1]);


full_adder (a[1], b[1], c[0], c[1], s[2]);
full_adder (a[2], b[2], c[1], c[2], s[3]);
full_adder (a[3], b[3], c[2], c[3], s[4]);
...   ...   ...
full_adder (a[N-1], b[N-1], c[N-2], c[N-1], s[N-1]);
```



FOR-LOOP UNROLLING SIZE

FPGA

LUT available = 20800

```
 for (int i = 0; i < N; i++) {
#pragma HLS UNROLL
  task();
 }
```

Task needs 100 LUTs

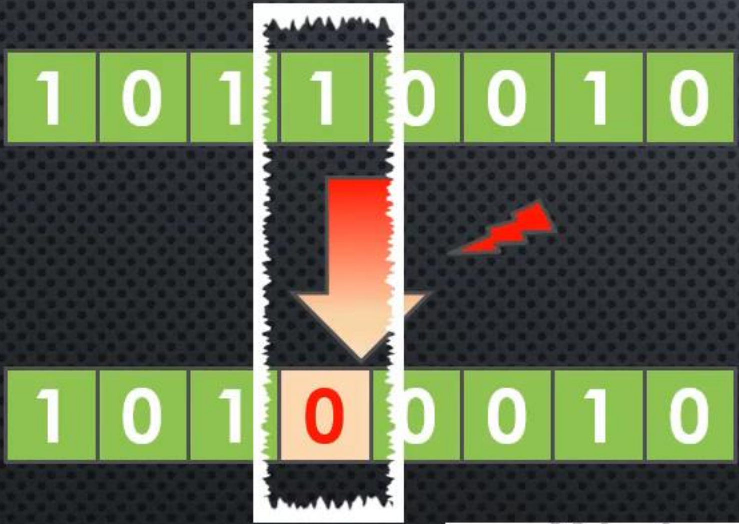
If N = 10 → Used LUTs = 1000 < 20800 ✓

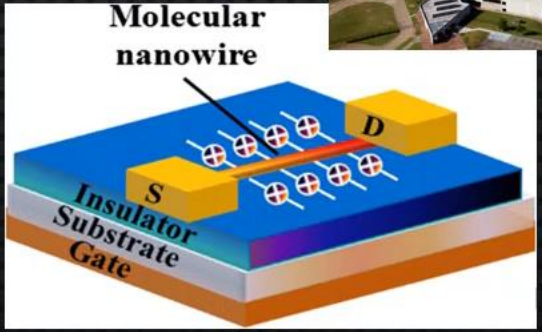
If N = 1000 → Used LUTs = 100000 > 20800 ✗

PARITY BIT DEFINITION

Lecture-6

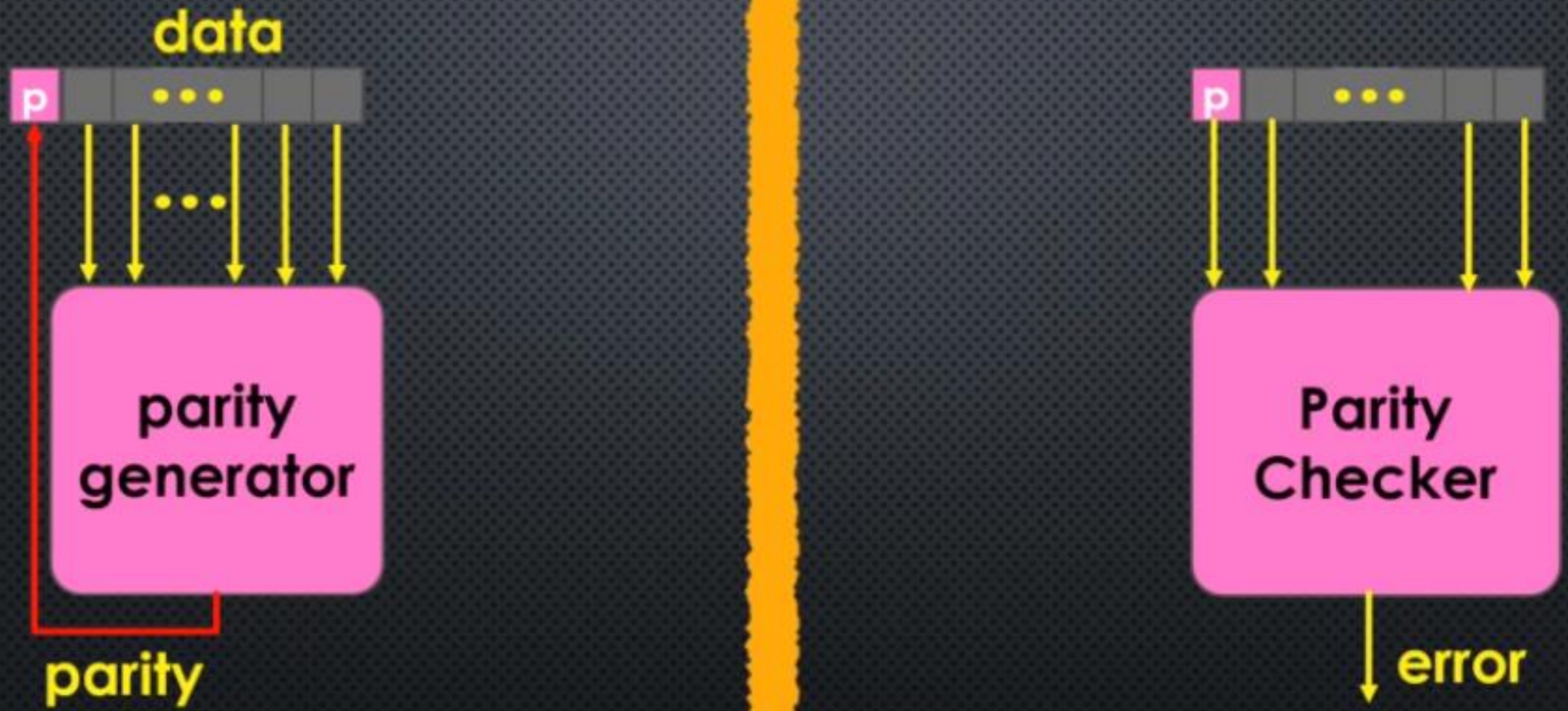
SINGLE BIT ERROR





A single event upset in the flight computers of Airbus A330 on 7 October 2008 is suspected to result in an aircraft upset that nearly ended in its crashing after the computers underwent several malfunctions.

WHAT IS PARITY?



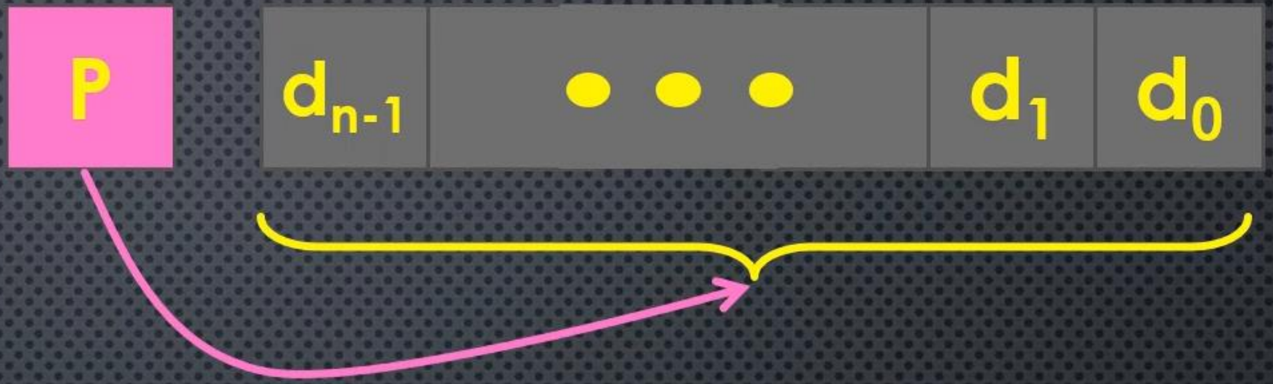
Data Producer Side

Data Consumer Side

PARITY TYPES

Parity

- odd
- even



makes #of “1”s odd or even

Examples

7 bits of data	(count of 1-bits)	8 bits including parity	
		even	odd
1101001	4	**0**1101001	**1**1101001
1011101	5	**1**1011101	**0**1011101

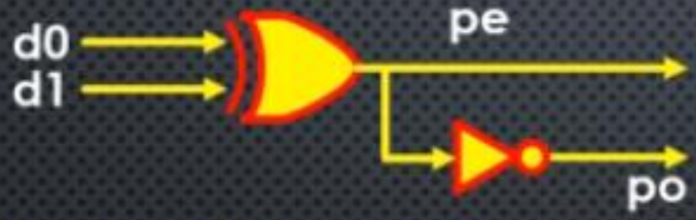
PARITY BIT DESIGN

Lecture-6

PARITY GENERATOR IDEA

p | d1 | d0

d0	d1	pe	po
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1



Two-bit parity generator

pe = d0 XOR d1;

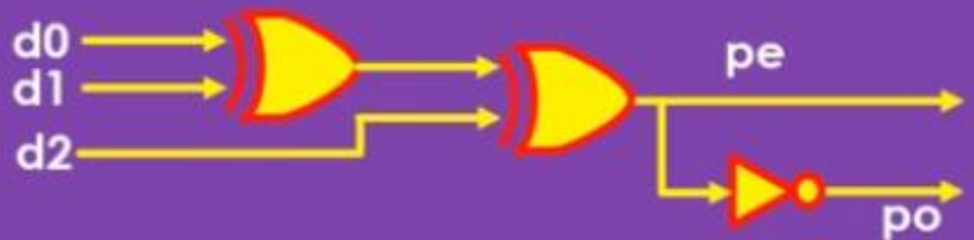
po = NOT(d0 XOR d1);

C code

```
pe = d0 ^ d1;
po = ~(d0 ^ d1);
```

p | d2 | d1 | d0

Three-bit parity generator

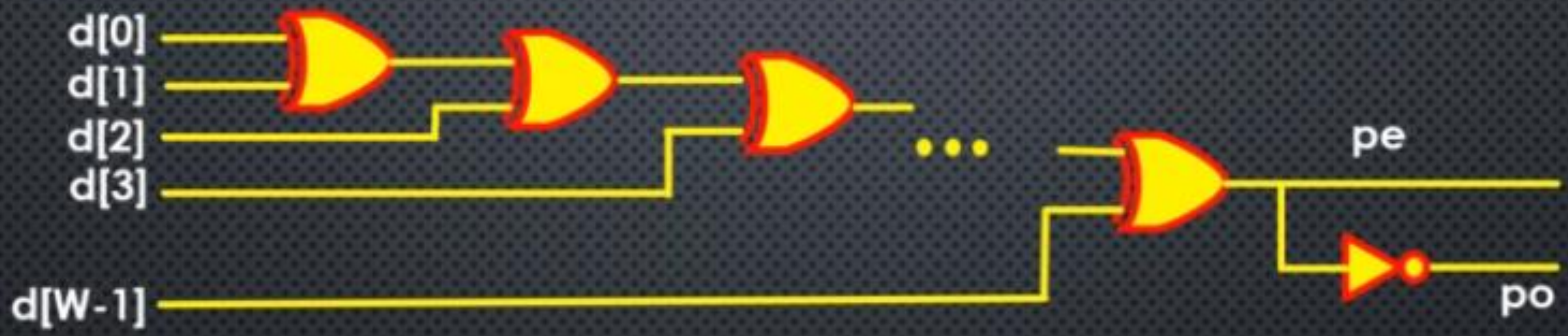


po = NOT(d0 XOR d1 XOR d3);

C code

```
po = ~(d0 ^ d1^d3);
```

PARITY FOR W-BIT DATA

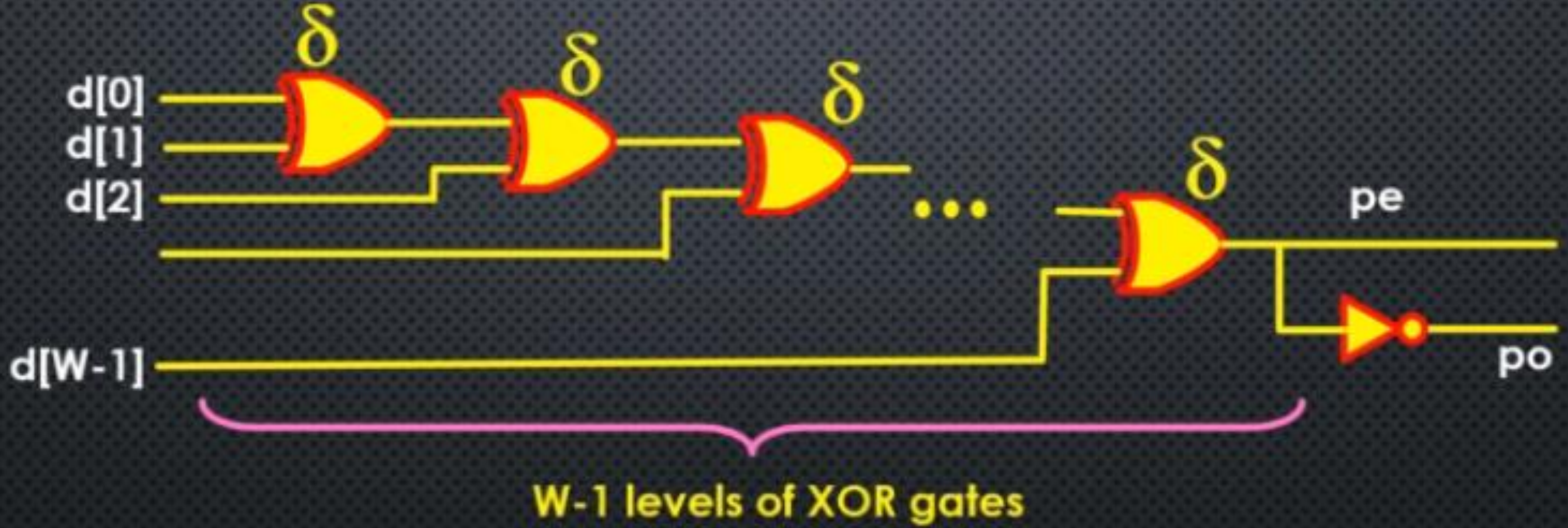


p = d[0] XOR d[1] XOR d[2] XOR d[3] ... XOR a[W-1]

d = 0b11001100 → pe= 1 ^ 1 ^ 0 ^ 0 ^ 1 ^ 1 ^ 0 ^ 0 = 0

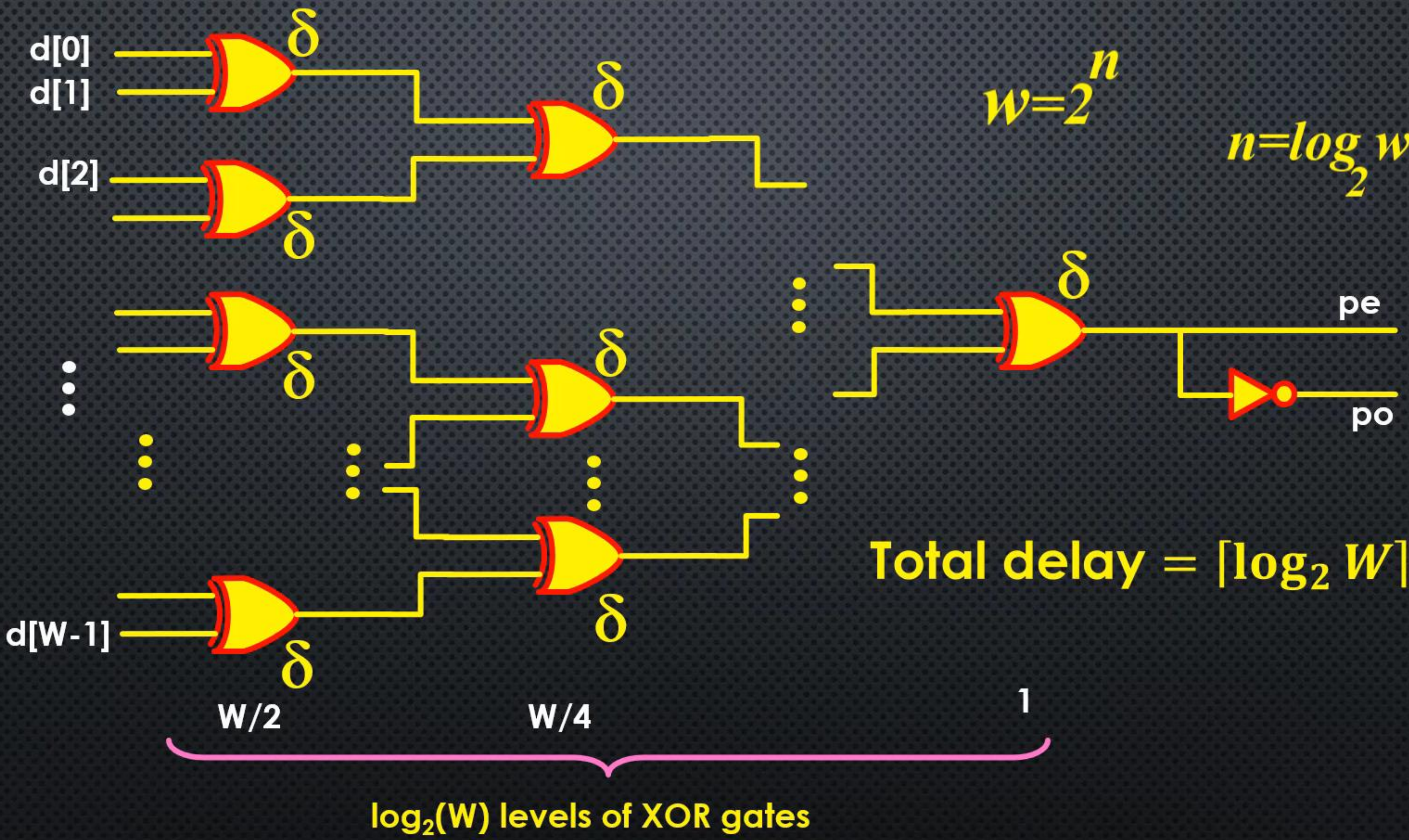
pe=0

PARITY GENERATOR: PROPAGATION DELAY



Total propagation delay = (W-1)*δ

PARITY GENERATOR : BALANCED TREE



$w=2^n$

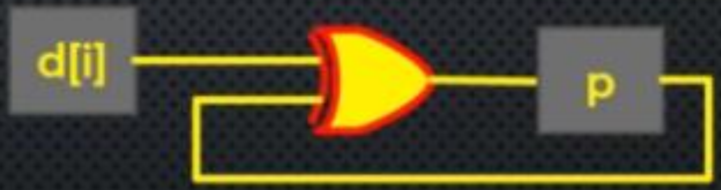
$n=\log_2 w$

Total delay = $\lceil \log_2 W \rceil * \delta$

IMPLEMENTATION IN C

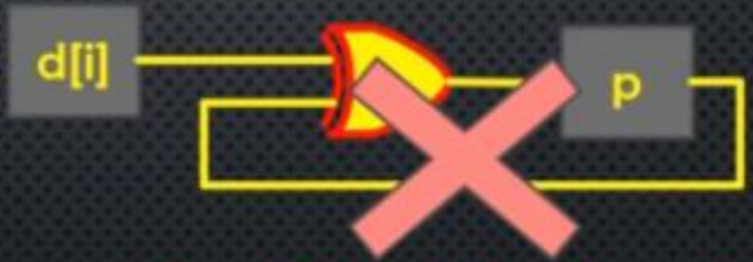
p = d[0] XOR d[1] XOR d[2] XOR d[3] … XOR d[w-1]

```
#define W 32
p = 0;
for (i = 0; i < W; i++)
 p = p ^ d[i];
```



IMPLEMENTATION IN C

```
#define W 32
p = 0;
for (i = 0; i < W; i++)
  p = p ^ d[i];
```

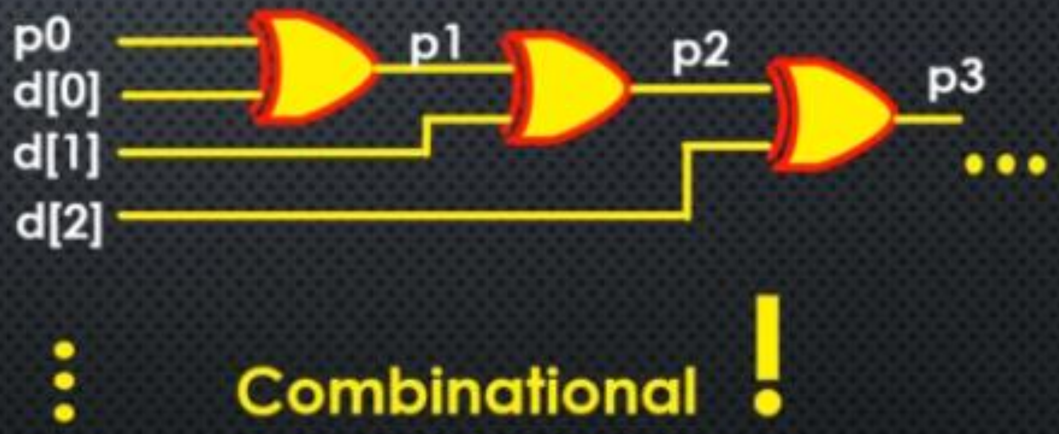


Not Combinational

IMPLEMENTATION CONCEPT

Loop unrolling

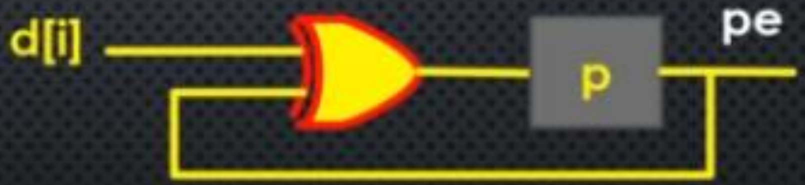
```
p0 = 0;
p1 = p0 ^ d[0];
p2 = p1 ^ d[1];
p3 = p2 ^ d[2];
....
```



Combinational !

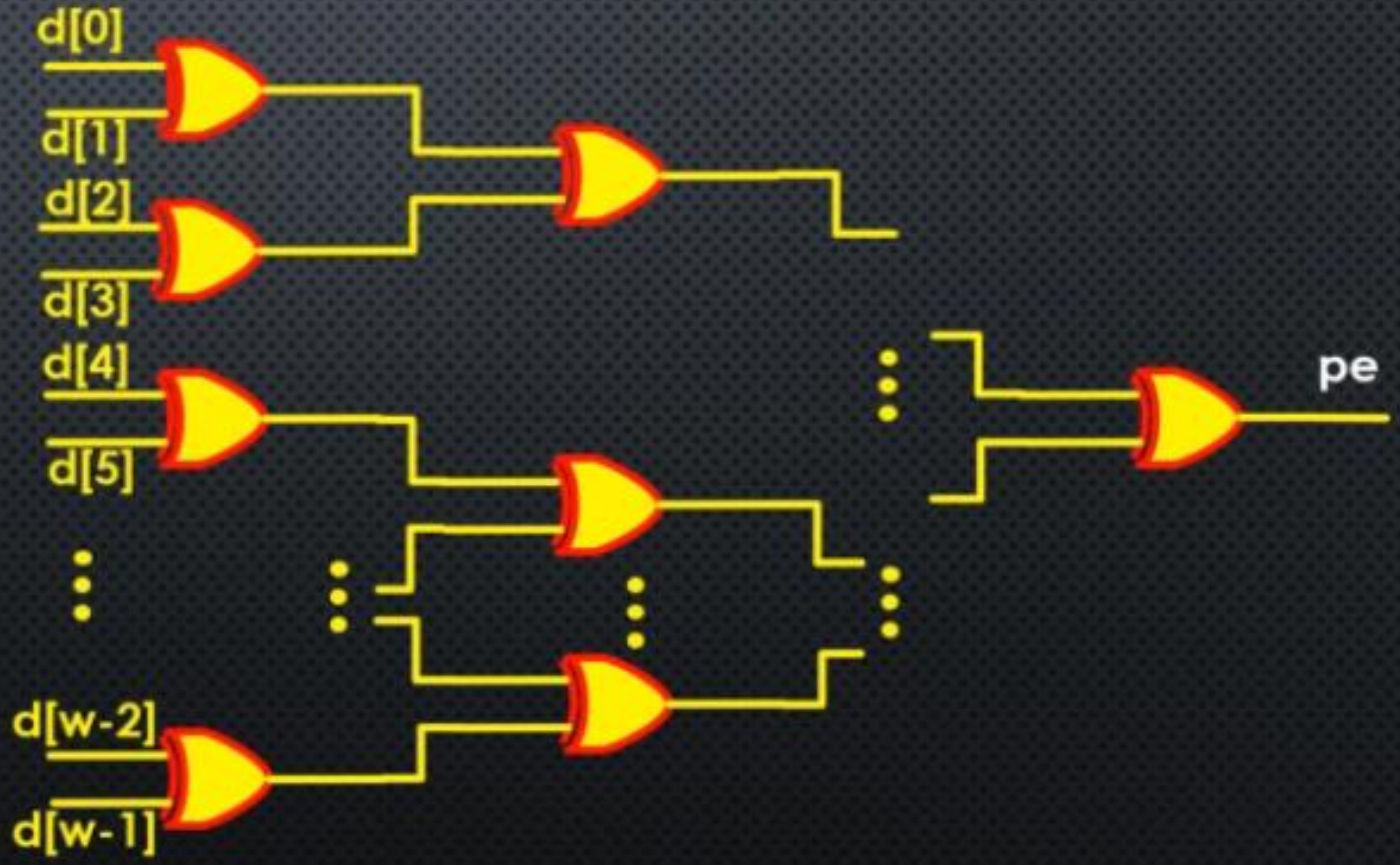
IMPLEMENTATION IN HLS

```
const int w = 32;
p = 0;
for (i = 0; i < W; i++)
#pragma HLS UNROLL
  p = p ^ d[i];
```



IMPLEMENTATION IN HLS

```
const int w = 32;
p = 0;
for (i = 0; i < W; i++)
#pragma HLS UNROLL
  p = p ^ d[i];
```



PARITY BIT DESIGN-VIVADO HLS IMPLEMENTATION

Lecture-6

Any Question...

Thank you