# INTEGER ARITHMETIC

Lecture-7

# OPERATORS

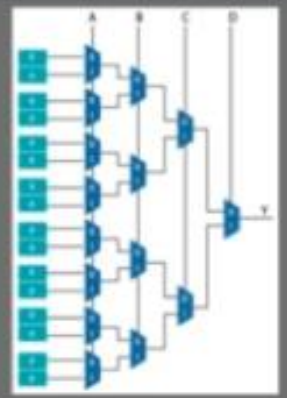
```
ap_int<32>  a;
ap_int<32>  b;
ap_int<32>  c;

a = 12;
b = 345;
c = a * b;
```

```
void arith20 (
    ap_int<20> A,      ap_int<20> B,
    ap_int<20> C,      ap_int<20> D,
    ap_int<20> *out1, ap_int<20> *out2,
    ap_int<20> *out3
) {
    // Basic arithmetic operations
    *out1 = A + B;
    *out2 = A - B;
    *out3 = C * D
}
```

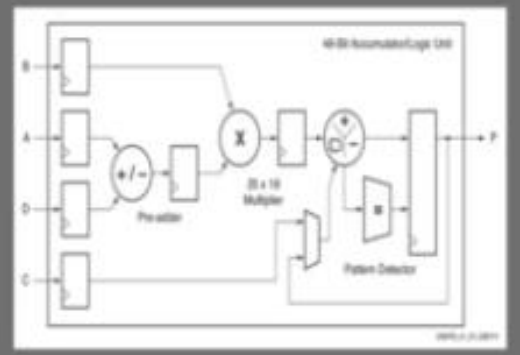
# RESOURCE ALLOCATION

Addition (+)
Subtraction (-)
Multiplication (*)
Division (/)
Modulus (%)

LUT



DSP



a

b

Operator

f

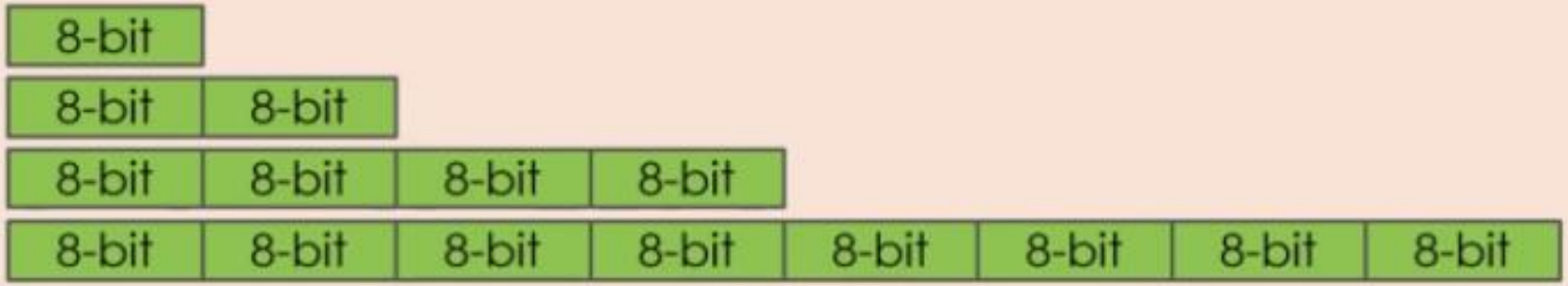
```
#pragma HLS RESOURCE variable= f core=…
```

# BASIC ARITHMETIC OPERATORS

| | |
|---|---|
| Addition | a+b |
| Subtraction | a-b |
| Multiplication | a*b |
| Division | a/b |
| Modulus (Reminder ) | a%b |

**Basic arithmetic operators can almost applied on all data types defined in HLS.**
**For user defined data type the operators should be implemented by users.**

C-based native data types are all on 8-bit boundaries:



# BASIC ARITHMETIC OPERATORS EXAMPLES

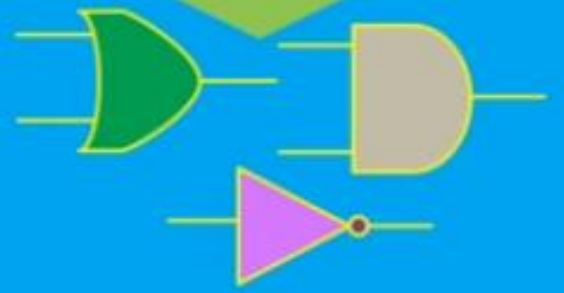
```
typedef int DTYPE;

void arith (
 DTYPE A, DTYPE B, DTYPE C, DTYPE D,
 DTYPE *out1, DTYPE *out2,
 DTYPE *out3, DTYPE *out4
) {
    // Basic arithmetic operations
    *out1 = A * B;
    *out2 = B + A;
    *out3 = C / A;
    *out4 = D % A;
}
```

```
typedef int     DTYPE1;
typedef short DTYPE2;
typedef char DTYPE3;

void arith (
 DTYPE3 A, DTYPE2 B,  DTYPE1 C, DTYPE3 D,
 DTYPE1 *out1, DTYPE1 *out2,
 DTYPE1 *out3, DTYPE1 *out4
) {
    // Basic arithmetic operations
    *out1 = A * B;
    *out2 = B + A;
    *out3 = C / A;
    *out4 = D % A;
}
```

# ADD/SUB/MUL INTEGER ARITHMETIC OPERATORS

| Data type | Bit width |
|---|---|
| bool | 1 |
| char | 8 |
| unsigned char | 8 |
| short int | 16 |
| unsigned short int | 16 |
| int | 32 |
| unsigned int | 32 |
| long int | 32 |
| unsigned long int | 32 |
| long long int | 64 |
| unsigned long long int | 64 |

Addition (+)
Subtraction (-)
Multiplication (*)

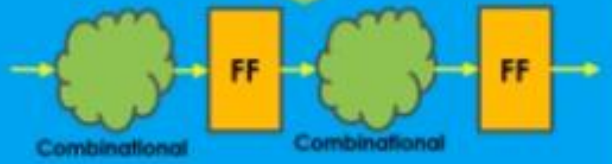


Combinational

Clock period > Design propagation delay

Resource Con

Addition (+)
Subtraction (-)
Multiplication (*)



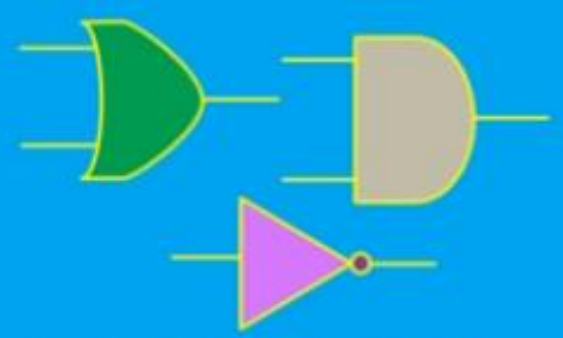

Pipelined

Clock period < Design propagation delay

Resource Co

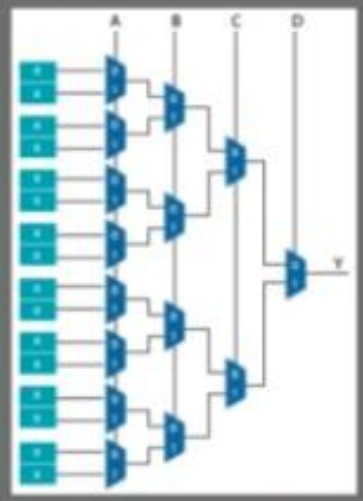
# ADD/SUB/MUL INTEGER ARITHMETIC OPERATORS

Addition (+)
Subtraction (-)
Multiplication (*)

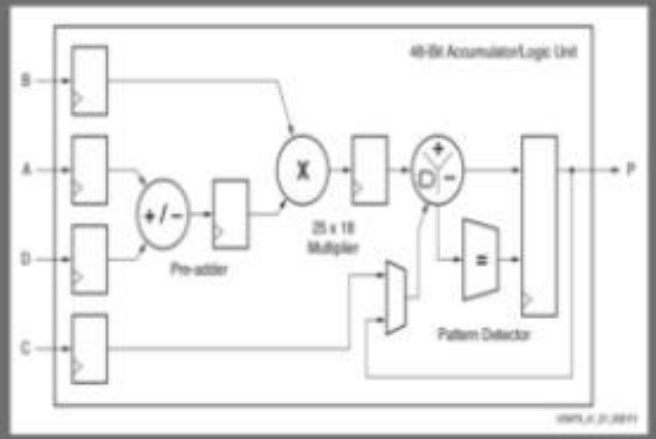


Combinational

LUT




DSP




Resource Constraint

Clock period > Design propagation delay

# OPERATOR OVERLOADING

Standard binary integer arithmetic operators are overloaded to provide arbitrary precision arithmetic.

```
ap_int<32>  a;
ap_int<32>  b;
ap_int<32>  c;

a = 12;
b = 345;
c = a * b;
```

Two operands of ap_[u]int, or One ap_[u]int type and one C/C++ basic integer data type

```
ap_int<32>  a;
int         b;
long long int   c;

a = 12;
b = 345;
c = a * b;
```

# DIFFERENT DATATYPE VERSIONS

```
void arith32 (
    int A,      int B,
    int C,      int D,
    int *out1, int *out2,
    int *out3
) {
    // Basic arithmetic operations
    *out1 = A + B;
    *out2 = A - B;
    *out3 = C * D;
}
```

```
void arith20 (
    ap_int<20> A,      ap_int<20> B,
    ap_int<20> C,      ap_int<20> D,
    ap_int<20> *out1, ap_int<20> *out2,
    ap_int<20> *out3
) {
    // Basic arithmetic operations
    *out1 = A + B;
    *out2 = A - B;
    *out3 = C * D
}
```

# PERFORMANCE COMPARISON

## arith32

Timing

Summary

| Clock | Target | Estimated | Uncertainty |
|---|---|---|---|
| ap_clk | 10.00 ns | 8.470 ns | 1.25 ns |

Latency

Utilization Estimates

Summary

| Name | BRAM_18K | DSP48E | FF | LUT | URAM |
|---|---|---|---|---|---|
| DSP | - | - | - | - | - |
| Expression | - | 3 | 0 | 99 | - |
| FIFO | - | - | - | - | - |
| Instance | - | - | - | - | - |
| Memory | - | - | - | - | - |
| Multiplexer | - | - | - | - | - |
| Register | - | - | - | - | - |
| Total | 0 | 3 | 0 | 99 | 0 |
| Available | 100 | 90 | 41600 | 20800 | 0 |
| Utilization (%) | 0 | 3 | 0 | ~0 | 0 |

Detail

## arith20

Timing

Summary

| Clock | Target | Estimated | Uncertainty |
|---|---|---|---|
| ap_clk | 10.00 ns | 5.850 ns | 1.25 ns |

Utilization Estimates

Summary

| Name | BRAM_18K | DSP48E | FF | LUT | URAM |
|---|---|---|---|---|---|
| DSP | - | - | - | - | - |
| Expression | - | 2 | 0 | 66 | - |
| FIFO | - | - | - | - | - |
| Instance | - | - | - | - | - |
| Memory | - | - | - | - | - |
| Multiplexer | - | - | - | - | - |
| Register | - | - | - | - | - |
| Total | 0 | 2 | 0 | 66 | 0 |
| Available | 100 | 90 | 41600 | 20800 | 0 |
| Utilization (%) | 0 | 2 | 0 | ~0 | 0 |

# GOOD PRACTICE

**types.h**

```
...
typedef char type1;
typedef short type2;
typedef int   type3;
...
```

**arith.cpp**

```
#include "types.h"

void arith (
    type1 A,      type1 B,
    type2 C,      type2 D,
    type3 *out1,  type3 *out2,
    type2 *out3,  type2 *out4
) {
    // Basic arithmetic operations
    *out1 = A + B;
    *out2 = A - B;
    *out3 = C / D;
}
```

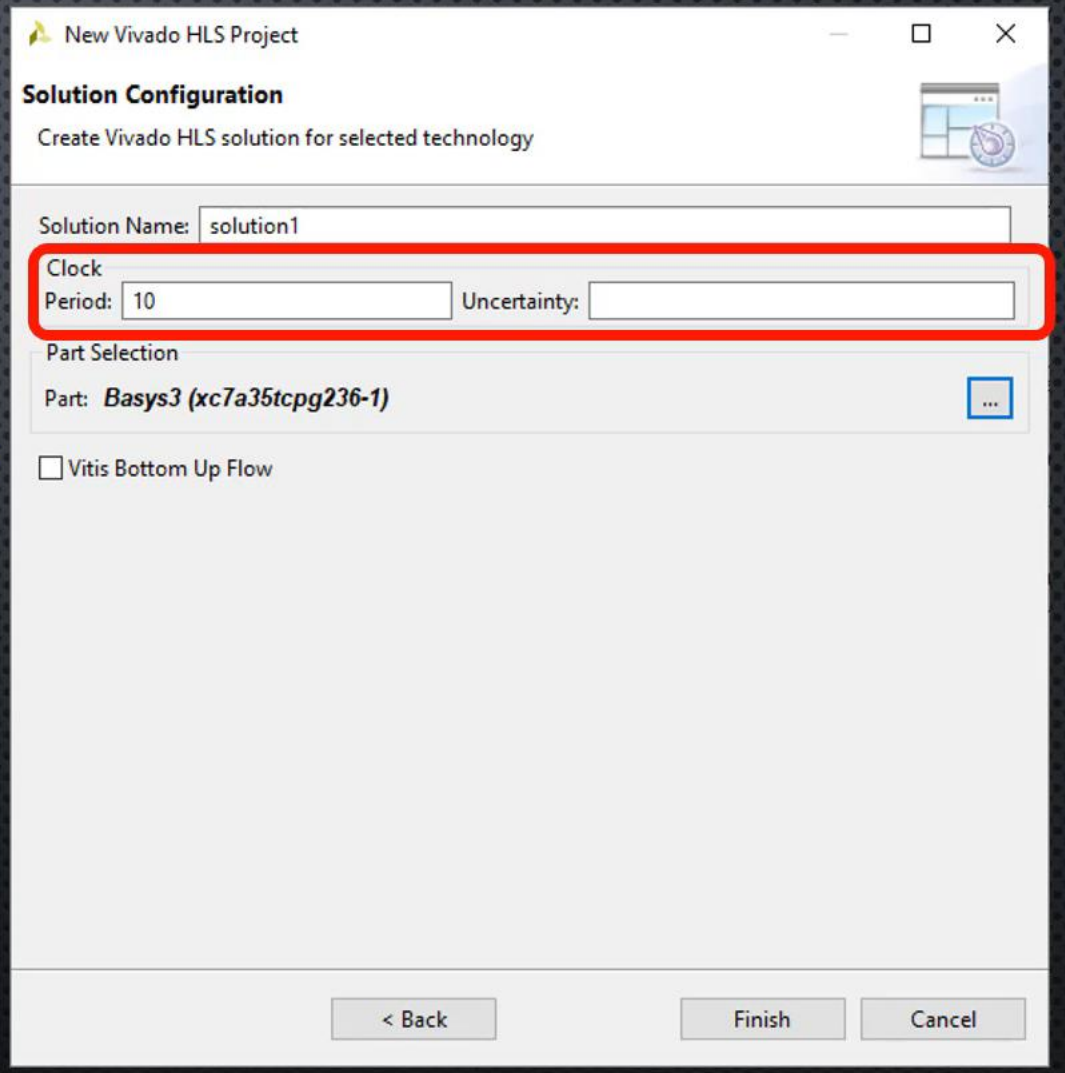
# DESIGN CLOCK CONSTRAINT

**There is a clock constraints assigned To each HLS design.**

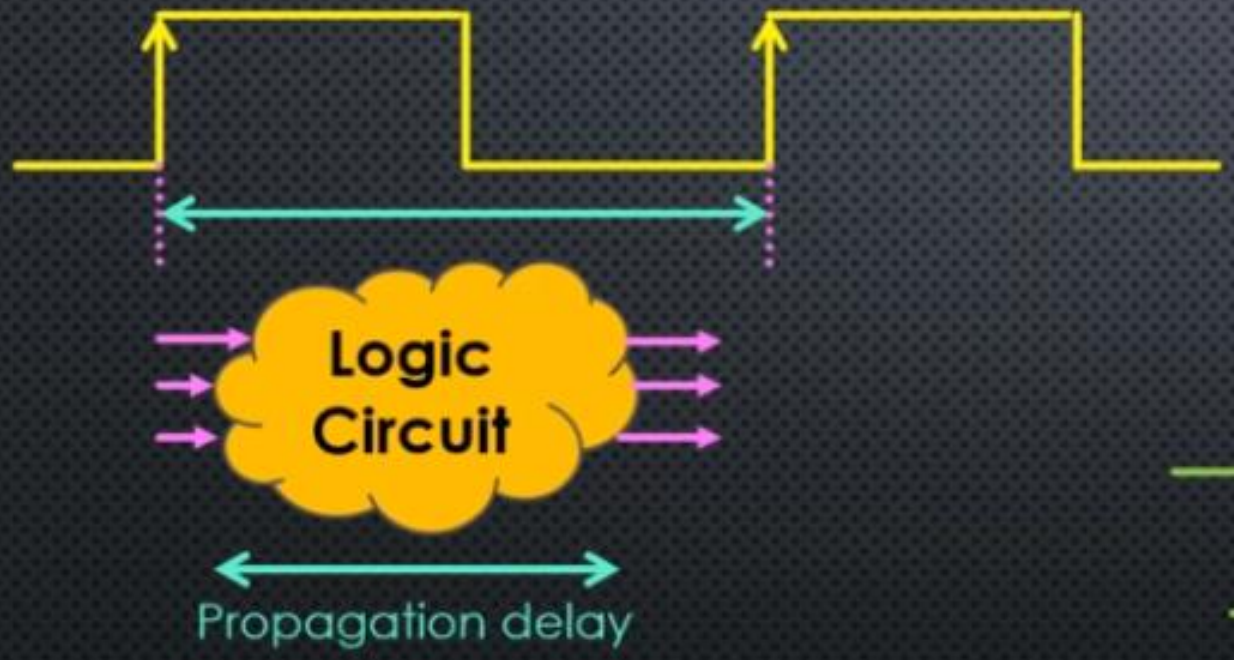

$$f\text{: Frequency (MHz)} = \frac{1}{T}$$

**Clock**

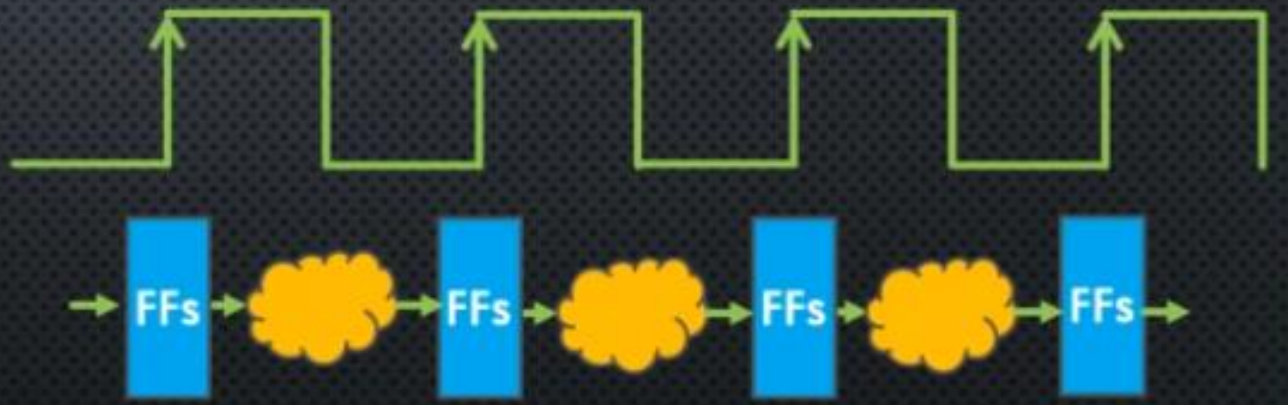
- Period in units of ns or Frequency value in MHz suffix
- Uncertainty



# DESIGN CLOCK CONSTRAINT




**Combinational**

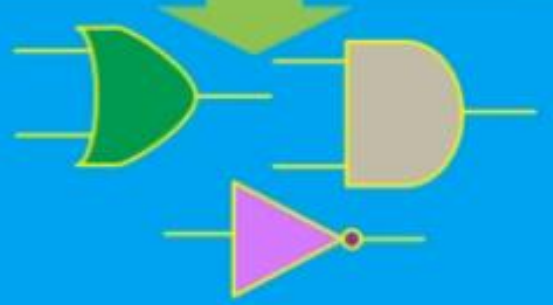



**Pipelined**

# ADD/SUB/MUL INTEGER ARITHMETIC OPERATORS

| Data type | Bit width |
|---|---|
| bool | 1 |
| char | 8 |
| unsigned char | 8 |
| short int | 16 |
| unsigned short int | 16 |
| int | 32 |
| unsigned int | 32 |
| long int | 32 |
| unsigned long int | 32 |
| long long int | 64 |
| unsigned long long int | 64 |

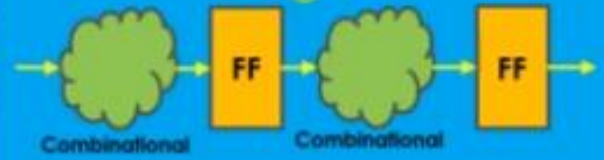
Addition (+)
Subtraction (-)
Multiplication (*)



Combinational

Clock period > Design propagation delay

Addition (+)
Subtraction (-)
Multiplication (*)




Pipelined

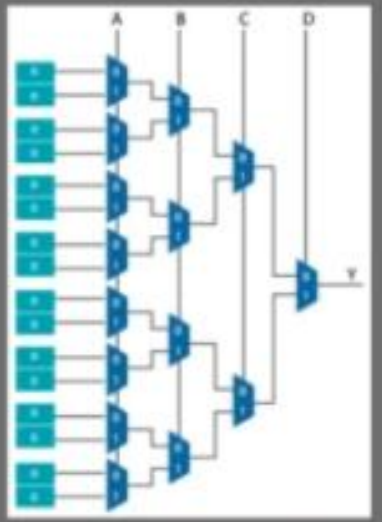
Clock period < Design propagation delay

# DSP RESOURCES

# ARITHMETIC RESOURCES

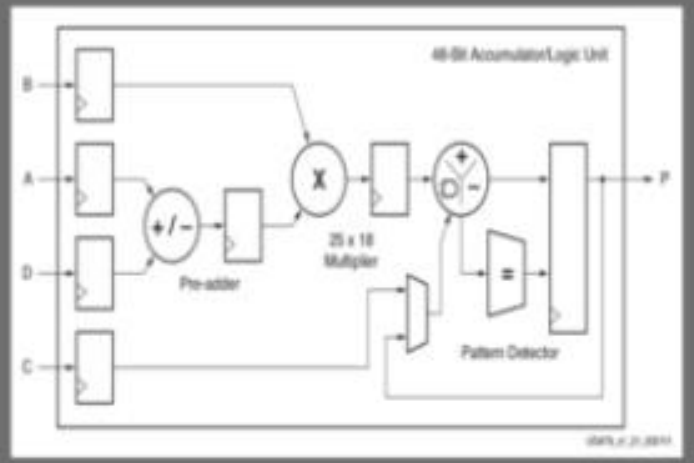
Addition (+)
Subtraction (-)
Multiplication (*)
Division (/)
Modulus (%)

LUT



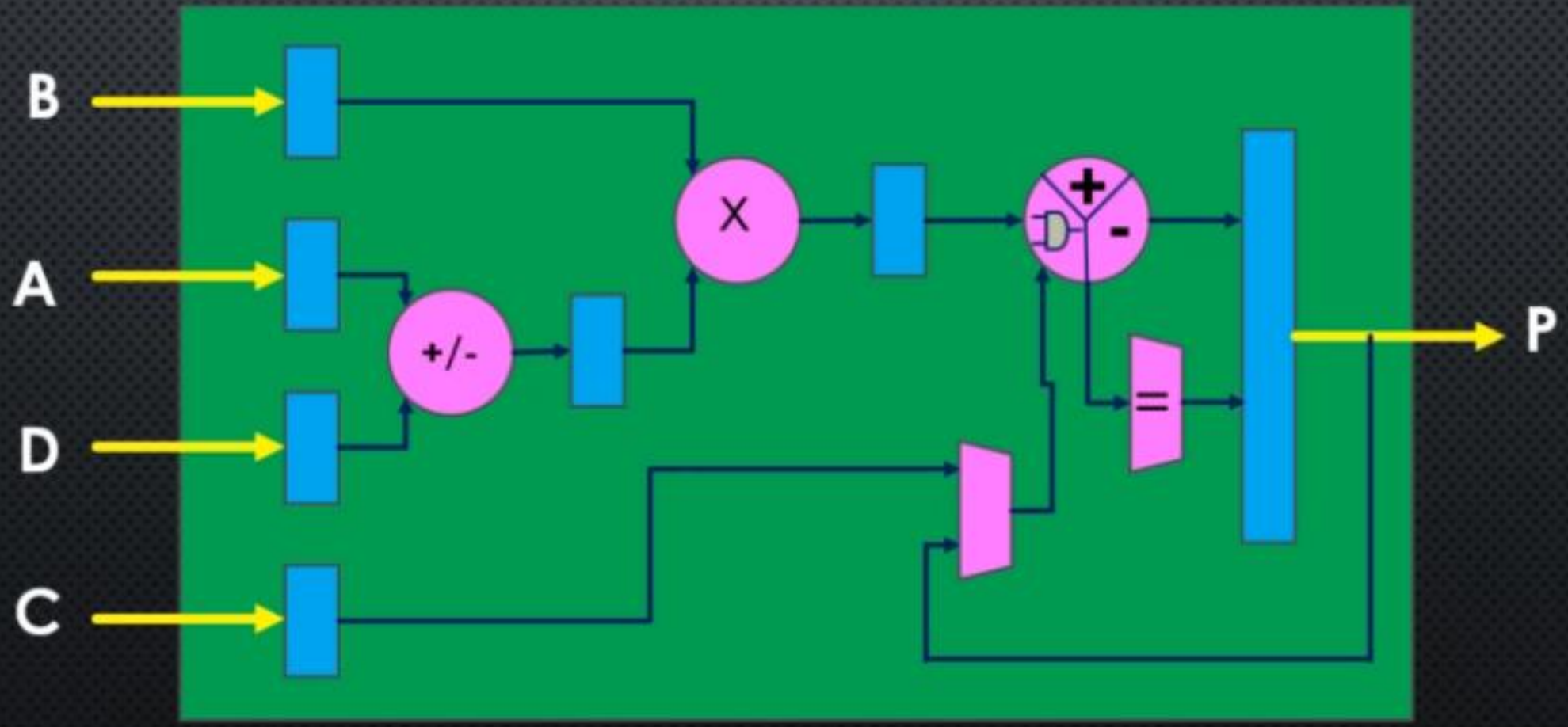

Logic-Gates in the form of LUTs

DSP

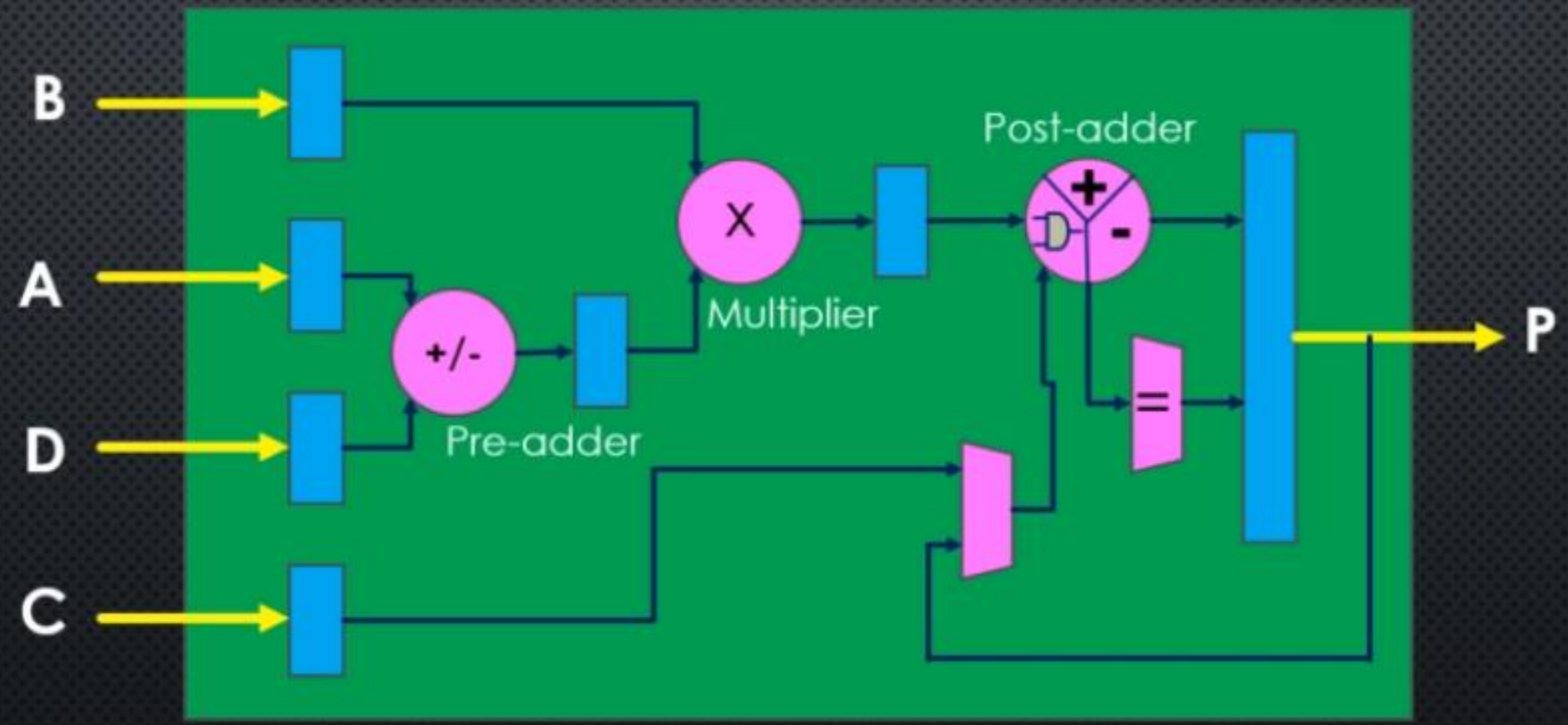


# DSP

**The DSP48 block is the most complex computational block available in a Xilinx FPGA.**

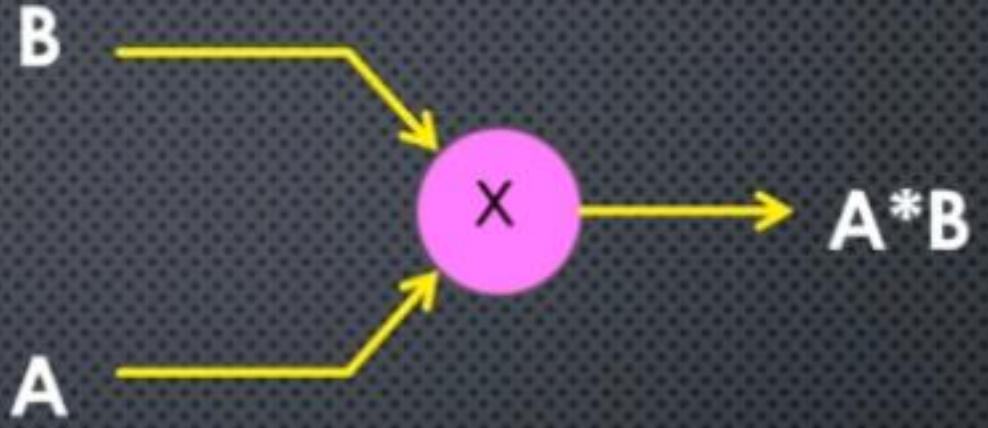


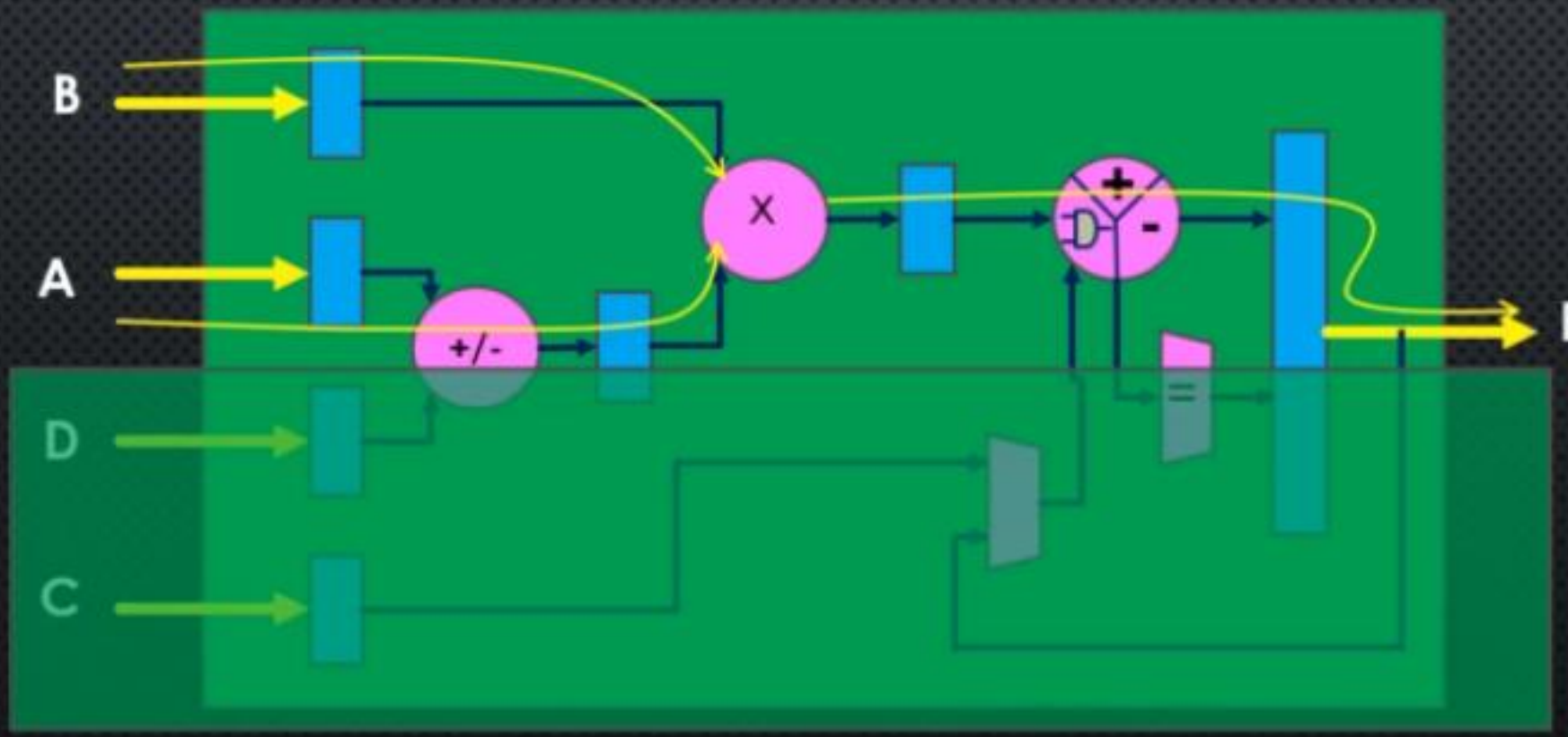
# DSP STRUCTURE



# DSP EXAMPLE

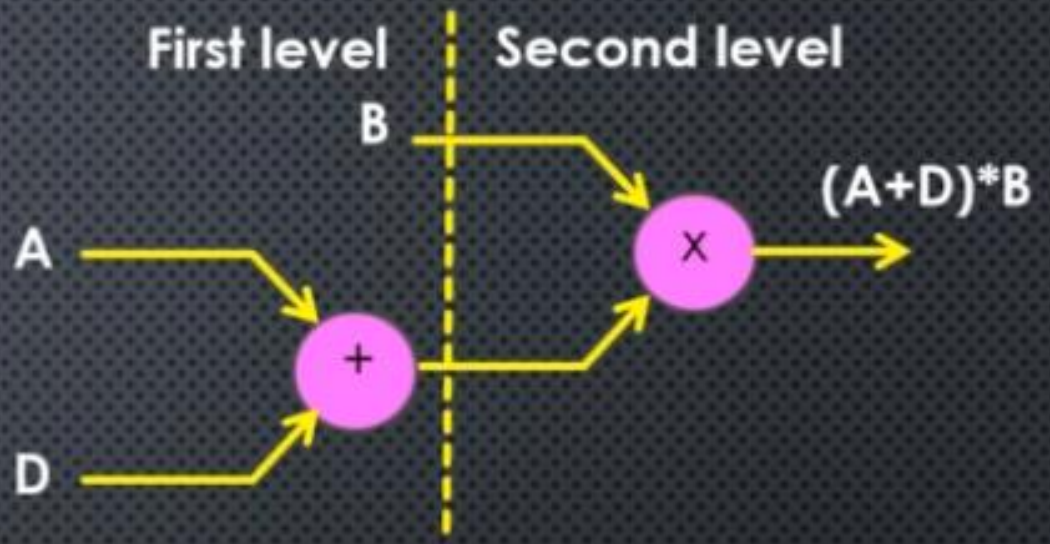
$f = A*B$

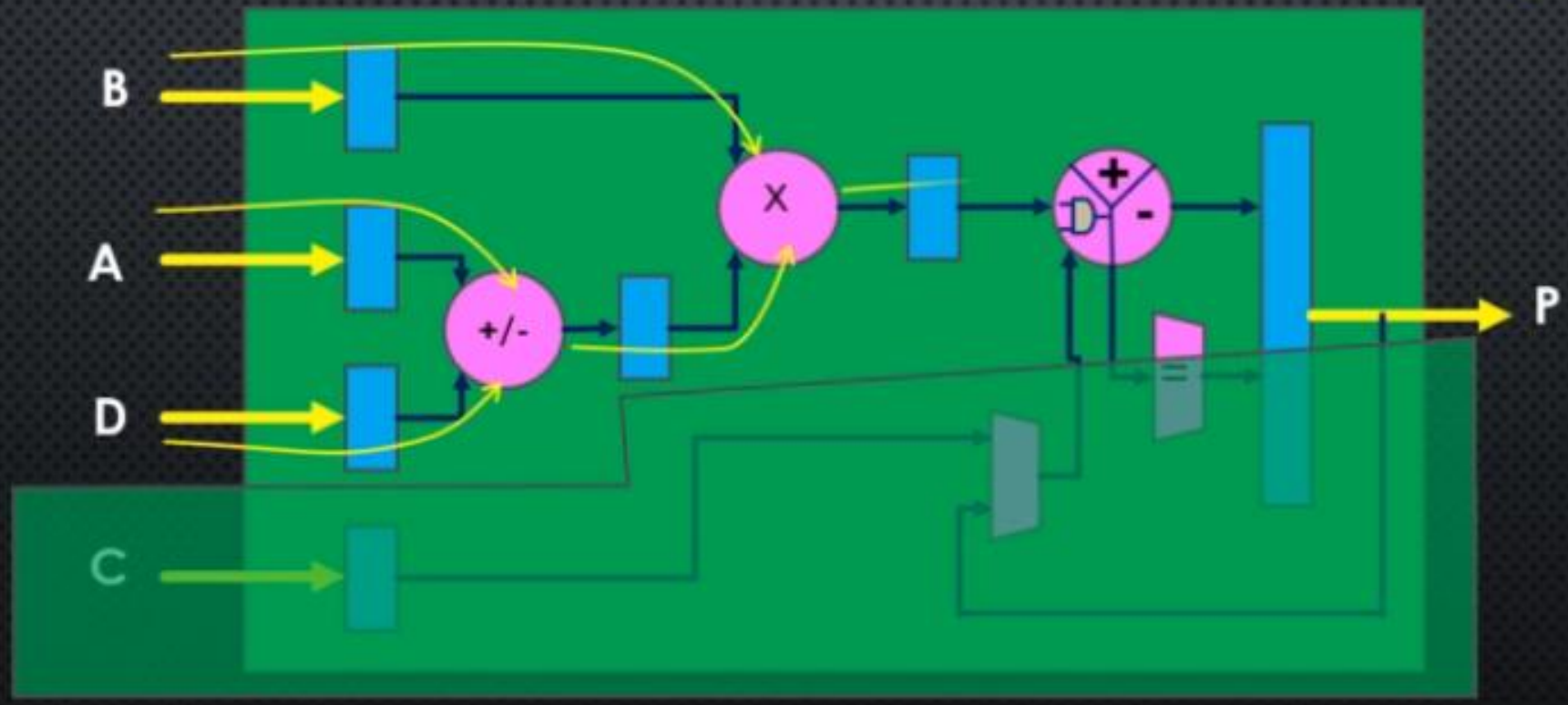


# DSP EXAMPLE

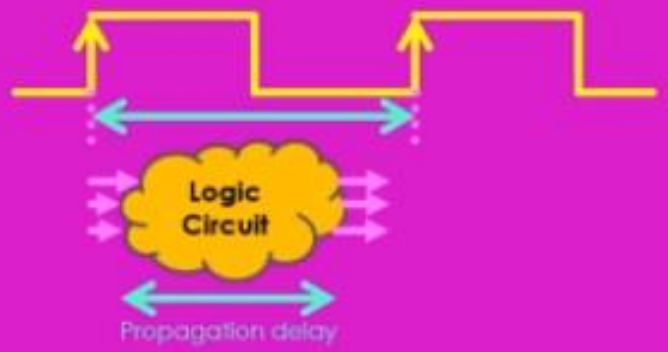
$f = (A+D)*B$



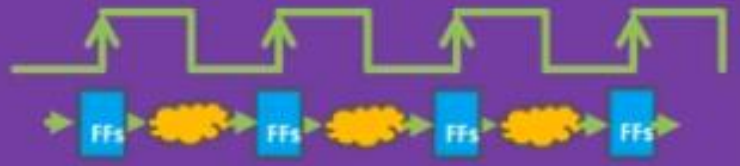



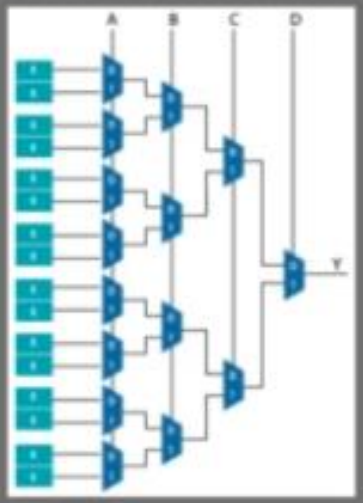
# LUT VS DSP

## Combinational

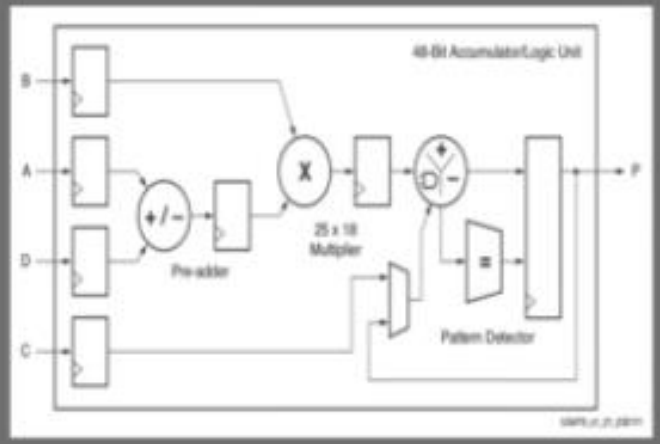



## Pipelined




LUT




DSP

# ARITHMETIC RESOURCES


add
mul
sdiv
srem
sub
udiv
urem
ashr
dadd
dmul
...

The operators represent operations in the C code such as additions, multiplications, array reads, and writes.

Cores are the specific hardware components used to create the design (such as adders, multipliers, pipelined multipliers, and block RAM).

AddSub
AddSubnS
AddSub_DSP
DivnS
DSP48
MulnS
Mul_LUT

# COMPILER DIRECTIVES

**ALLOCATION**

limit how many
- operators,
- cores
- or functions

are used in a design

**RESOURCE**

The RESOURCE directive is used to explicitly specify which core to use for specific operations.

# CORES

| | |
|---|---|
| AddSub | This core is used to implement both adders and subtractors. |
| AddSubnS | N-stage pipelined adder or subtractor. Vivado HLS determines how many pipeline stages are required. |
| AddSub_DSP | This core ensures that the add or sub operation is implemented using a DSP48 (Using the adder or subtractor inside the DSP48). |
| DivnS | N-stage pipelined divider. |
| DSP48 | Multiplications with bit-widths that allow implementation in a single DSP48. |

# CORES

| | |
|---|---|
| Mul | Combinational multiplier with bit-widths that exceed the size of a standard DSP48 macrocell.<br>Multipliers that can be implemented with a single DSP48 macrocell are mapped to the DSP48 core. |
| MulnS | N-stage pipelined multiplier with bit-widths that exceed the size of a standard DSP48 macrocell.<br>Multiplications which are >= 10 bits are implemented on a DSP48 macro cell. Multiplication lower than this limit are implemented using LUTs. Multipliers that can be implemented with a single DSP48 macrocell are mapped to the DSP48 core. |
| Mul_LUT | Multiplier implemented with LUTs.<br>Note: This only applies to C POD (plain old data) types. This cannot be used with Vivado HLS types (ap_int, ap_fixed, etc). |

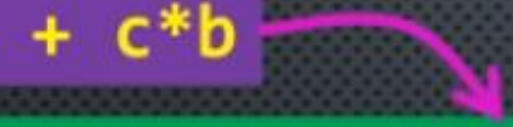
# RESOURCE PRAGMA


```
f = a + b
#pragma HLS RESOURCE variable=f core=AddSub_DSP
```

```
r = a * b
#pragma HLS RESOURCE variable=r core=Mul_LUT
```

# RESOURCE PRAGMA

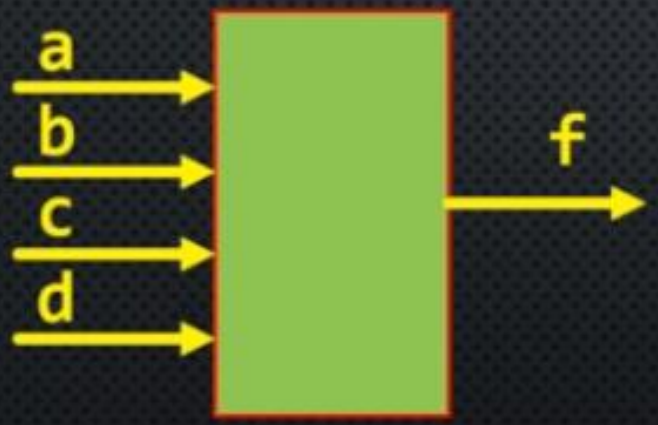
```
r = a + c*b
```

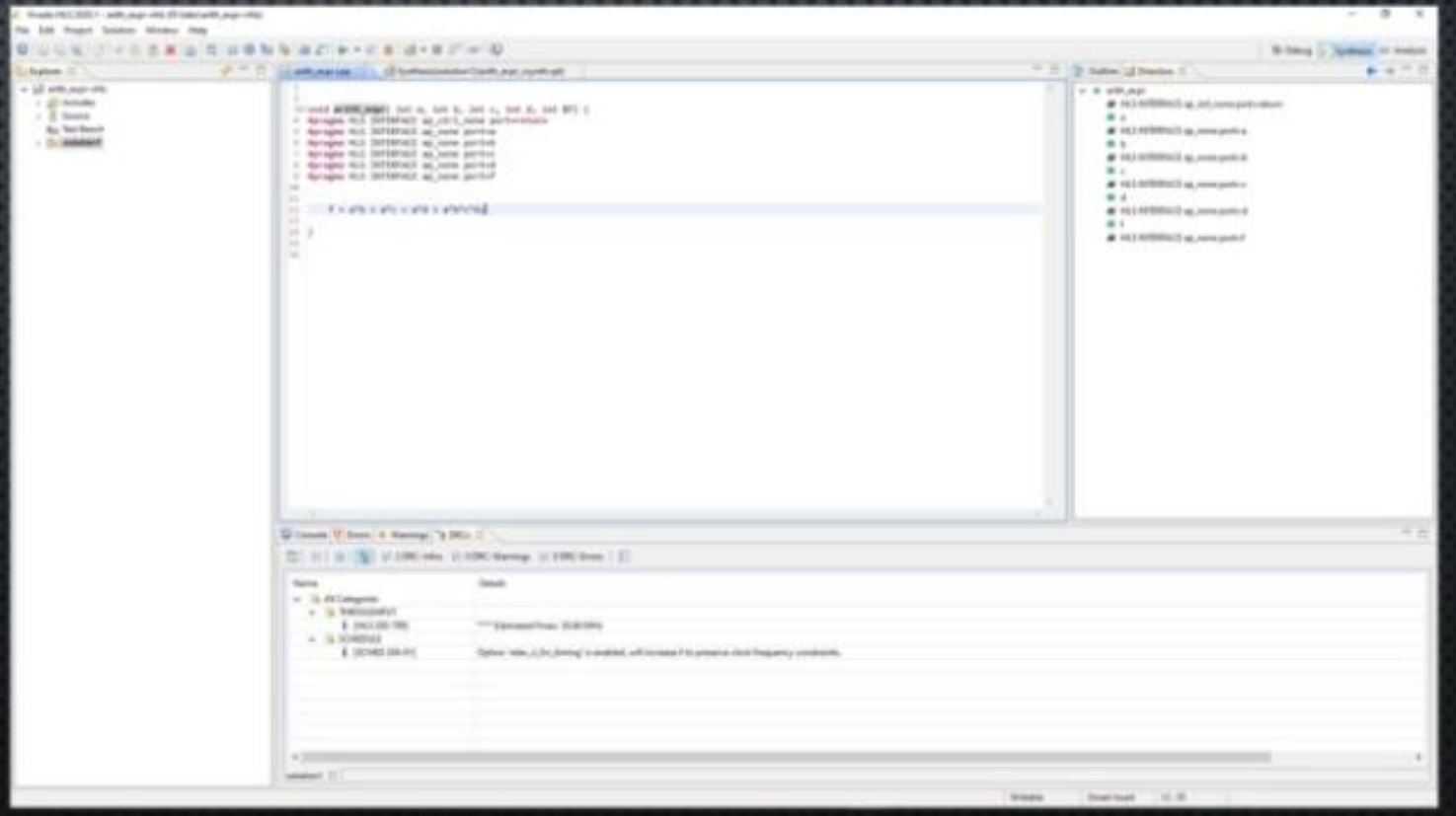


```
t = c*b
r = a + t;
#pragma HLS RESOURCE variable=t core=Mul_LUT
#pragma HLS RESOURCE variable=r core=AddSub_DSP
```

# VIVADO-HLS IDE

```
f = a*b + a*c + a*d + a*b*c*d;
```



# TAKEAWAY

By adding the RESOURCE directive, we can guide the HLS tool to select the desired hardware resource for implementing operators.

# DIVISION & REMAINDER

**Division (/)** ..... Returns the quotient of two integer values

**Modulus (%)** ..... Returns the modulus, or remainder of integer division, for two integer values.

# DIVIDE BY A CONSTANT

```
const int n = 128405;
r = a / n;
```

```
#pragma HLS RESOURCE variable=r core=DivnS
```

**Performance Estimates**

**Timing**

**Summary**

| Clock | Target | Estimated | Uncertainty |
|---|---|---|---|
| ap_clk | 0.20 us | 15.544 ns | 25.00 ns |

**Latency**

**Summary**

| Latency (cycles) | | Latency (absolute) | | Interval (cycles) | | |
|---|---|---|---|---|---|---|
| min | max | min | max | min | max | Type |
| 0 | 0 | 0 ns | 0 ns | 0 | 0 | none |

Detail

**Utilization Estimates**

**Summary**

| Name | BRAM_18K | DSP48E | FF | LUT | URAM |
|---|---|---|---|---|---|
| DSP | - | - | - | - | - |
| Expression | - | - | 4 | 0 | 197 | - |
| FIFO | - | - | - | - | - |
| Instance | - | - | - | - | - |
| Memory | - | - | - | - | - |
| Multiplexer | - | - | - | - | - |
| Register | - | - | - | - | - |
| Total | 0 | 0 | 4 | 0 | 197 | 0 |
| Available | 100 | 90 | 41600 | 20800 | 0 |
| Utilization (%) | 0 | 0 | 4 | 0 | ~0 | 0 |

Detail

**Interface**

**Summary**

| RTL Ports | Dir | Bits | Protocol | Source Object | C Type |
|---|---|---|---|---|---|
| a | in | 32 | ap_none | a | scalar |
| b | in | 32 | ap_none | b | scalar |
| r | out | 32 | ap_none | r | pointer |

Export the report(.html) using the Export Wizard

Open Analysis Perspective Analysis Perspective

**Performance Estimates**

**Timing**

**Summary**

| Clock | Target | Estimated | Uncertainty |
|---|---|---|---|
| ap_clk | 0.20 us | 4.157 ns | 25.00 ns |

**Latency**

**Summary**

| Latency (cycles) | | Latency (absolute) | | Interval (cycles) | | |
|---|---|---|---|---|---|---|
| min | max | min | max | min | max | Type |
| 35 | 35 | 7.000 us | 7.000 us | 35 | 35 | none |

Detail

**Utilization Estimates**

**Summary**

| Name | BRAM_18K | DSP48E | FF | LUT | URAM |
|---|---|---|---|---|---|
| DSP | - | - | - | - | - |
| Expression | - | - | - | - | - |
| FIFO | - | - | - | - | - |
| Instance | - | - | 2283 | 1738 | - |
| Memory | - | - | - | - | - |
| Multiplexer | - | - | - | 165 | - |
| Register | - | - | 36 | - | - |
| Total | 0 | 0 | 2319 | 1903 | 0 |
| Available | 100 | 90 | 41600 | 20800 | 0 |
| Utilization (%) | 0 | 0 | 5 | 9 | 0 |

**Interface**

**Summary**

| RTL Ports | Dir | Bits | Protocol | Source Object | C Type |
|---|---|---|---|---|---|
| ap_clk | in | 1 | ap_ctrl_none | integer_division | return value |
| ap_rst | in | 1 | ap_ctrl_none | integer_division | return value |
| a | in | 32 | ap_none | a | scalar |
| b | in | 32 | ap_none | b | scalar |
| r | out | 32 | ap_none | r | pointer |

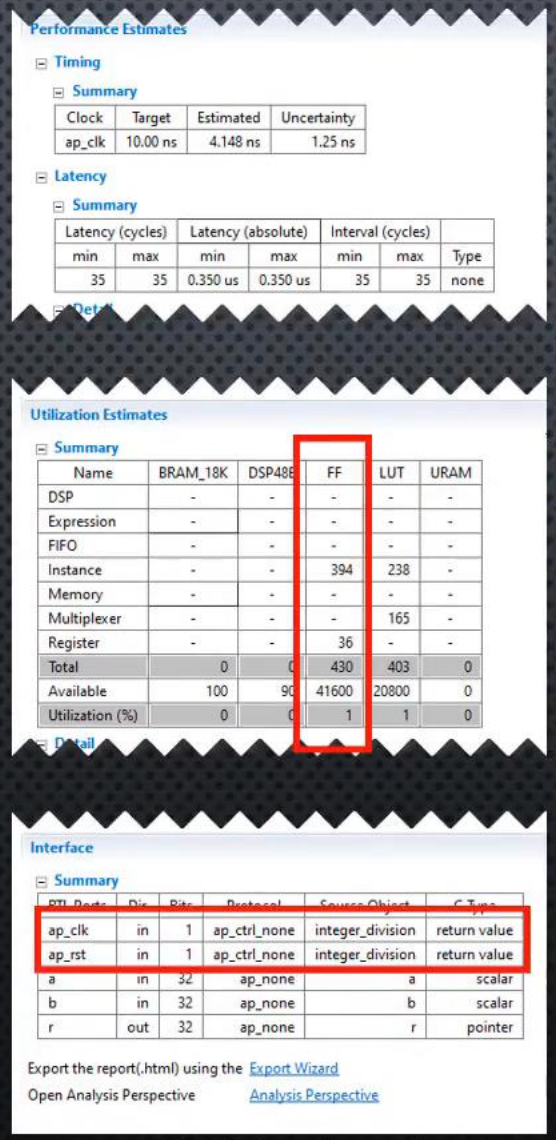
Export the report(.html) using the Export Wizard

Open Analysis Perspective Analysis Perspective

# DIVIDE BY A VARIABLE

```
r = a / b;
```

```
#pragma HLS RESOURCE variable=r core=DivnS
```



Performance Estimates

Timing

Summary

| Clock | Target | Estimated | Uncertainty |
|---|---|---|---|
| ap_clk | 10.00 ns | 4.148 ns | 1.25 ns |

Latency

Summary

| Latency (cycles) | | Latency (absolute) | | Interval (cycles) | | |
|---|---|---|---|---|---|---|
| min | max | min | max | min | max | Type |
| 35 | 35 | 0.350 us | 0.350 us | 35 | 35 | none |

Utilization Estimates

Summary

| Name | BRAM_18K | DSP48E | FF | LUT | URAM |
|---|---|---|---|---|---|
| DSP | - | - | - | - | - |
| Expression | - | - | - | - | - |
| FIFO | - | - | - | - | - |
| Instance | - | - | 394 | 238 | - |
| Memory | - | - | - | - | - |
| Multiplexer | - | - | - | 165 | - |
| Register | - | - | 36 | - | - |
| Total | 0 | 0 | 430 | 403 | 0 |
| Available | 100 | 90 | 41600 | 20800 | 0 |
| Utilization (%) | 0 | 0 | 1 | 1 | 0 |

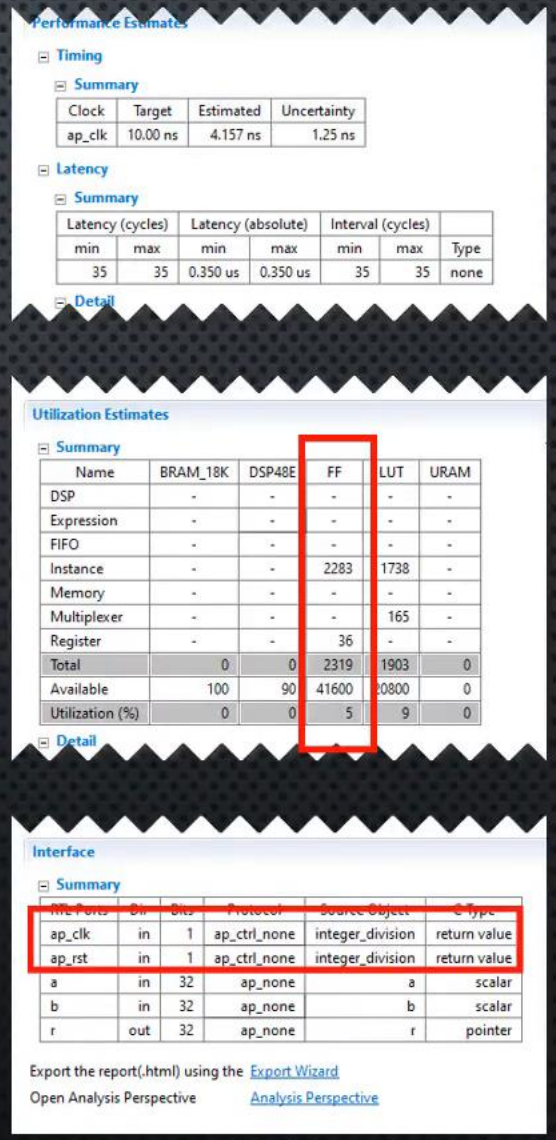
Interface

Summary

| RTL Ports | Dir | Bits | Protocol | Source Object | C Type |
|---|---|---|---|---|---|
| ap_clk | in | 1 | ap_ctrl_none | integer_division | return value |
| ap_rst | in | 1 | ap_ctrl_none | integer_division | return value |
| a | in | 32 | ap_none | a | scalar |
| b | in | 32 | ap_none | b | scalar |
| r | out | 32 | ap_none | r | pointer |

Export the report(.html) using the Export Wizard

Open Analysis Perspective Analysis Perspective



Performance Estimates

Timing

Summary

| Clock | Target | Estimated | Uncertainty |
|---|---|---|---|
| ap_clk | 10.00 ns | 4.157 ns | 1.25 ns |

Latency

Summary

| Latency (cycles) | | Latency (absolute) | | Interval (cycles) | | |
|---|---|---|---|---|---|---|
| min | max | min | max | min | max | Type |
| 35 | 35 | 0.350 us | 0.350 us | 35 | 35 | none |

Detail

Utilization Estimates

Summary

| Name | BRAM_18K | DSP48E | FF | LUT | URAM |
|---|---|---|---|---|---|
| DSP | - | - | - | - | - |
| Expression | - | - | - | - | - |
| FIFO | - | - | - | - | - |
| Instance | - | - | 2283 | 1738 | - |
| Memory | - | - | - | - | - |
| Multiplexer | - | - | - | 165 | - |
| Register | - | - | 36 | - | - |
| Total | 0 | 0 | 2319 | 1903 | 0 |
| Available | 100 | 90 | 41600 | 20800 | 0 |
| Utilization (%) | 0 | 0 | 5 | 9 | 0 |

Detail

Interface

Summary

| RTL Ports | Dir | Bits | Protocol | Source Object | C Type |
|---|---|---|---|---|---|
| ap_clk | in | 1 | ap_ctrl_none | integer_division | return value |
| ap_rst | in | 1 | ap_ctrl_none | integer_division | return value |
| a | in | 32 | ap_none | a | scalar |
| b | in | 32 | ap_none | b | scalar |
| r | out | 32 | ap_none | r | pointer |

Export the report(.html) using the Export Wizard

Open Analysis Perspective Analysis Perspective

# REMAINDER SYNTHESIS

```
const int n = 128405;
r = a % n;
```

---

```
r = a % b;
```

```
#pragma HLS RESOURCE variable=r core=DivnS
```

**Performance Estimates**

**Timing**

**Summary**

| Clock | Target | Estimated | Uncertainty |
|---|---|---|---|
| ap_clk | 10.00 ns | 4.148 ns | 1.25 ns |

**Latency**

**Summary**

| Latency (cycles) | | Latency (absolute) | | Interval (cycles) | | |
|---|---|---|---|---|---|---|
| min | max | min | max | min | max | Type |
| 35 | 35 | 0.350 us | 0.350 us | 35 | 35 | none |

**Utilization Estimates**

**Summary**

| Name | BRAM_18K | DSP48E | FF | LUT | URAM |
|---|---|---|---|---|---|
| DSP | - | - | - | - | - |
| Expression | - | - | - | - | - |
| FIFO | - | - | - | - | - |
| Instance | - | - | 394 | 238 | - |
| Memory | - | - | - | - | - |
| Multiplexer | - | - | - | 165 | - |
| Register | - | - | 36 | - | - |
| Total | 0 | 0 | 430 | 403 | 0 |
| Available | 100 | 90 | 41600 | 20800 | 0 |
| Utilization (%) | 0 | 0 | 1 | 1 | 0 |

**Detail**

**Interface**

**Summary**

| RTL Ports | Dir | Bits | Protocol | Source Object | C Type |
|---|---|---|---|---|---|
| ap_clk | in | 1 | ap_ctrl_none | integer_division | return value |
| ap_rst | in | 1 | ap_ctrl_none | integer_division | return value |
| a | in | 32 | ap_none | a | scalar |
| b | in | 32 | ap_none | b | scalar |
| r | out | 32 | ap_none | r | pointer |

Export the report(.html) using the Export Wizard

Open Analysis Perspective Analysis Perspective

**Performance Estimates**

**Timing**

**Summary**

| Clock | Target | Estimated | Uncertainty |
|---|---|---|---|
| ap_clk | 10.00 ns | 4.157 ns | 1.25 ns |

**Latency**

**Summary**

| Latency (cycles) | | Latency (absolute) | | Interval (cycles) | | |
|---|---|---|---|---|---|---|
| min | max | min | max | min | max | Type |
| 35 | 35 | 0.350 us | 0.350 us | 35 | 35 | none |

**Utilization Estimates**

**Summary**

| Name | BRAM_18K | DSP48E | FF | LUT | URAM |
|---|---|---|---|---|---|
| DSP | - | - | - | - | - |
| Expression | - | - | - | - | - |
| FIFO | - | - | - | - | - |
| Instance | - | - | 2283 | 1738 | - |
| Memory | - | - | - | - | - |
| Multiplexer | - | - | - | 165 | - |
| Register | - | - | 36 | - | - |
| Total | 0 | 0 | 2319 | 1903 | 0 |
| Available | 100 | 90 | 41600 | 20800 | 0 |
| Utilization (%) | 0 | 0 | 5 | 9 | 0 |

**Detail**

**Interface**

**Summary**

| RTL Ports | Dir | Bits | Protocol | Source Object | C Type |
|---|---|---|---|---|---|
| ap_clk | in | 1 | ap_ctrl_none | integer_division | return value |
| ap_rst | in | 1 | ap_ctrl_none | integer_division | return value |
| a | in | 32 | ap_none | a | scalar |
| b | in | 32 | ap_none | b | scalar |
| r | out | 32 | ap_none | r | pointer |

Export the report(.html) using the Export Wizard

Open Analysis Perspective Analysis Perspective

# REMAINDER COMBINATIONAL

```
const int n = 128405;
r = a % n;
```

```
r = a - n*(a / n);
```

# Any Question...

# Thank you