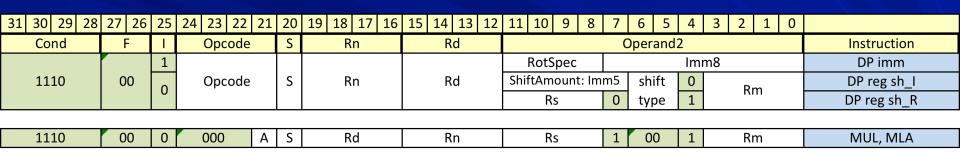
# Lab Assignment 10

## Multiply instructions

- MUL, MLA
- SMULL, SMLAL, UMULL, UMLAL



Rd <= Rm \* Rs Rd <= Rm \* Rs + Rn

1110	00	0	01	Ū	Α	S	RdHi	RdLo	Rs	1	00	1	Rm	U S MULL, MLAL

RdHi, RdLo <= Rm \* Rs RdHi, RdLo <= Rm \* Rs + RdHi, RdLo

# 4 x 4 multiplication

 $x b_3$ 

```
a_3 a_2 a_1 a_0 \times b_0
a_3 a_2 a_1 a_0 \times b_1
a_3 a_2 a_1 a_0 \times b_2
```

 $a_3 a_2 a_1 a_0$ 

### 4x4 unsigned

# 4x4 signed

$0 \ 0 \ 0 \ a_3 \ a_2 \ a_1 \ a_0$	$x b_0$
$0 \ 0 \ a_3 \ a_2 \ a_1 \ a_0$	$xb_1$
$0 a_3 a_2 a_1 a_0$	xb <sub>2</sub>
$a_3 a_2 a_1 a_0$	$x b_3$

```
a_3 a_3 a_3 a_3 a_2 a_1 a_0 \times b_0
a_3 a_3 a_3 a_2 a_1 a_0 \times b_1
a_3 a_3 a_2 a_1 a_0 \times b_2
a_3 a_2 a_1 a_0 \times b_2
```

## 4x4 unsigned

#### 4x4 signed

0 0 0 
$$a_3 a_2 a_1 a_0 \times b_0$$
  
0 0  $a_3 a_2 a_1 a_0 \times b_1$   
0  $a_3 a_2 a_1 a_0 \times b_2$   
 $a_3 a_2 a_1 a_0 \times b_3$ 

```
a_3 a_3 a_3 a_3 a_2 a_1 a_0 \times b_0
a_3 a_3 a_3 a_2 a_1 a_0 \times b_1
a_3 a_3 a_2 a_1 a_0 \times b_2
\overline{a_3} \overline{a_2} a_1 a_0 \times b_3
```

## Common part

```
0 0 0 a_3 a_2 a_1 a_0 \times b_0

0 0 a_3 a_2 a_1 a_0 \times b_1

0 a_3 a_2 a_1 a_0 \times b_2

a_3 a_2 a_1 a_0 \times b_3
```

```
a_3 a_3 a_3 a_3 a_2 a_1 a_0 x b_0
a_3 a_3 a_3 a_2 a_1 a_0 x b_1
a_3 a_3 a_2 a_1 a_0 x b_2
\overline{a_3} \overline{a_2} a_1 a_0 x b_3
```

# Signed, unsigned multiplication

#### Signed multiplication:

```
signal a_s, b_s : signed (31 downto 0);
signal p_s : signed (63 downto 0);
p_s <= a_s * b_s;
```

#### **Unsigned multiplication:**

```
signal a_u, b_u : unsigned (31 downto 0);
signal p_u : unsigned (63 downto 0);
p_u <= a_u * b_u;
```

# Synthesizing multipliers

#### Signed multiplication:

```
signal a_s, b_s: signed (31 downto 0);
  signal p_s: signed (63 downto 0);
  p s <= a s * b s; -- uses 4 DSP48E1
Unsigned multiplication:
  signal a_u, b_u: unsigned (31 downto 0);
  signal p_u: unsigned (63 downto 0);
  p_u <= a_u * b_u; -- uses 4 DSP48E1
```

# Combining results

```
a s <= signed (operand1);
b s <= signed (operand2);
a u <= unsigned (operand1);
b u <= unsigned (operand2);
p s \le a s * b s;
p u <= a u * b u;
result <= std logic vector(p s) when instr = smull
         else std logic vector(p u);
    -- uses 7 DSP48E1, but fails during routing
```

### Another way

```
x1 <= operand1(31) when instr = smull else '0';
x2 <= operand2(31) when instr = smull else '0';
operand1 33 \le x1 \& operand1;
operand2 33 \le x2 \& operand2;
a s <= signed (operand1 33);
b s <= signed (operand2 33);
p s <= a s * b s;
result 66 <= std_logic_vector(p_s);
    -- uses 4 DSP48E1!
```