COL215 Digital Logic and System Design

Designing with VHDL

Sequential Multiplier Design

12 September 2017

How to design using VHDL?

- Think in terms of VHDL constructs
- Any syntactically valid VHDL description is fine
- The tools should make correct sense out of it
- VHDL is just another programming language

How to design using VHDL?

- Think in terms of circuit structure
- Describe it in VHDL
- Use well understood abstractions
- Do not lose sight of underlying hardware

Design Styles

- An ARCHITECTURE can have -
  - Component instances (structural)
  - Processes (procedural)
  - Concurrent assignments (data flow)

These can be mixed but not advisable

- Structural hierarchy at the top
- Behavioural descriptions at the bottom

Semantics

- The whole design is a collection of concurrent processes
- Processes communicate with each other through signals
- No internal signals within processes, though the synthesizer may create some
- Ensure that each signal is driven by only one process (for the present)

What does a process mean?

- Repeated computation of values and assignment to signals
- A process executes in
  - zero time
  - non-zero time (not studied yet)
- A zero time process may represent
  - Either a combinational circuit
  - Or a Sequential circuit

Signals and processes

- A signal is either an output of a gate or a flip-flop (or an array of these)
- Driven by a process of appropriate type (combinational or sequential)
  - no mix up
  - no latches created by synthesizer
- Each signal driven by a unique process
- A process may drive multiple signals

Process describing a comb. circuit

- Sensitive to all its inputs
- Assigns to all outputs under all conditions

so that no latch is created.

Process describing a sequential circuit

- Sensitive to a clock
- All assignments to outputs on clock edge
- Optionally, sensitive to additional signal(s)
- Use this only for initialization
- Initialization with declaration is good only for simulation
- Use DFF description as template

D Flip-flop with synch S/R

```
ARCHITECTURE synchronous OF DFFsr IS
BEGIN
  PROCESS (clk)
  BEGIN
    IF clk = '1' AND clk'EVENT THEN
      IF s = '1' THEN        q <= '1';
      ELSIF r = '1' THEN     q <= '0';
      ELSE                    q <= d;
      END IF;
    END IF;
  END PROCESS;
END ARCHITECTURE synchronous;
```

D Flip-flop with asynch S/R

```
ARCHITECTURE asynchronous OF DFFsr IS
BEGIN
  PROCESS (clk, s, r)
  BEGIN
    IF s = '1' THEN                              q <= '1' ;
    ELSIF r = '1' THEN                           q <= '0' ;
    ELSIF clk = '1' AND clk'EVENT THEN   q <= d;
    END IF;
  END PROCESS;
END ARCHITECTURE asynchronous;
```

Iteration in space / time

FOR i IN 0 TO 15 LOOP

   sequential statements

END LOOP

=> iteration in time

not studied yet

FOR i IN 0 TO 15 GENERATE

   concurrent statements

END GENERATE

=> iteration in space

Signal vs Variable

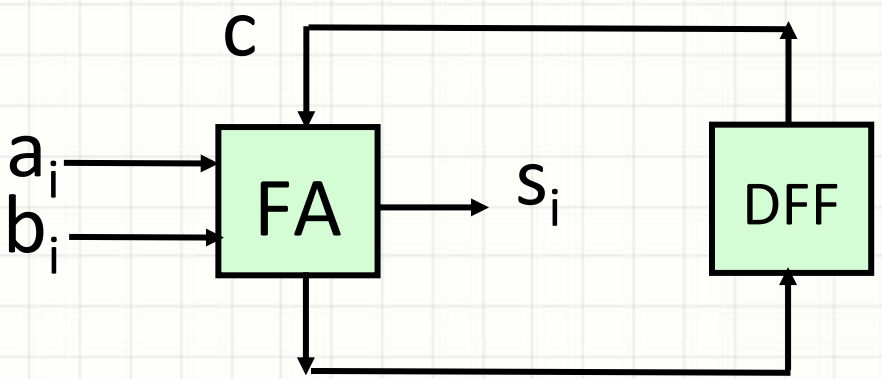
- Signals interconnect processes
- Variables are internal to processes
- A variable is updated instantly
- A signal is updated after a delay (the default is delta)

Serial arithmetic circuits

Parallel arithmetic circuits

Adders and Multipliers

- A serial adder can add two numbers of arbitrary size.



- Can a similar serial multiplier be designed?

Shift add multiplier (sequential)

$$A \times B = \sum_{i=0}^{n-1} A \cdot B_i \times 2^i$$

```
s = 0;
for i in 0 to n – 1 loop
    wait for clock edge;
    s = s + A · Bi x 2i ;
end loop;
```

```
s = 0;
for i in 0 to n - 1 loop
    wait for clock edge;
    s = s + A · Bi;
     A = shift_left (A);
end loop;
```

Shift-add multiply algorithm

shift s, rather than A

```
s = 0
for i in 0 to n-1 loop
    if (B0) then s = s+A
    end if
    A = 2 x A
    B = B / 2
end loop
```

```
s = 0
for i in 0 to n-1 loop
    if (B0)then sH=sH+A
    end if
    s = s / 2
    B = B / 2
end loop
```

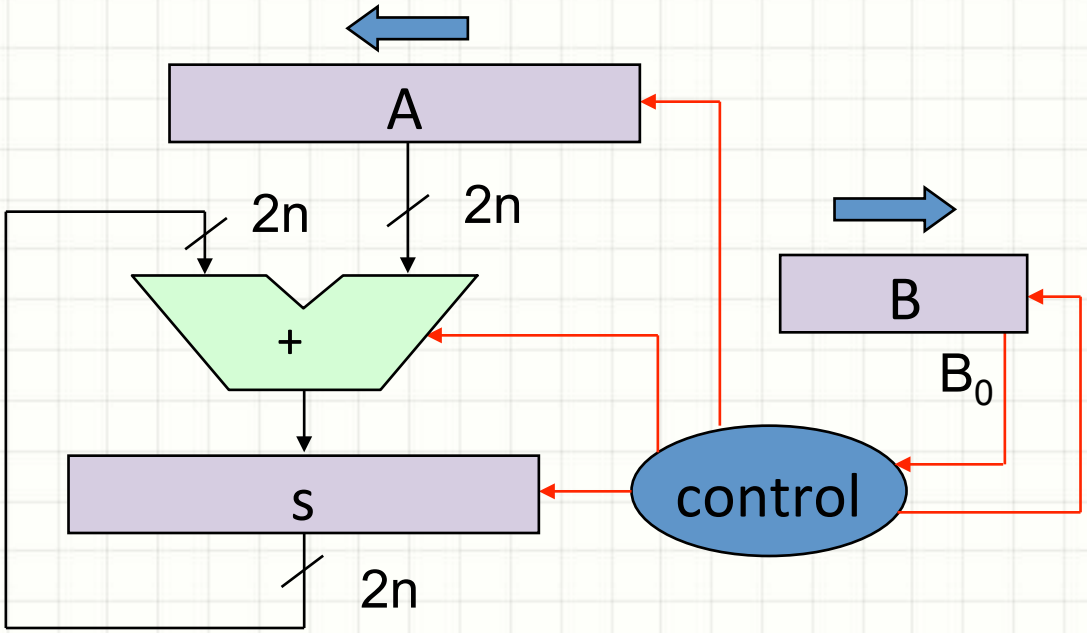
Shift-add multiply algorithm

place operand B in s

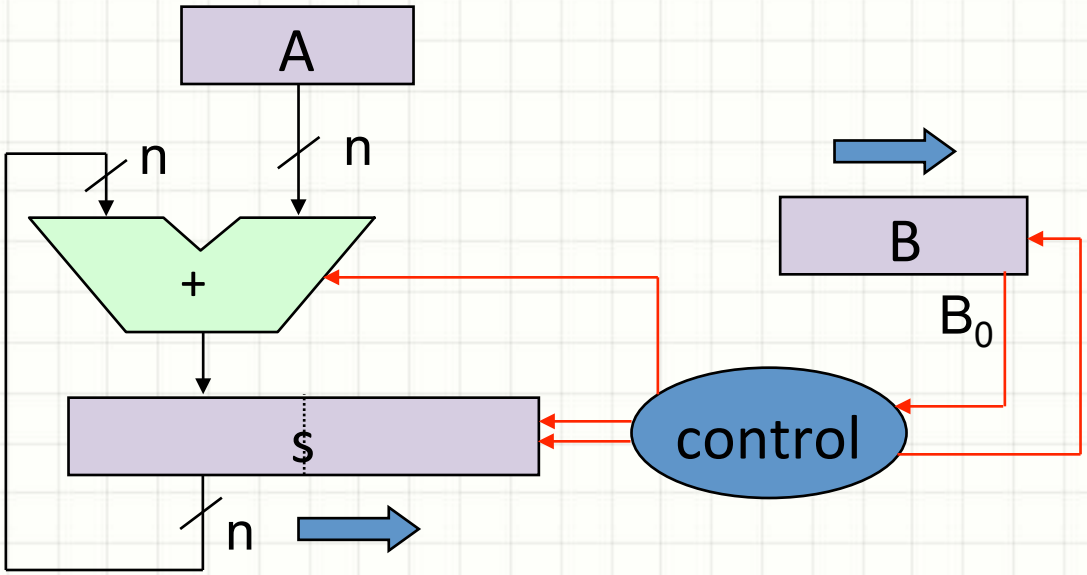
```
s = 0
for i in 0 to n-1 loop
    if (B0)then sH=sH+A
    end if
    s = s / 2
    B = B / 2
end loop
```

```
s = 0 | B
for i in 0 to n-1 loop
    if (s0) then sH=sH+A
    end if
    s = s / 2
end loop
```

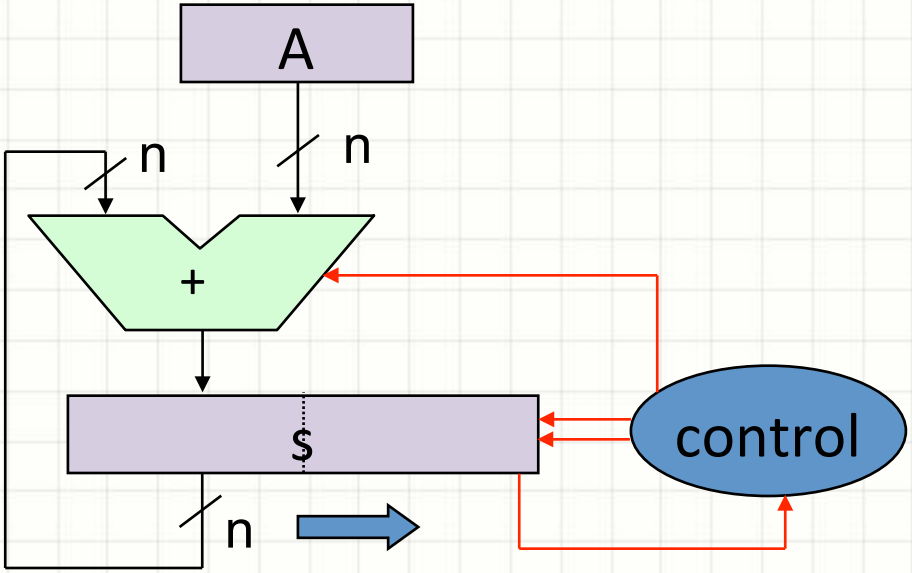
Sequential shift add multiplier 1



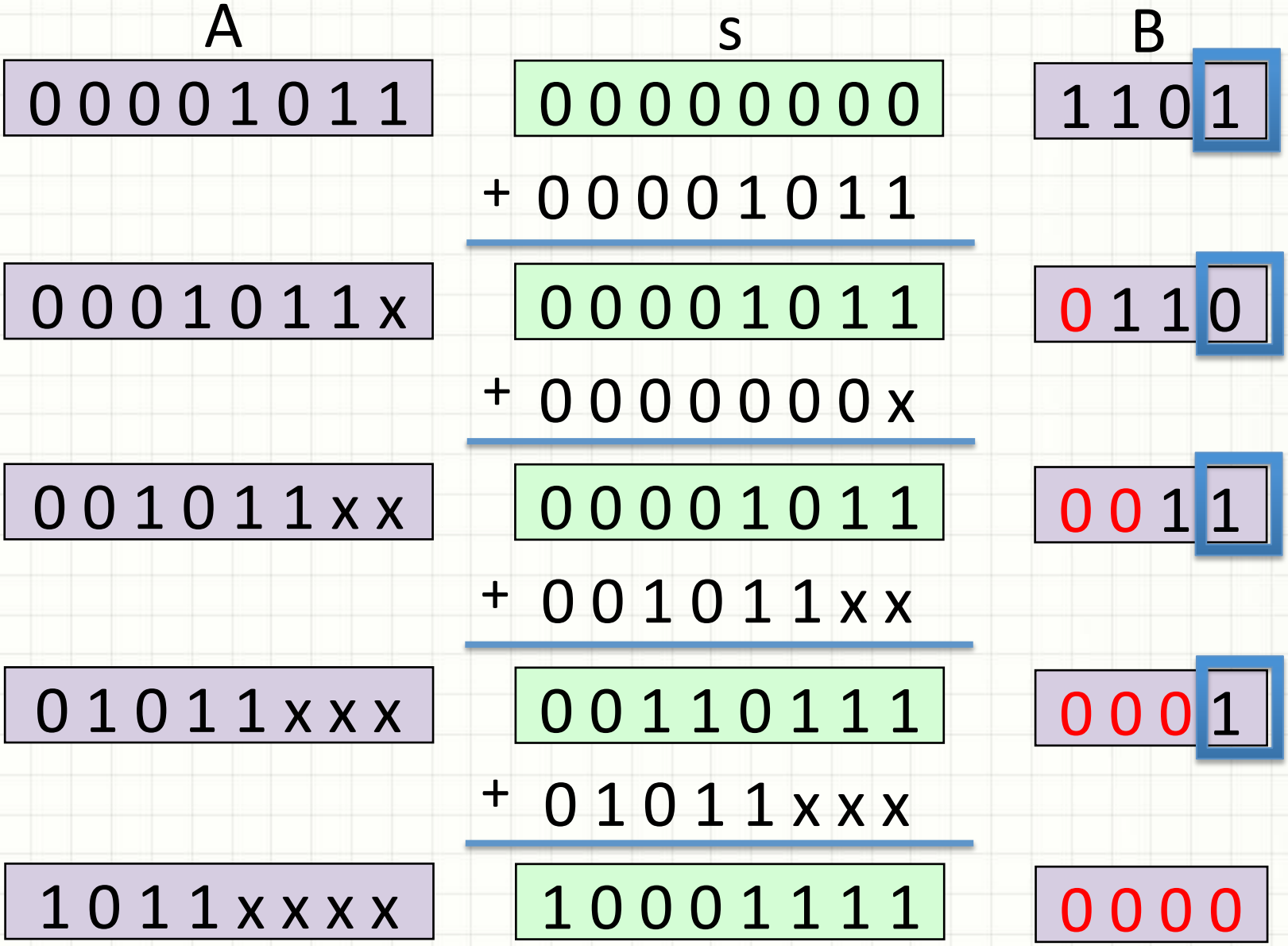
Sequential shift add multiplier 2



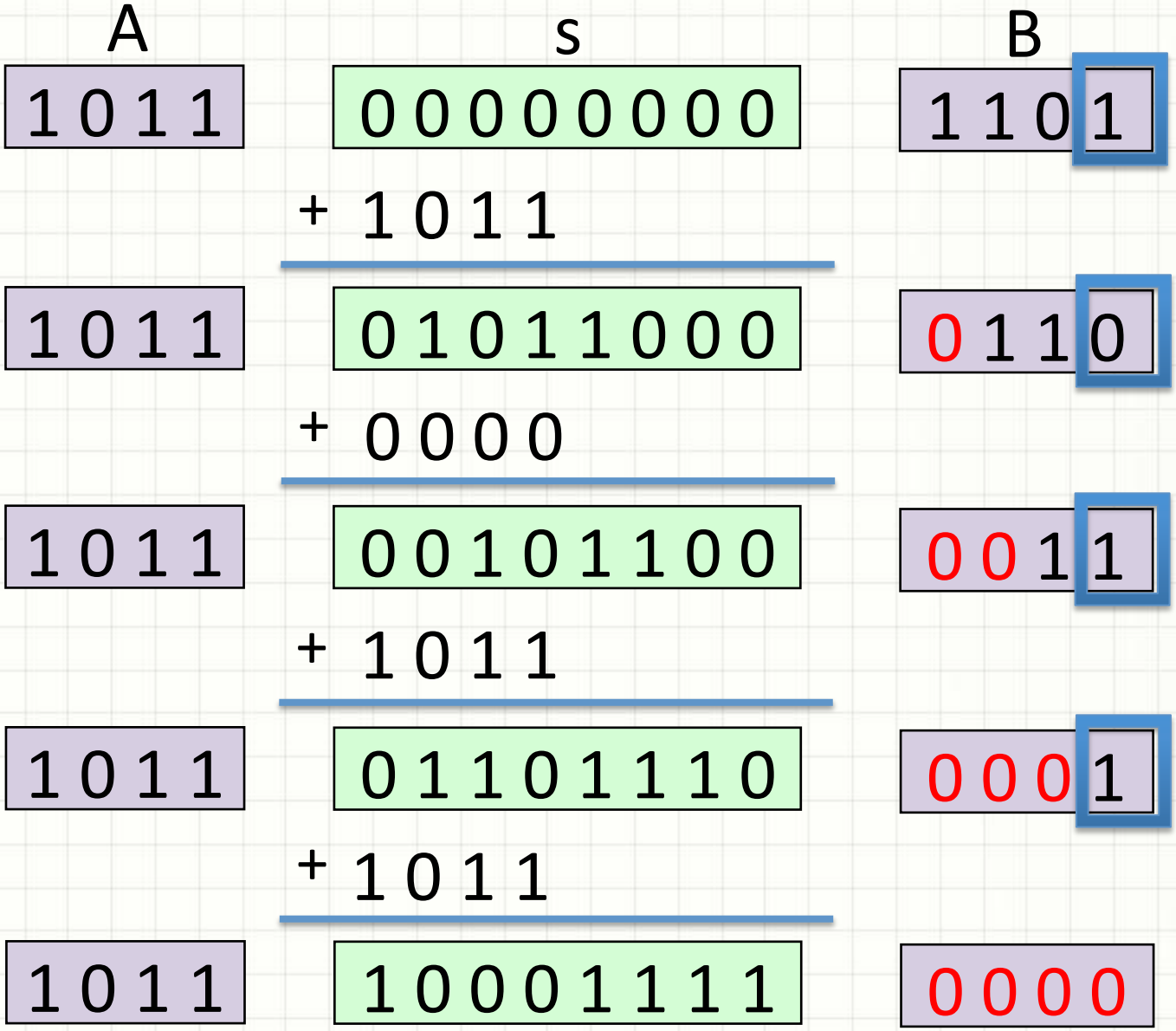
Sequential shift add multiplier 3



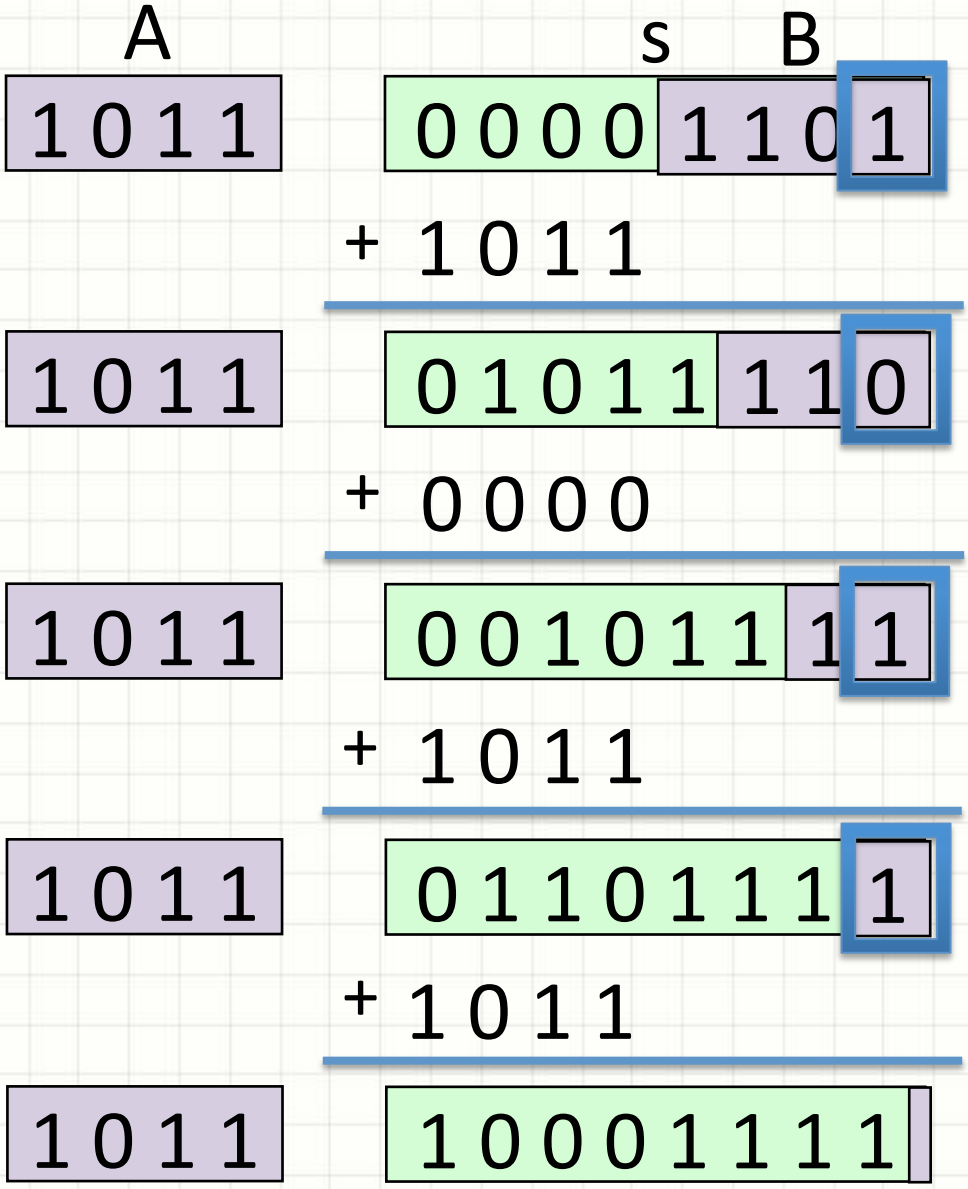
Shift-add multiply operation - 1



Shift-add multiply operation - 2



Shift-add multiply operation - 3



THANKS