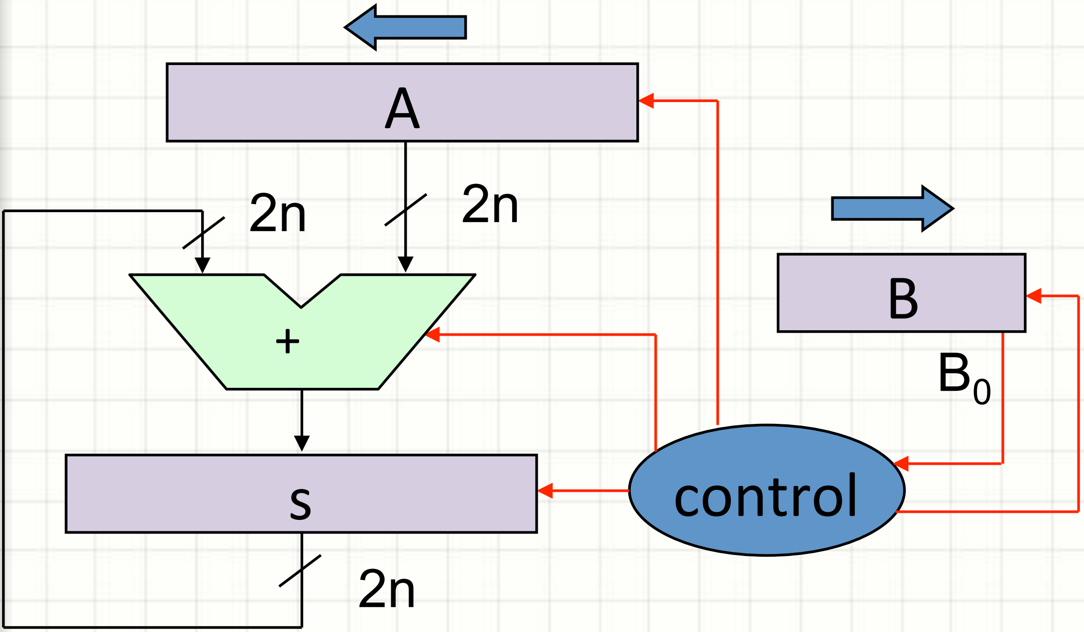
CSL215 Digital Logic and System Design

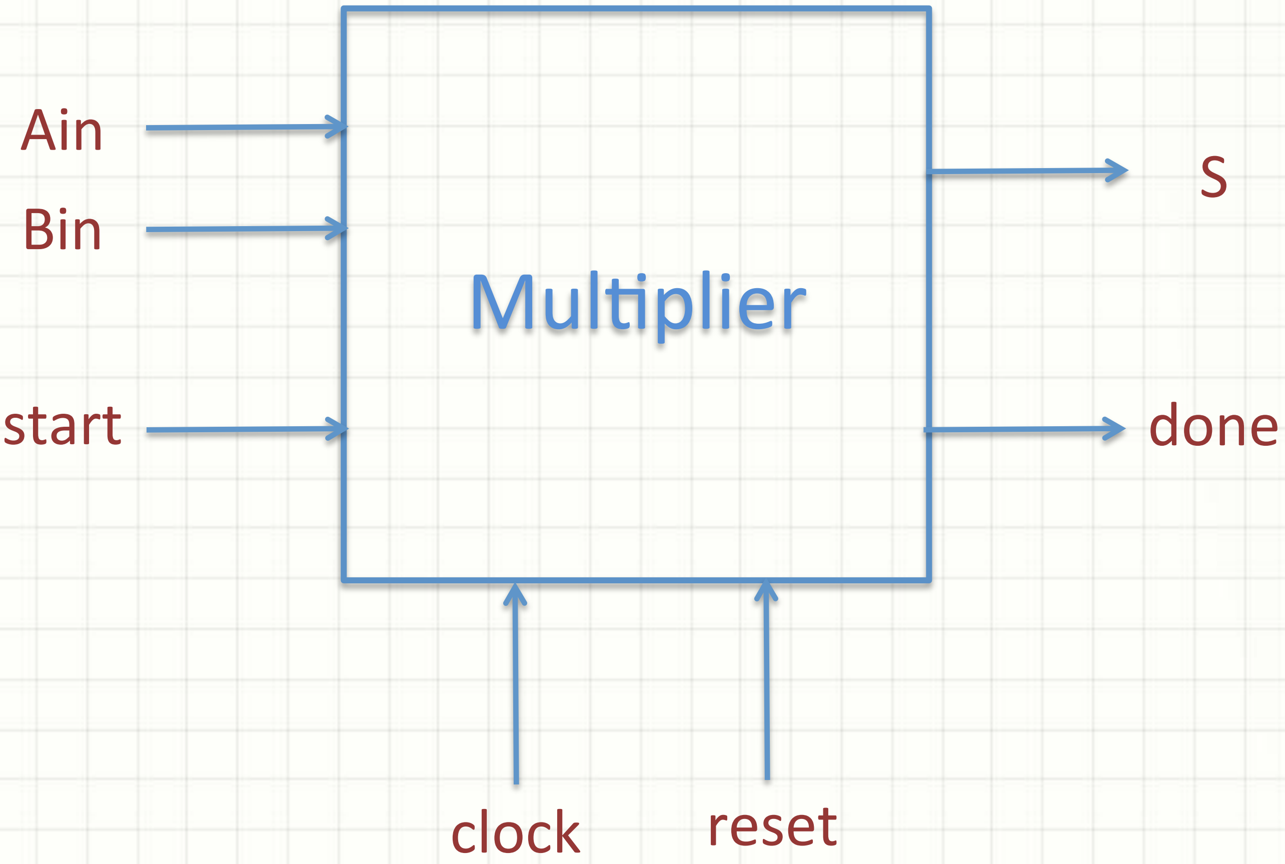
Sequential Multiplier Design

Sequential shift add multiplier

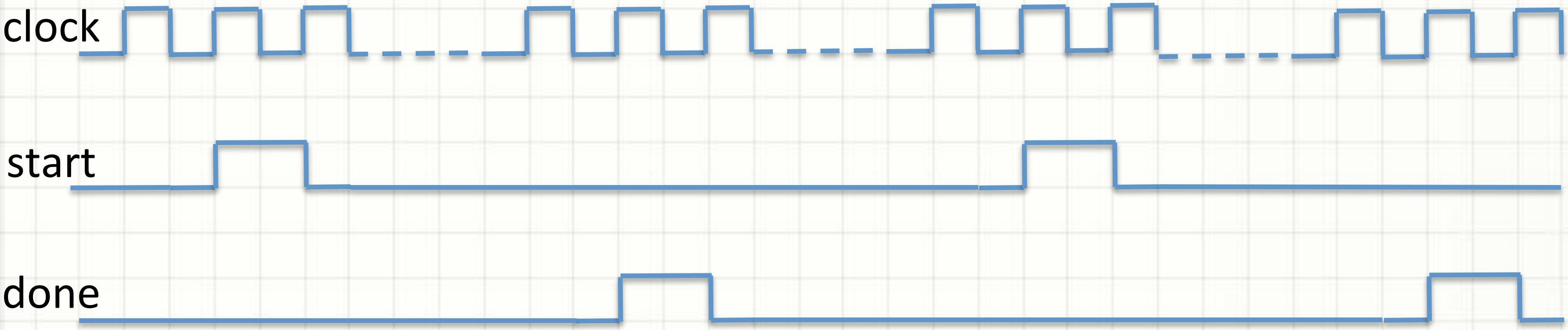


```
s = 0
for i in 0 to n-1 loop
    if ($B_0$) then s = s+A
    end if
    A = 2 x A
    B = B / 2
end loop
```

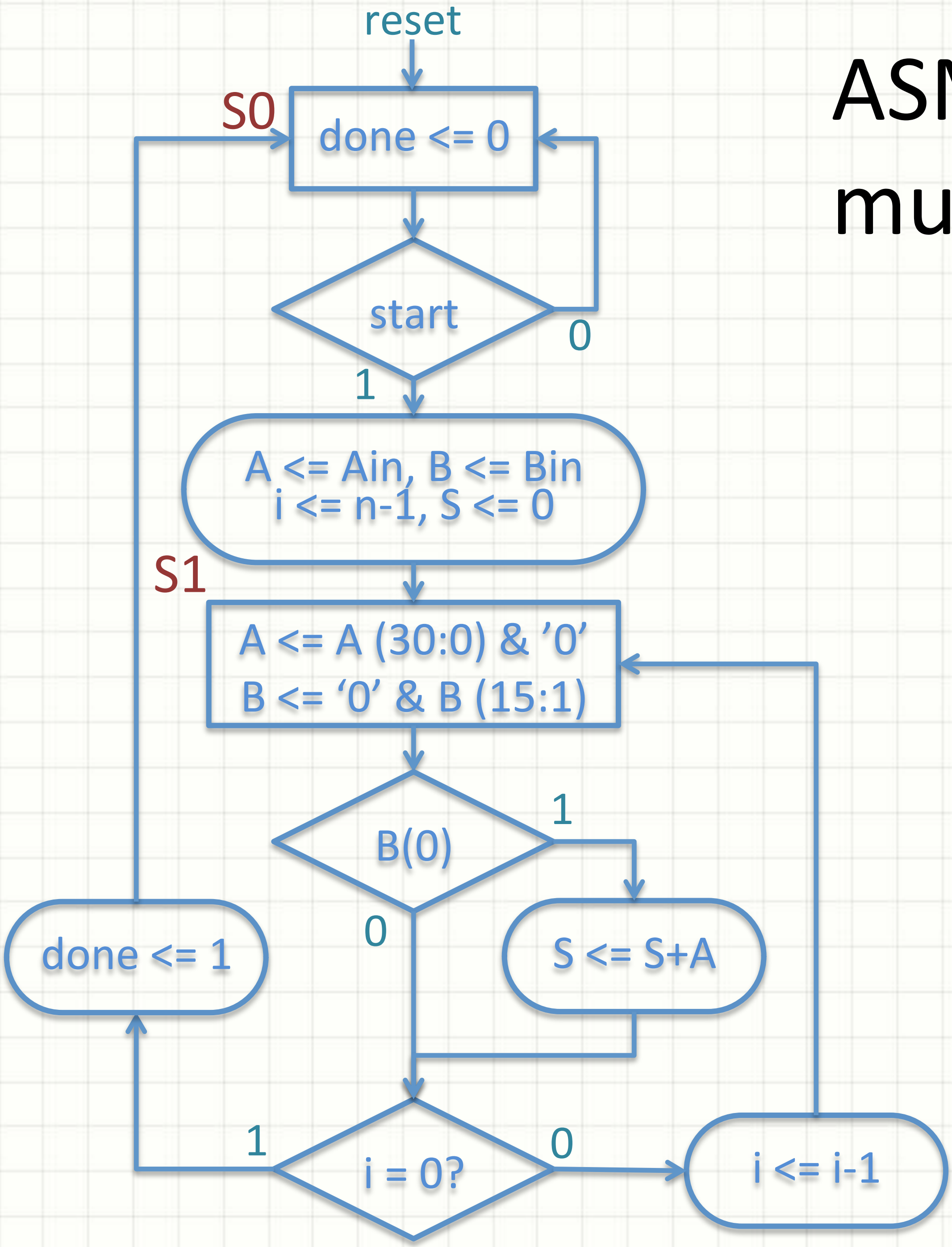
Interface



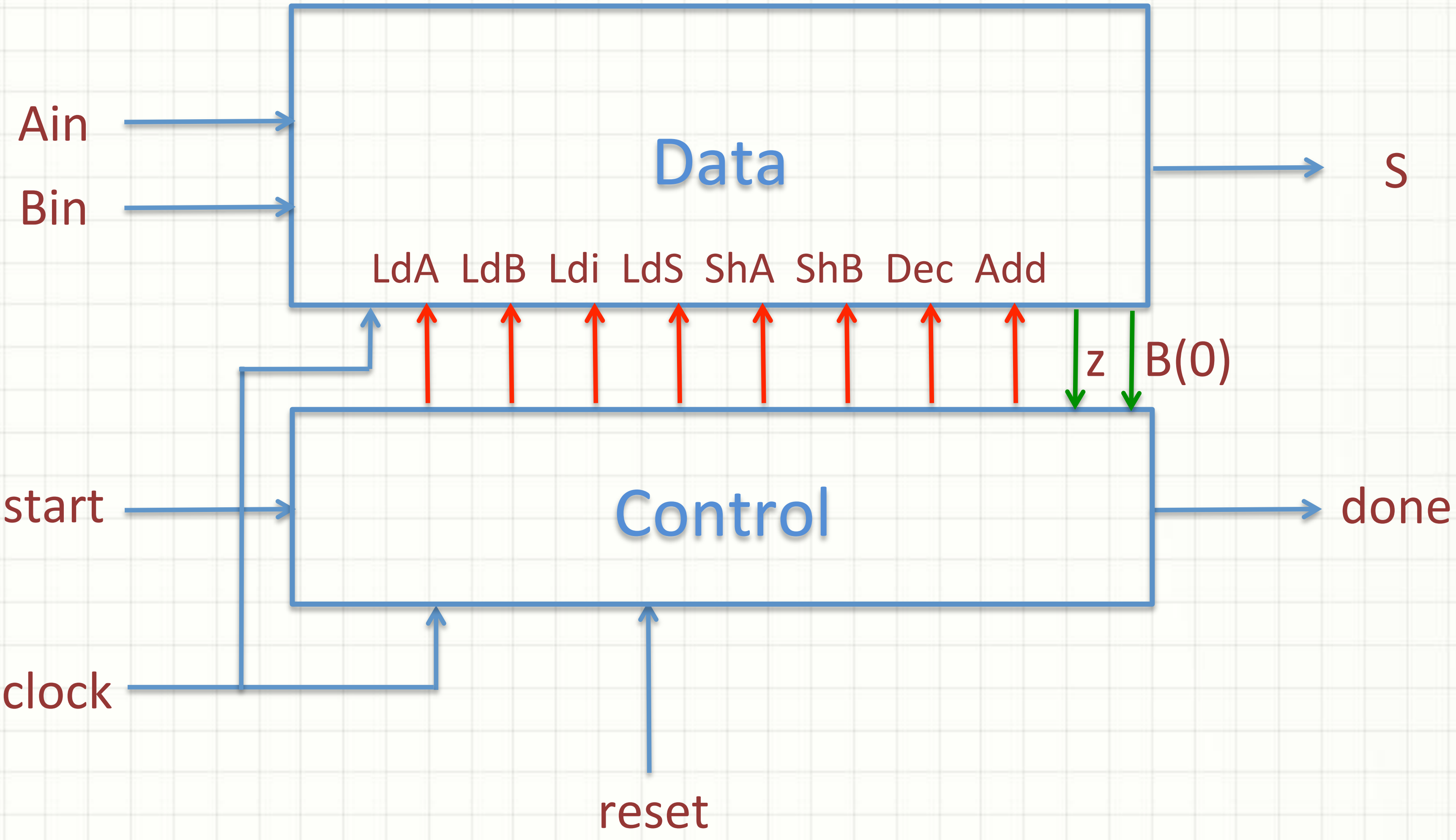
Synchronizing signals



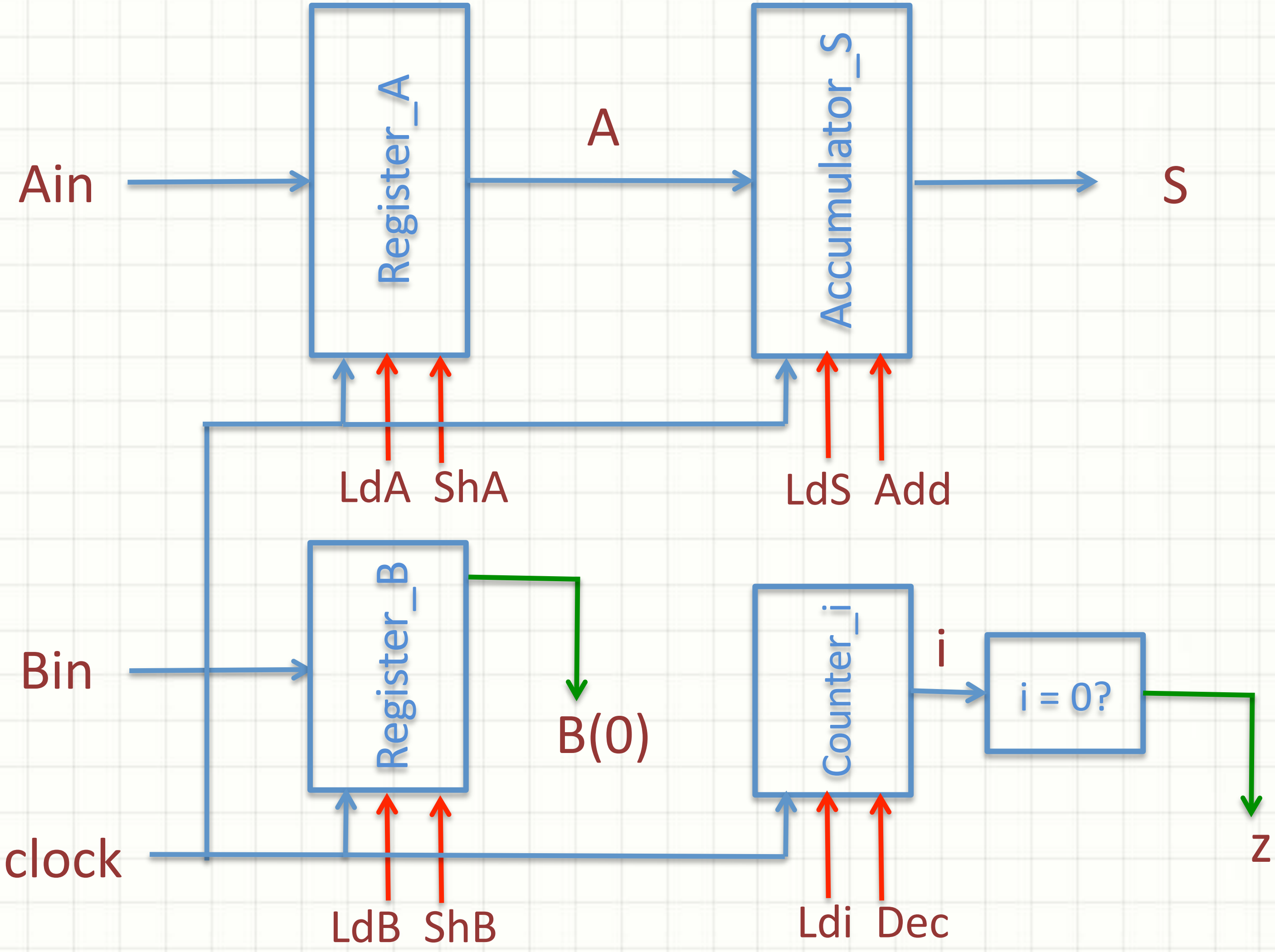
ASM chart for multiplier



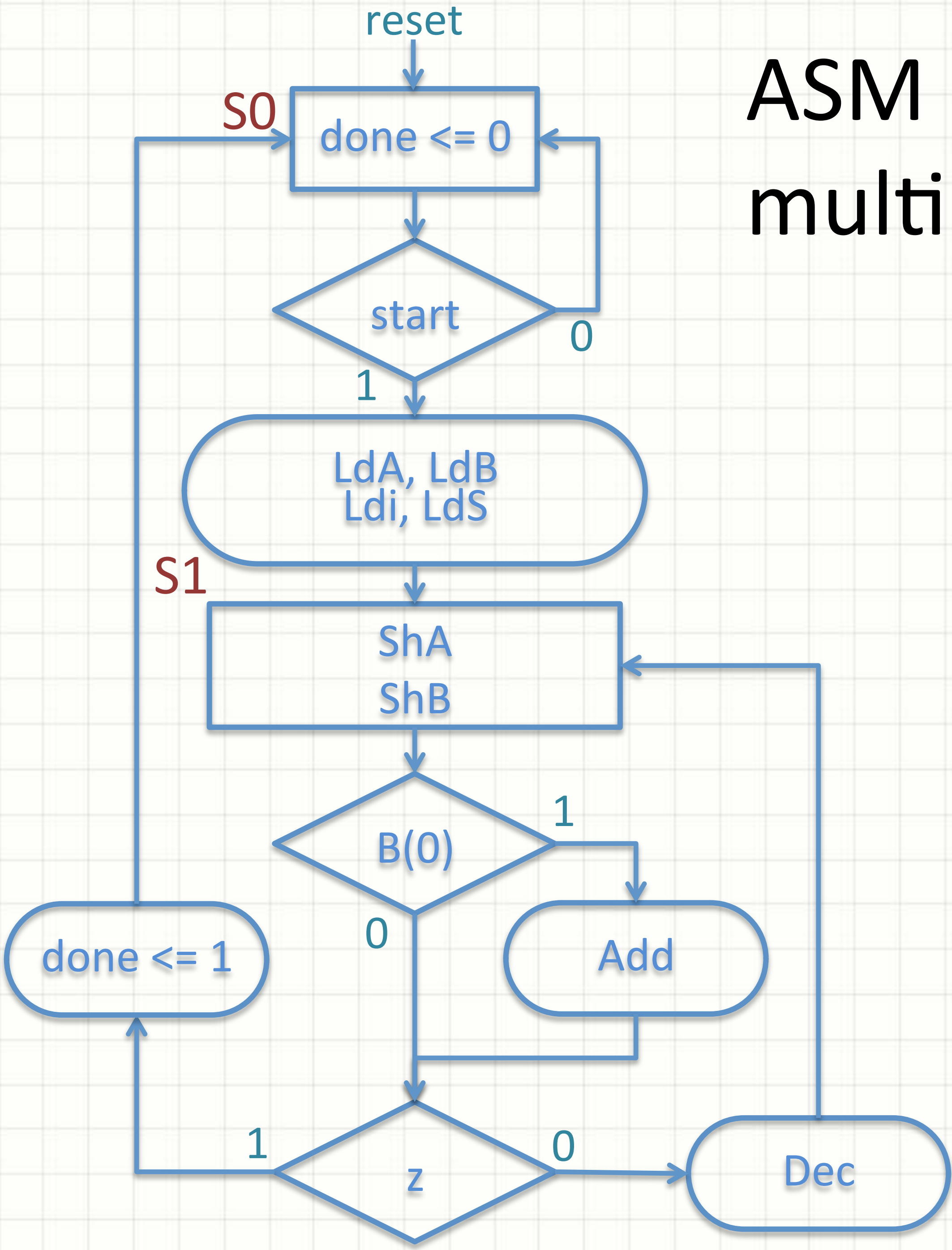
Control-data partition



Data path


Register_A
A
Accumulator_S
Ain
S
LdA ShA
LdS Add
Register_B
B(0)
Counter_i
i
i = 0?
Bin
z
clock
LdB ShB
Ldi Dec

ASM chart for multiplier controller



Multiplier VHDL description

```
mult_proc: process (reset, clock)
begin
    if reset = ‘1’ then state <= S0;
    elsif (clock’event and clock = ‘1’) then
        case state is
            when S0 =>
                . . . .
            when S1 =>
                . . . .
        end case;
    end if;
end process;
```

Multiplier VHDL description

```
when S0 =>
    done <= ‘0’;
    if start = ‘1’ then
        A <= zero16 & Ain;
        B <= Bin;
        i <= n-1;
        S <= zero32;
        state <= S0;
    end if;
```

Multiplier VHDL description

```
when S1 =>
    A <= A (30 downto 0) & ‘0’;
    B <= ‘0’ & B (15 downto 1);

    if B (0) = ‘1’ then
        S <= S + A;
    end if;

    if i = 0 then
        done <= 1;
        state <= S0;
    else
        i <= i - 1;
    end if;
```

Controller (state transitions) in VHDL

```
mult_control_trans_proc: process (reset, clock)
begin
    if reset = ‘1’ then state <= S0;
    elsif (clock’event and clock = ‘1’) then
        case state is
            when S0 =>
                if start = ‘1’ then state <= S1; end if;
            when S1 =>
               if z = ‘1’ then state <= S0; end if;
        end case;
    end if;
end process;
```

Controller (outputs) in VHDL

```
mult_control_out_proc: process (state, start, B, z)
begin
    LdA <= ‘0’; LdB <= ‘0’; Ldi <= ‘0’; LdS <= ‘0’;
    ShA <= ‘0’; ShB <= ‘0’; Add <= ‘0’; Dec <= ‘0’; done <= ‘0’;
    case state is
        when S0 =>
            if start = ‘1’ then LdA <= ‘1’; LdB <= ‘1’; Ldi <= ‘1’; LdS <= ‘1’;
            end if;
        when S1 =>
            ShA <= ‘1’; ShB <= ‘1’;
            if B (0) = ‘1’ then Add <= ‘1’; end if;
            if z = ‘1’ then done <= ‘1’; else Dec <= ‘1’; end if;
    end case;
end process;
```

Datapath in VHDL : Register A

```
Register_A: process (clock)
begin
   if (clock’event and clock = ‘1’) then
      if LdA = ‘1’ then A <= zero16 & Ain;
      elsif ShA = ‘1’ then A <= A (30 downto 0) & ‘0’;
      end if;
   end if;
end process;
```

Datapath in VHDL : Register B

```
Register_B: process (clock)
begin
    if (clock’event and clock = ‘1’) then
        if LdB = ‘1’ then B <= Bin;
        elsif ShB = ‘1’ then B <= ‘0’ & B (15 downto 1);
        end if;
    end if;
end process;
```

Datapath in VHDL : Accumulator S

```
Accumulator_S: process (clock)
begin
   if (clock’event and clock = ‘1’) then
      if LdS = ‘1’ then S <= zero32;
      elsif Add = ‘1’ then S <= S + A;
      end if;
   end if;
end process;
```

Datapath in VHDL : Counter i

```
Counter_i: process (clock)
begin
    if (clock’event and clock = ‘1’) then
        if Ldi = ‘1’ then i <= 0;
        elsif Dec = ‘1’ then i <= i - 1;
        end if;
    end if;
end process;
```

THANKS