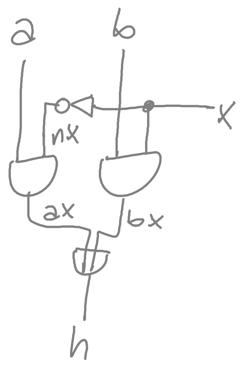
Embedded Systems Problem Set 5

Jan Hendrik Dittmar 2031259
Pascal Gwosdek 2505221

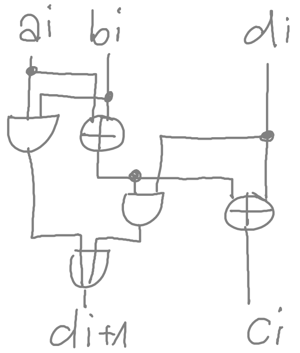
Problem 1:

```
entity multiplexer is
    port (x, a, b: in Bit; h: out Bit);
end multiplexer;


▶ architecture behavior of multiplexer is
  begin
      h <= (a and not(x)) or (b and x) after 10ns ;
  end


▶ architecture structure of multiplexer is
  signal      nx, ax, bx : Bit
  begin

          i: NOT port map (x, nx);
          j: AND port map (a, nx, ax);
          k: AND port map (b, x, bx);
          l: OR  port map (ax, bx, h);
  end     structure
```



Problem2

- 4 bits adder



```
▶ architecture behavior of adder is
  signals   d1, d2, d3 : Bit;
  begin
          c0  <=  a0 xor b0  after  10 ns ;
          d1  <=  a0 and b0  after  10 ns ;
          c1  <=  (a1 xor b1) xor d1 after 10ns ;
          d2  <=  (a1 and b1) or ((a1 xor b1) and d1) after 10ns ;
          c2  <=  (a2 xor b2) xor d2 after 10ns;
          d3  <=  (a2 and b2) or ((a2 xor b2) and d2 ) after 10ns;
          c3  <=  (a3 xor b3) xor d3 after 10ns;
          d   <=  (a3 and b3) or ((a3 xor b3) and d3) after 10ns;
  end

▶ architecture structure of adder is
  signals
  begin
          i:  half_adder port map (a0, b0, d1, c0);
          j:  full_adder port map (a1, b1, d1, d2, c1);
          k:  full_adder port map (a2, b2, d2, d3, c2);
          l:  full_adder port map (a3, b3, d3, d, c3);
  end  structure
```

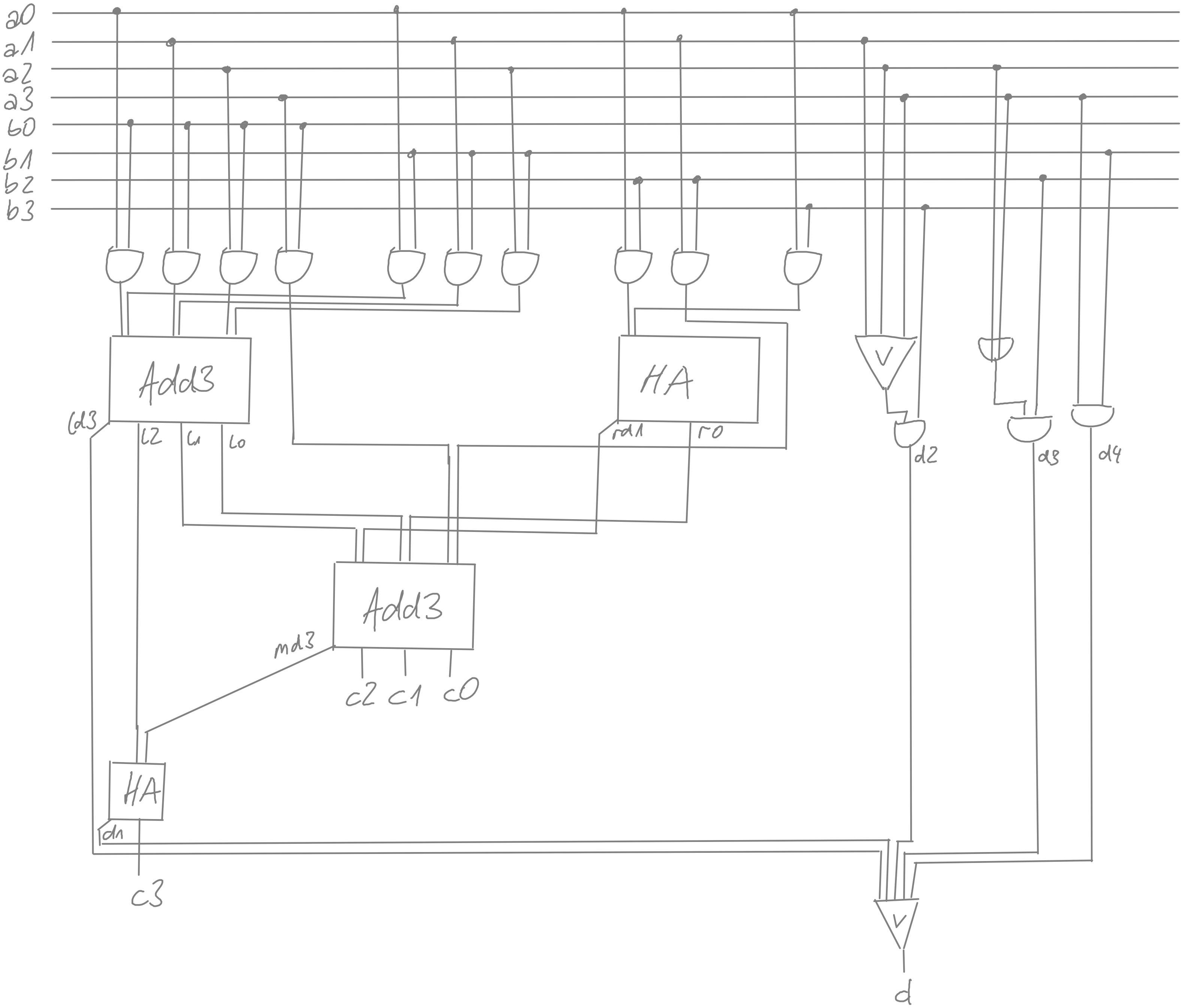
- saturating 4 bits adder

Idea:
Use FA and overwrite
with 1111 iff d=1.

```
▶ architecture behavior of adder is
  signals    o0, o1, o2, o3, d1, d2, d3;
  begin
          o0  ⇐  a0 xor b0  after  10ns;
          d1  ⇐  a0 and b0  after  10ns;
          o1  ⇐  (a1 xor b1) xor d1 after 10ns;
          d2  ⇐  (a1 and b1) or ((a1 xor b1) and d1) after 10ns;
          o2  ⇐  (a2 xor b2) xor d2 after 10ns;
          d3  ⇐  (a2 and b2) or ((a2 xor b2) and d2 ) after 10ns;
          o3  ⇐  (a3 xor b3) xor d3 after 10ns;
          d   ⇐  (a3 and b3) or ((a3 xor b3) and d3) after 10ns;
          c0  ⇐  d or o0 after 10ns;
          c1  ⇐  d or o1 after 10ns;
          c2  ⇐  d or o2 after 10ns;
          c3  ⇐  d or o3 after 10ns;
  end

▶ architecture structure of adder is
  signals    o0, o1, o2, o3, d1, d2, d3;
  begin
          i:  half_adder port map (a0, b0, d1, o0);
          j:  full_adder port map (a1, b1, d1, d2, o1);
          k:  full_adder port map (a2, b2, d2, d3, o2);
          l:  full_adder port map (a3, b3, d3, d, o3);
          m:  OR         port map (d, o0, c0);
          n:  OR         port map (d, o1, c1);
          o:  OR         port map (d, o2, c2);
```

```
p : OR        port map (d, o3, c3 );
end structure
```

Problem2



- 4 bits multiplier

```
▶ architecture behavior of multiplier is
  signals c0, c1, c2, cd1, cd2, cd3, r0, rd1, md1, md2, md3: Bit;
  begin
        c0  <= (a2 and b0) xor (a2 and b1) after 10 ns;
        cd1 <= (a2 and b0) and (a2 and b1) after 10ns;
        c1  <= ((a1 and b0) xor (a1 and b1)) xor cd1 after 10ns;
        cd2 <= ((a1 and b0) and (a1 and b1))
               or (((a1 and b0) xor (a1 and b1)) and cd1
               after 10ns;
        c2  <= ((a0 and b0) xor (a0 and b1)) xor cd2 after 10ns;
        cd3 <= ((a0 and b0) and (a0 and b1))
               or (((a0 and b0) xor (a0 and b1)) and cd2
               after 10ns;
        r0  <= (a0 and b2) xor (a0 and b3) after 10ns;
        rd1 <= (a0 and b2) and (a0 and b3) after 10ns;
        co  <= (a3 and b0) xor (a1 and b2) after 10ns;
        md1 <= (a3 and b0) and (a1 and b2) after 10ns;
        c1  <= (c0 xor r0) xor md1 after 10ns;
        md2 <= (c0 and r0) or ((c0 xor r0) and md1
               after 10ns;
        c2  <= ((c1 xor r1) xor md2 after 10ns;
        md3 <= (c1 and r1) or ((c1 xor r1) and md2
               after 10ns;
        c3  <= md3 xor c2 after 10ns;
        d   <= ((md3 and c2) or ((cd3 or (b3 and (a1 or
               a2) or a3))) or ((b2 and (a2 or a3)) or (b1 and a3))))
               after 10ns;
```

```
end
```

- In the following, we expect Add3 to be built like the adder of exercise 2, except for the signal c3. d is then given by d3.

```
architecture structure of multiplier is
signals n0, n1, n2, n3, n4, n5, n6, n7, n8, n9, C0, C1, C2, Cd3, r0, rd1, a23, a123,
        md3, d1, d2, d3, d4, d01, d23, d234: Bit;
begin
        i1:  AND         port map (a0, b0, n0);
        i2:  AND         port map (a1, b0, n1);
        i3:  AND         port map (a2, b0, n2);
        i4:  AND         port map (a3, b0, n3);
        i5:  AND         port map (a0, b1, n4);
        i6:  AND         port map (a1, b1, n5);
        i7:  AND         port map (a2, b1, n6);
        i8:  AND         port map (a0, b2, n7);
        i9:  AND         port map (a1, b2, n8);
        i10: AND         port map (a0, b3, n9);
        i11: half_adder  port map (n7, n9, rd1, r0);
        i12: Add3        port map (n0, n1, n2, n4, n5, n6, C2, C1, C0, Cd3);
        i13: Add3        port map (C1, C0, n3, rd1, r0, n8, c1, c2, c0, md3);
        i14: half_adder  port map (C2, md3, d1, c3);
        i15: OR          port map (a2, a3, a23);
        i16: OR          port map (a23, a1, a123);
        i17: AND         port map (b3, a123, d2);
        i18: AND         port map (b2, a23, d3);
        i19: AND         port map (a3, b1, d4);
        i20: OR      port map (cd3, d1, d01);
        i21: OR      port map (cd3, d1, d01);
        i22: OR      port map (d2, d3, d23);
        i23: OR      port map (d23, d4, d234);
        i24: OR      port map (d01, d234, d);
end
```

- saturating 4 bits multiplier

```
▶ architecture behavior of multiplier is
  signals   c0, c1, c2, cd1, cd2, cd3, r0, rd1, md1, md2, md3, o0, o1, o2,
            o3 : Bit;
  begin
            c0  <= (a2 and b0) xor (a2 and b1) after 10ns;
            cd1 <= (a2 and b0) and (a2 and b1) after 10ns;
            c1  <= ((a1 and b0) xor (a1 and b1)) xor cd1 after 10ns;
            cd2 <= ((a1 and b0) and (a1 and b1))
                   or (((a1 and b0) xor (a1 and b1)) and cd1
                   after 10ns;
            c2  <= ((a0 and b0) xor (a0 and b1)) xor cd2 after 10ns;
            cd3 <= ((a0 and b0) and (a0 and b1))
                   or (((a0 and b0) xor (a0 and b1)) and cd2
                   after 10ns;
            r0  <= (a0 and b2) xor (a0 and b3) after 10ns;
            rd1 <= (a0 and b2) and (a0 and b3) after 10ns;
            o0  <= (a3 and b0) xor (a1 and b2) after 10ns;
            md1 <= (a3 and b0) and (a1 and b2) after 10ns;
            o1  <= (c0 xor r0) xor md1 after 10ns;
            md2 <= (c0 and r0) or ((c0 xor r0) and md1
            after 10ns;
    o2  ⇐ ((l1 xor r1) xor md2 after 10ns;
    md3 ⇐ ((l1 and r1) or ((l1 xor r1) and md2
            after 10ns;
    o3  ⇐ md3 xor l2 after 10ns;
    d   ⇐ ((md3 and l2) or ((ld3 or (b3 and ((a1 or
            a2) or a3))) or ((b2 and (a2 or a3)) or (b1 and a3))))
            after 10ns ;
    c0  ⇐  o0 or d after 10ns;
    c1  ⇐  o1 or d after 10ns;
    c2  ⇐  o2 or d after 10ns;
    c3  ⇐  o3 or d after 10ns;
end
```

- In the following, we expect Add3 to be built like the adder of exercise 2, except for the signal c3. d is then given by d3.

```
architecture structure of multiplier is
signals n0, n1, n2, n3, n4, n5, n6, n7, n8, n9, l0, l1, l2, ld3, r0, rd1, a23, a123,
        md3, d1, d2, d3, d4, d01, d23, d234, o0, o1, o2, o3: Bit;
begin
        i1: AND      port map (a0, b0, n0);
        i2: AND      port map (a1, b0, n1);
        i3: AND      port map (a2, b0, n2);
        i4: AND      port map (a3, b0, n3);
        i5: AND      port map (a0, b1, n4);
        i6: AND      port map (a1, b1, n5);
        i7: AND      port map (a2, b1, n6);
        i8: AND      port map (a0, b2, n7);
        i9: AND          port map (a1, b2, n8);
        i10: AND         port map (a0, b3, n9);
        i11: half_adder  port map (n7, n9, rd1, r0);
        i12: Add3        port map (n0, n1, n2, n4, n5, n6, c2, c1, c0, cd3);
        i13: Add3        port map (c1, c0, n3, rd1, r0, n8, o1, o2, o0, md3);
        i14: half_adder  port map (c2, md3, d1, o3);
        i15: OR          port map (a2, a3, a23);
        i16: OR          port map (a23, a1, a123);
        i17: AND         port map (b3, a123, d2);
        i18: AND         port map (b2, a23, d3);
        i19: AND         port map (a3, b1, d4);
        i20: OR          port map (cd3, d1, d01);
        i21: OR          port map (cd3, d1, d01);
        i22: OR          port map (d2, d3, d23);
        i23: OR          port map (d23, d4, d234);
        i24: OR          port map (d01, d234, d);
        i25: OR          port map (d, o0, c0);
        i26: OR          port map (d, o1, c1);
        i27: OR          port map (d, o2, c2);
        i28: OR          port map (d, o3, c3);
end
```