Hardware design of a Sign-Magnitude sequential multiplier

Oprițoiu Flavius
flavius.opritoiu@cs.upt.ro

December 9, 2023

Introduction

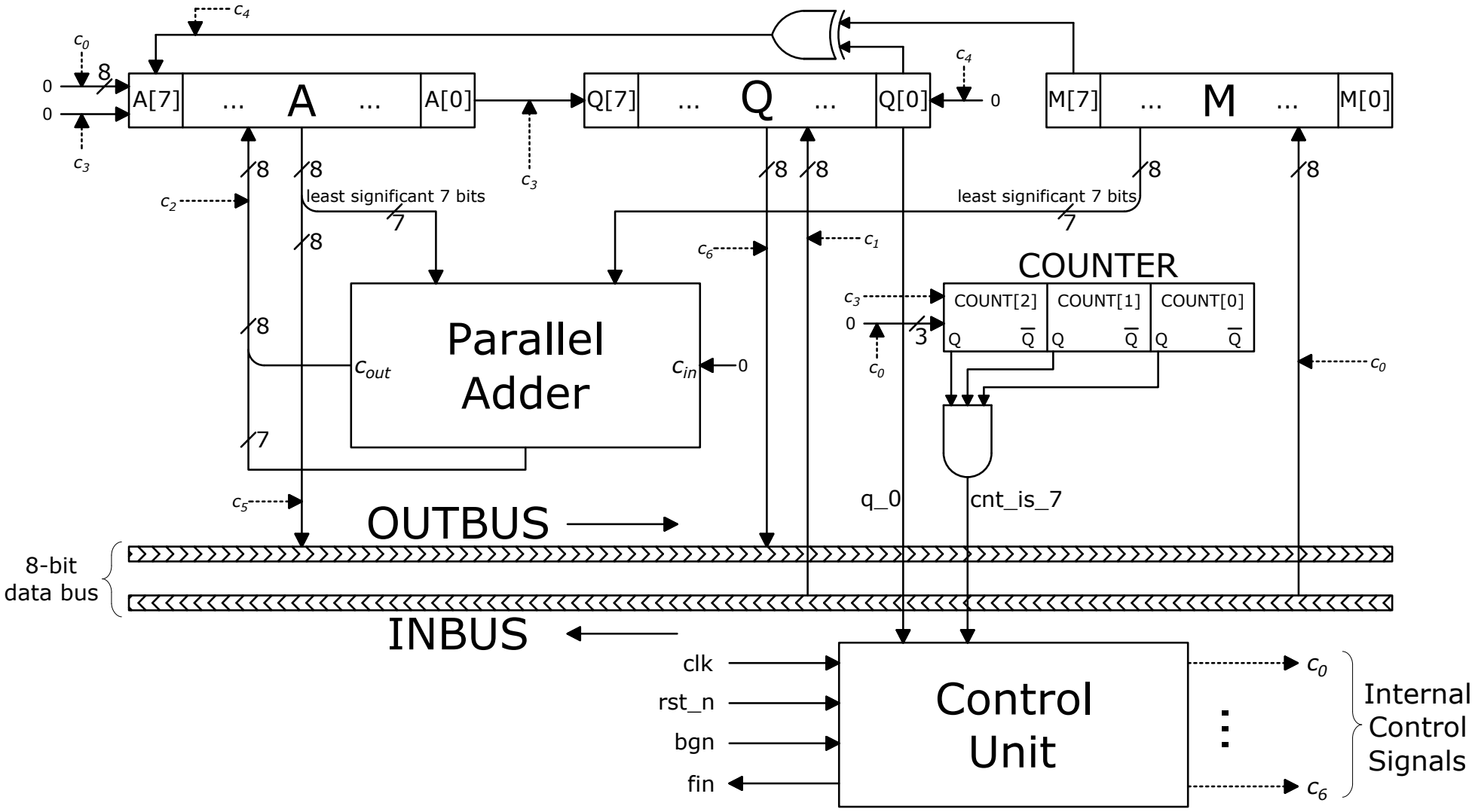
Objectives:

- Construct a sequential architecture for Sign-Magnitude multiplication

Architecture's characteristics:

- Operate fractional numbers on 8 bits
- Design and customize multiplier's components
- Design the control path of the architecture

Sign-Magnitude sequential multiplier architecture



Note: Except `rst_b`, all other signals are *synchronous*.

Register M (solved)

Stores the multiplicand, having the following interface:

- `ld_ibus`: loads multiplicand's value from INBUS
  - `ibus`: the INBUS lines
- `q`: register's content

```
1  module reg_m(
2    input clk, rst_b, ld_ibus, [7:0] ibus,
3    output reg [7:0] q
4  );
5    always @ (posedge clk, negedge rst_b)
6      if (!rst_b)                    q <= 0;
7      else if (ld_ibus)              q <= ibus;
8  endmodule
```

Register Q

Stores the multiplier, having the following interface:

- `clr_lsb`: clear register's LSB
- `ld_ibus`: loads multiplier's value from INBUS
  - `ibus`: the INBUS lines
- `sh_r`: right shifts the register's content
  - `sh_i`: bit to be loaded into MSB during right shift
- `ld_obus`: deliver accumulator's content onto OUTBUS
  - `obus`: the OUTBUS lines
- `q`: register's content

```
1  module reg_q(
2    input clk, rst_b, clr_lsb, ld_ibus, ld_obus, sh_r,
3    input sh_i, [7:0] ibus,
4    output reg [7:0] obus, [7:0] q
5  );
6    always @(posedge clk, negedge rst_b)
7      //treat inputs rst_b, clr_lsb, ld_ibus, sh_r here

9    always @(*) //write content to obus when ld_obus==1
10     obus = (ld_obus) ? q : 8'bz;
11 endmodule
```

Register A

Multiplier's accumulator, it has the following interface:

- `clr`: synchronous, active high reset
- `ld_sum`: loads adder's result into accumulator
  - `sum`: adder's result to be loaded into accumulator
- `ld_sgn`: loads accumulator's sign
  - `sgn`: accumulator's sign
- `ld_obus`: unload accumulator's content onto OUTBUS
  - `obus`: the OUTBUS lines
- `sh_r`: right shift register's content
  - `sh_i`: bit to be loaded into MSB during right shift
- `q`: accumulator's content

```
1  module reg_a(
2    input clk, rst_b, clr, sh_r, ld_sgn, ld_obus, ld_sum,
3    input sh_i, sgn, [7:0] sum,
4    output reg [7:0] obus, [7:0] q
5  );
6    //implementation here

8    //write content to obus
9  endmodule
```

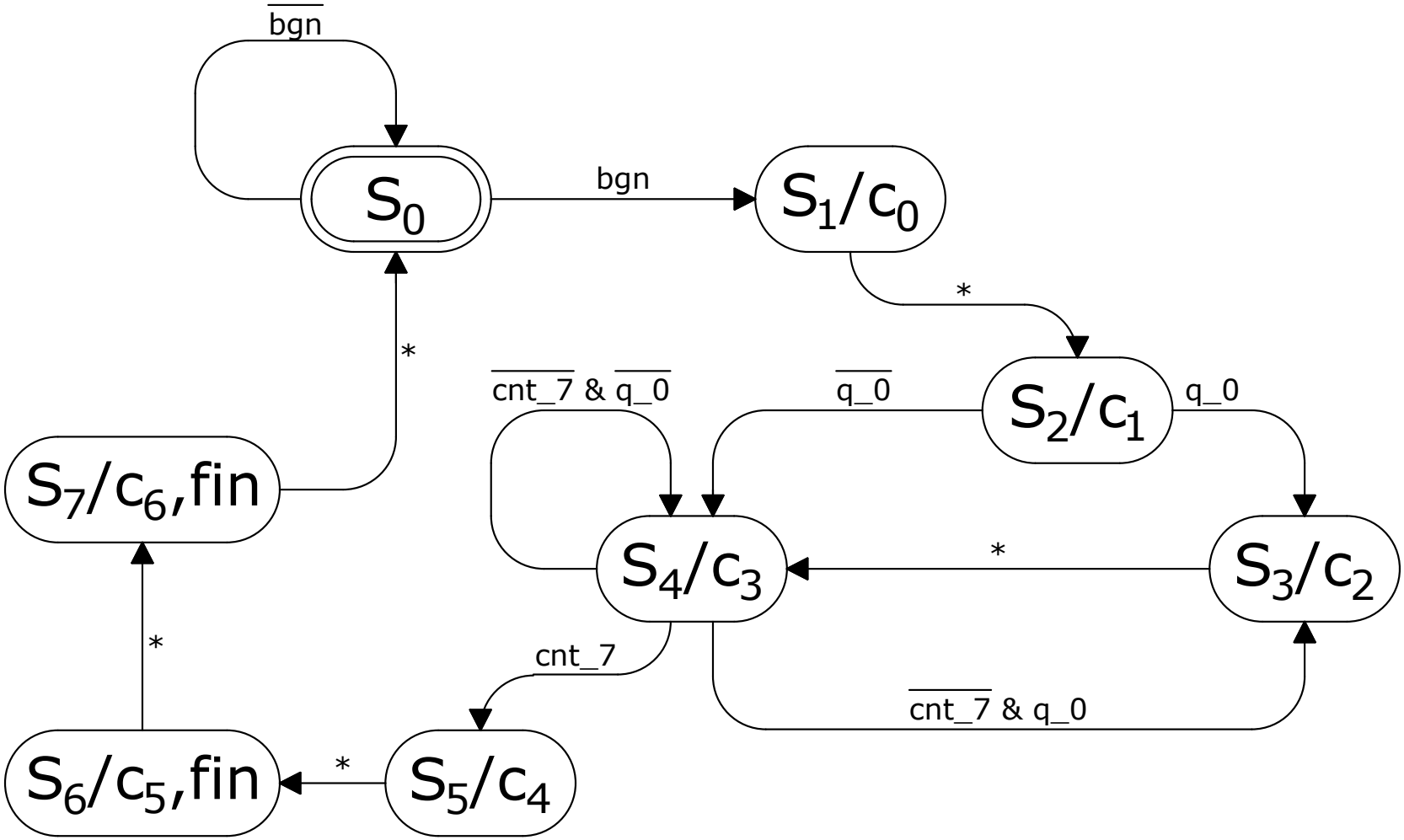
Control Unit

The control unit has the following interface:

- `bgn`: starts the multiplication
- `q_0`: register Q's LSB
- `cnt_is_7`: active if COUNTER's value is 7
- `c0`: clears registers A and CUNTER, loads M from INBUS
- `c1`: loads Q from INBUS
- `c2`: loads adder's result back into A
- `c3`: right-shifts A concatenated with Q (introduce a bit of `0` in register A's MSB), increments COUNTER
- `c4`: sets sign of register A, clear register Q's LSB
- `c5`/`c6`: deliver register A/Q to OUTBUS
- `fin`: marks multiplication completion, activated along `c5`, `c6`

```
1  module ctrl_u(
2    input clk, rst_b, bgn, q_0, cnt_is_7,
3    output c0, c1, c2, c3, c4, c5, c6, fin
4  );
5    //implementation here
6  endmodule
```

Control Unit (contd.)



Important: Control unit must be triggered by the opposite edge of the clock, compared to all the other sequential elements in the architecture!

Putting together all the components

The Sign-Magnitude sequential multiplier's architecture:

```
1   module sm_unit (
2     input clk, rst_b, bgn, [7:0] ibus,
3     output fin, [7:0] obus
4   );
5     //implementation here
6   endmodule

8   module sm_unit_tb;
9       reg clk, rst_b, bgn; reg [7:0] ibus; wire fin; wire [7:0] obus;

11      sm_unit test (.clk(clk), .rst_b(rst_b), .bgn(bgn), .ibus(ibus),
12          .fin(fin), .obus(obus));
13      localparam CLK_PERIOD=100, CLK_CYCLES=17, RST_PULSE=25;
14      localparam X=8'b10010111/*=-23*2^(-7)*/, Y=8'b10000011;/*=-3*2^(-7)*/
15      initial begin clk=1'd0; repeat (CLK_CYCLES*2) #(CLK_PERIOD/2) clk=~clk; end
16      initial begin rst_b=1'd0; #(RST_PULSE); rst_b=1'd1; end
17      initial begin bgn=1'd1; #200; bgn=1'd0; end
18      initial begin ibus=0; #100 ibus=X; #100 ibus=Y; end
19  endmodule
```

Testbench keeps OUTBUS in high impedance until 1300 time units, when, for 1 clock cycle OUTBUS becomes 8'b00000000, than, for another clock cycle, OUTBUS becomes 8'b10001010 only to remain in high impedance afterwards.