APPENDIX B

JULIÁN FRANCISCO ROMERO AMAYA
JEANPAULL VALENCIA QUINTERO

2025

Veriloga files used in the design process:

1 CTLE with cross-coupled pair and cherry-hooper active-inductor

```
‘include "constants.vams"
‘include "disciplines.vams"

module CTLE_ch_xcp_comportamental (Vinp, Vinn, Voutp, Voutn, Vref);

        input Vinp, Vinn;
        output Voutp, Voutn;
        inout Vref;
        electrical Vinp, Vinn, Voutp, Voutn, Vref,
            N01, N02, N03, N04, N05, N06;

        parameter real gm = 2.85m;
        parameter real Rs = 95;
        parameter real Cs = 846f;

        parameter real gm_ch = 7m;
        parameter real R_ch = 516;
        parameter real C_ch = 38f;
```

```
        parameter real gm_xcp = 5.13m;
        parameter real C_xcp = 51.3f;

        analog begin

        //Basic stage

        I(N01, Vref) <+ V(N01, Vref)/(Rs/2);
        I(N01, Vref) <+ (2*Cs)*ddt(V(N01, Vref));
        I(N02, Vref) <+ V(N02, Vref)/(Rs/2);
        I(N02, Vref) <+ (2*Cs)*ddt(V(N02, Vref));


        I(Voutn, N01) <+ gm*V(Vinp, N01);
        I(Voutp, N02) <+ gm*V(Vinn, N02);

        //Cherry Hooper

        I(Voutn, N03) <+ V(Voutn, N03)/R_ch;
        I(N03, Vref) <+ C_ch*ddt(V(N03, Vref));
        I(Voutn, Vref) <+ gm_ch*V(N03, Vref);

        I(Voutp, N04) <+ V(Voutp, N04)/R_ch;
        I(N04, Vref) <+ C_ch*ddt(V(N04, Vref));
        I(Voutp, Vref) <+ gm_ch*V(N04, Vref);

        //Cross-coupled pair

        I(Voutn, N05) <+ gm_xcp*V(Voutp, N05);
        I(N05, Vref) <+ (2*C_xcp)*ddt(V(N05, Vref));
        I(N05, Vref) <+ V(N05, Vref)/1000000;

        I(Voutp, N06) <+ gm_xcp*V(Voutn, N06);
        I(N06, Vref) <+ (2*C_xcp)*ddt(V(N06, Vref));
        I(N06, Vref) <+ V(N06, Vref)/1000000;

        end
endmodule
```

2 CTLE with gain-enhanced active-inductor

```
`include "constants.vams"
`include "disciplines.vams"

module CTLE_ge_comportamental (Vinp, Vinn, Voutp, Voutn, Vref);

        input Vinp, Vinn;
        output Voutp, Voutn;
        inout Vref;
        electrical Vinp, Vinn, Voutp, Voutn, Vref,
            N01, N02, N03, N04, N05, N06;

        //Basic parameters

        parameter real gm = 2.85m;
        parameter real Rs = 95;
        parameter real Cs = 846f;

        //Gain-enhance active inductor parameters

        parameter real Rg = 333;
        parameter real Rd = 100;
        parameter real Cgs = 75f;
        parameter real Cds = 1.5f;

        analog begin

        //Basic stage

        I(N01, Vref) <+ V(N01, Vref)/(Rs/2);
        I(N01, Vref) <+ (2*Cs)*ddt(V(N01, Vref));
        I(N02, Vref) <+ V(N02, Vref)/(Rs/2);
        I(N02, Vref) <+ (2*Cs)*ddt(V(N02, Vref));


        I(Voutn, N01) <+ gm*V(Vinp, N01);
        I(Voutp, N02) <+ gm*V(Vinn, N02);
```

```
        //Gain-enhance active inductor

        I(Voutn, N03) <+ V(Voutn, N03)/R_ch;
        I(N03, Vref) <+ C_ch*ddt(V(N03, Vref));
        I(Voutn, Vref) <+ gm_ch*V(N03, Vref);

        I(Voutp, N04) <+ V(Voutp, N04)/R_ch;
        I(N04, Vref) <+ C_ch*ddt(V(N04, Vref));
        I(Voutp, Vref) <+ gm_ch*V(N04, Vref);

        end

endmodule
```

3 CTLE cross-coupled pair and gain-enhanced active-inductor

```
    // Created Thu Mar 13 08:39:08 2025

`include "constants.vams"
`include "disciplines.vams"

module CTLE_ge_xcp_behavioral (Vinp, Vinn, Voutp, Voutn, Vref);

        input Vinp, Vinn;
        output Voutp, Voutn;
        inout Vref;
        electrical Vinp, Vinn, Voutp, Voutn, Vref, Vx1, Vx2, Vg1, Vg2,
            N01, N02, N03, N04, N05, N06;

        parameter real gm = 20m;
        parameter real Rs = 220;
        parameter real Cs = 630f;

        parameter real gm_ge = 6m;
        parameter real Rg_ge = 333;
        parameter real Rd_ge = 100;
        parameter real Cgs_ge = 75f;
        parameter real Cgd_ge = 1.5f;

        // Cross-coupled pair parameters
        parameter real gm_xcp = 2m;  // Example value, adjust as needed
        parameter real C_xcp = 50f;  // Example value, adjust as needed

        analog begin

        // Basic stage
        I(N01, Vref) <+ V(N01, Vref)/(Rs/2);
        I(N01, Vref) <+ (2*Cs)*ddt(V(N01, Vref));
        I(N02, Vref) <+ V(N02, Vref)/(Rs/2);
        I(N02, Vref) <+ (2*Cs)*ddt(V(N02, Vref));
```

```
        I(Voutn, N01) <+ gm*V(Vinp, N01);
        I(Voutp, N02) <+ gm*V(Vinn, N02);

        // Gain-enhance
        I(Vx1, Vref) <+ V(Vx1, Vref)/ro_ge;
        I(Vx1, Vref) <+ gm_ge*V(Vg1, Vref);
        I(Vx1, Voutn) <+ V(Vx1, Voutn)/Rd_ge;
        I(Vg1, Voutn) <+ V(Vg1, Voutn)/Rg_ge;

        I(Vx2, Vref) <+ V(Vx2, Vref)/ro_ge;
        I(Vx2, Vref) <+ gm_ge*V(Vg2, Vref);
        I(Vx2, Voutn) <+ V(Vx2, Voutn)/Rd_ge;
        I(Vg2, Voutn) <+ V(Vg2, Voutn)/Rg_ge;

        // Cross-coupled pair
        I(Vx1, N05) <+ gm_xcp*V(Vx2, N05);
        I(N05, Vref) <+ (2*C_xcp)*ddt(V(N05, Vref));
        I(N05, Vref) <+ V(N05, Vref)/1000000;

        I(Vx2, N06) <+ gm_xcp*V(Vx1, N06);
        I(N06, Vref) <+ (2*C_xcp)*ddt(V(N06, Vref));
        I(N06, Vref) <+ V(N06, Vref)/1000000;

        end
endmodule
```

4 PRBS31

```
`include "constants.vams"
`include "disciplines.vams"

(* instrument_module *)
module prbs31_jitter (Vplus, Vminus);
    inout Vplus, Vminus;
    electrical Vplus, Vminus;

    // Pseudo random bit sequence parameters
    parameter real tperiod = 50p from (0:inf);
    parameter integer seed_prbs = 1;      // Must be a non-zero 31-bit value
    parameter real vlogic_high = 1;
    parameter real vlogic_low = 0;
    parameter real tdel = 50p from (0:inf);
    parameter real trise = 1f;
    parameter real tfall = 1f;

    // Gaussian source parameters
    parameter real enable_gauss = 0;      // Enable Gaussian noise
    parameter real mean = 0.0;            // Mean of Gaussian noise [V]
    parameter real stddev = 1f;           // Standard deviation [V]
    parameter integer seed_n = 1;         // Seed for Gaussian noise

    // Uniform source parameters
    parameter real enable_uniform = 0;    // Enable uniform noise
    parameter real min_val_uniform = -1p; // Minimum noise value [V]
    parameter real max_val_uniform = 1p;  // Maximum noise value [V]

    // Sinusoidal source parameters
    parameter real enable_sin = 0;
    parameter real freq_sin = 2e9;
    parameter real amp_sin = 1p;

    // Internal variables
    real next, jitter, gauss_noise, uniform_noise,
        sin_noise, ideal_next, vout_val;
```

```
integer bit, lfsr_reg, feedback;  // LFSR state and feedback

analog begin
    @(initial_step) begin
        ideal_next = $abstime + tperiod;
        gauss_noise = $rdist_normal(seed_n, mean, stddev);
        uniform_noise = $rdist_uniform(seed_n, min_val_uniform,
            max_val_uniform);
        sin_noise = amp_sin * sin(2.0 * 3.14159 * freq_sin * $abstime);
        jitter = gauss_noise * enable_gauss + uniform_noise *
            enable_uniform + sin_noise * enable_sin;
        next = ideal_next + jitter;

        // Initialize LFSR (ensure non-zero 31-bit seed)
        lfsr_reg = seed_prbs & 31'h7FFFFFFF;  // Mask to 31 bits
        if (lfsr_reg == 0) lfsr_reg = 1;     // Prevent lock-up
        vout_val = vlogic_low;
    end

    $bound_step(tperiod/4);

    @(timer(next)) begin
        // Update noise sources
        gauss_noise = $rdist_normal(seed_n, mean, stddev);
        uniform_noise = $rdist_uniform(seed_n, min_val_uniform,
            max_val_uniform);
        sin_noise = amp_sin * sin(2.0 * 3.14159 * freq_sin * $abstime);
        jitter = gauss_noise * enable_gauss + uniform_noise *
            enable_uniform + sin_noise * enable_sin;

        // Schedule next edge
        ideal_next += tperiod;
        next = ideal_next + jitter;

        // Generate PRBS31 bit using LFSR (x^31 + x^28 + 1)
        bit = (lfsr_reg >> 30) & 1;          // Output = MSB (bit 30)
        feedback = ((lfsr_reg >> 30) & 1) ^
            ((lfsr_reg >> 27) & 1);  // Taps at bits 30 and 27
```

```
            lfsr_reg = ((lfsr_reg << 1) | feedback) &
                31'h7FFFFFFF;  // Shift and mask to 31 bits

            if (lfsr_reg == 0) lfsr_reg = 1;

            vout_val = (vlogic_high - vlogic_low) * bit + vlogic_low;
        end

        V(Vplus, Vminus) <+ transition(vout_val, tdel, trise, tfall);
    end
endmodule
```