ECE482 Final Project

Cristian Colon, Wei-Chun Hsu, Kriti Kumar, Mai Xu, Madhav Mehta

December 10, 2025

1 Overview of Design

1.1 Overall Purpose of the Design

The goal of this project was to implement a low-power 4-bit by 4-bit multiplier in the 45 nm CMOS process while meeting the constraints on latency, supply voltage and process-corner robustness. Our design is based on a classical array multiplier architecture, chosen for its regular structure, straightforward layout, and predictable timing behavior.

At a high level, the multiplier computes the partial products using a 4×4 matrix of AND gates and then sums those partial products through a grid of full adders and half adders. The array optimization allows the carries to ripple diagonally through the structure, creating a compact and fully combinational data path.

To satisfy the requirement that all external signals be synchronous, we added pipeline registers at both the inputs and the outputs of the multiplier. With these two register boundaries and the pipeline stages within the combinational array, the overall design achieves the required latency of three clock cycles.

Because the project places strong emphasis on minimizing power, we incorporated several architectural choices to reduce switching activity and to drive the large off-chip capacitive loads effectively. Most notably, dedicated super-buffers were inserted at the output stage to ensure that the product bits p_0 to p_7 can swing properly into the large loads specified in the test bench, without requiring excessively sized logic inside the multiplier core. This separation between the computational array and the high-drive output stage helps limit unnecessary dynamic power within the core logic.

Overall, the architecture is simple, fully deterministic, and well-suited for reliable operation across all five process corners while achieving low power and meeting the 3-cycle latency constraint.

1.2 Block Diagram Description

The top-level organization of the multiplier is divided into four major stages: the input registers, the array multiplier core, the output registers, and the output super-buffer stage. Figure 1 provides a high-level representation of the datapath, and the function of each block is summarized below.

The design begins with a bank of level shifters and input registers that captures the 1.8 V input signals $a[0:3]$ and $b[0:3]$ synchronously with the external clock. This ensures that the multiplier core always receives stable, time-aligned operands and satisfies the project requirement that all external inputs be registered. The outputs of these registers feed directly into the combinational array multiplier.

The core of the design is a classical 4×4 array multiplier consisting of a matrix of AND gates for partial-product generation and an array of full adders and half adders for summation.

The regularity of the array structure results in uniform routing and predictable timing, which simplifies the layout and helps ensure robustness across all process corners.

Following the array multiplier, the intermediate product signals are passed through a bank of output registers and level shifters located at the beginning of the output stage. These level shifters ensure compatibility between the core supply voltage and the I/O domain and provide correct logic levels before the final high-drive.

Because the eight output pins drive the large capacitive loads specified in the testbench, the level shifters are followed by super-buffers. These buffers isolate the multiplier core from the heavy off-chip loading and allow the product bits $p[0:7]$ to switch cleanly without forcing high drive strength inside the core logic. This completes the dataflow of the multiplier.

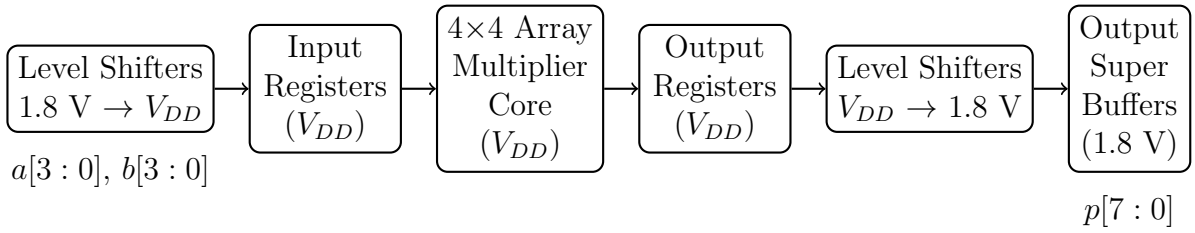

Figure 1: Top-level block diagram of the multiplier showing the 1.8 V → V_{DD} and V_{DD} → 1.8 V voltage transitions around the V_{DD} array core.

1.3 Special Architectural Features

Although the overall structure of the design follows a conventional 4-bit by 4-bit array multiplier, several architectural choices were made to meet the project requirements for low power, reliable I/O operation, and controlled voltage-domain transitions.

First, the input operands arrive as 1.8 V I/O signals, but the multiplier core is designed to operate at the lower core supply voltage of V_{DD}. Therefore, immediately before the input registers, a dedicated bank of 1.8 V → V_{DD} level shifters translates the signals into the core voltage domain. This allows all logic inside the array multiplier to run at V_{DD}, reducing dynamic power while keeping the internal circuit structure simple and fully combinational.

Second, the array multiplier core itself is strictly combinational and operates entirely at V_{DD}. The regular topology of the array ensures predictable timing and layout symmetry, which improves robustness across the five required process corners.

Third, because the external product bits $p[7:0]$ must ultimately be driven at 1.8 V, the outputs of the V_{DD} core are first passed through a bank of V_{DD} → 1.8 V level shifters. These restore the correct I/O voltage level before entering the high-drive output stage. A set of dedicated super–buffers follows the up-shifters to supply the large off-chip capacitive loads defined in the testbench without forcing any oversized devices inside the V_{DD} core. This

separation improves energy efficiency and isolates the core from the high-current output transitions.

Finally, the design uses both input and output register banks, as well as two pipeline stages within the multiplier. This results in a total latency of three clock cycles: two for the combinational multiplication at V_{DD}, and one for output capture at 1.8 V. This fully synchronous boundary on both sides simplifies timing closure and ensures reliable voltage-domain interaction.

2 Description of Each Block

2.1 DCVSL High to Low

The high-to-low level shifter does not have a gate-level representation, since it is an analog regenerative circuit rather than a Boolean logic operator. At a functional level, it translates 1.8 V input signals into V_{DD} core-compatible outputs while maintaining full logic swing and noise margin. The translation relies on a differential DCVSL structure with cross-coupled PMOS devices to regenerate the output levels. A detailed transistor-level schematic is provided in Section 5.

2.2 DCVSL Low to High

Like the high-to-low translator, the low-to-high level shifter does not map to a standard gate-level schematic. It restores V_{DD} core logic levels to the 1.8 V I/O domain using a differential DCVSL topology with a cross-coupled PMOS latch powered from VDDIO. This regenerative structure enables the circuit to produce a full 1.8 V logic '1' from a logic '1' V_{DD} input. The transistor-level implementation appears in Section 5.

2.3 C2MOS Register

The input and output pipeline registers are implemented using a C2MOS latch pair forming a master–slave flip-flop. Each latch consists of a transmission gate controlled by complementary clock phases (CLK and $\overline{\text{CLK}}$) feeding a static inverter-based storage node. When CLK is high, the master latch is transparent and the slave latch holds its previous value; when CLK is low, the master stores the input while the slave becomes transparent. This structure offers robust race-free operation and good noise margin with minimal power overhead, making it suitable for both the 1.8 V and V_{DD} domains.

2.4 FO1 Inverter and Transmission Gate

The FO1 inverter and transmission gate (T-gate) cells are used throughout the multiplier to buffer internal nodes, restore logic levels, and implement signal steering where needed. The FO1 inverter is a minimum-sized CMOS inverter designed to drive a fan-out of one, providing clean transitions with minimal area and power. It is primarily used for local regeneration of XOR and carry signals within the adder stages.

The transmission gate implements a bidirectional pass switch using parallel NMOS and PMOS devices controlled by complementary gate signals. This structure provides low-resistance signal transfer and is used in portions of the full-adder logic where conditional propagation of intermediate nodes is required. Both cells use minimum-length devices, with slight PMOS upsizing in the FO1 inverter to balance rise and fall delays.

2.5 AND Gate

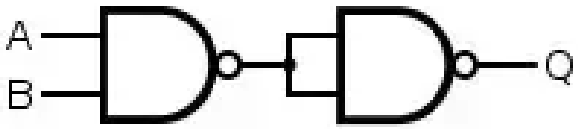

Figure 2: AND Gate Level Schematic

The AND gate used for partial-product generation is implemented using a standard NAND-plus-inverter structure. A 2-input NAND gate first computes

$$Y_{\text{nand}} = \overline{A \cdot B},$$

and a minimum-sized CMOS inverter restores the non-inverted logic level to produce

$$Y = A \cdot B.$$

This approach provides better drive strength and cleaner switching characteristics than implementing the AND function directly with series NMOS devices. All transistors use minimum channel length, and the PMOS widths are scaled to approximately twice the NMOS width to balance the rise/fall delays.

2.6 Half Adder

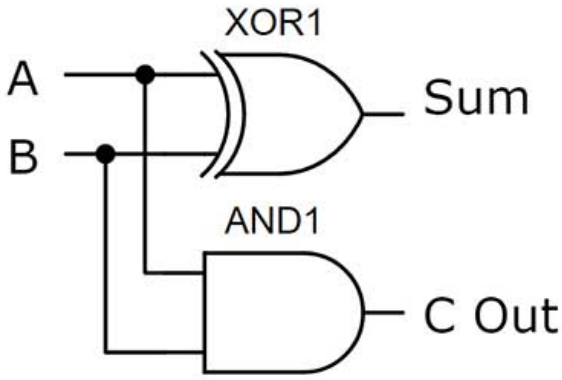

Figure 3: Half Adder Gate Level Schematic

The Half Adder design was implemented via transmission gate logic as this enabled the implementation to be completed with only 10 transistors. Consequently, reducing the power consumed by the block. The NMOS transistors were sized at W_{min} and $\beta = 2$ was used within the transmission gates. These values were chosen to reduce the power consumed by the transmission gates while ensuring symmetric noise margins.

2.7 Full Adder

The full adder computes the sum and carry-out from three inputs (A, B, C_{in}) using

$$\text{Sum} = A \oplus B \oplus C_{\text{in}}, \qquad C_{\text{out}} = AB + C_{\text{in}}(A \oplus B).$$

The implemented circuit uses a static-CMOS realization of the XOR and carry functions. A two-stage XOR topology is used for the sum output, while the carry-out is generated using a compact propagate/generate structure. Transistor sizing is kept close to minimum except in critical carry branches, where slight upsizing improves delay through the ripple path. Figure 4 shows the gate-level functional representation used during early design and verification. Figure 13 shows the transistor-level implementation used in the final layout.

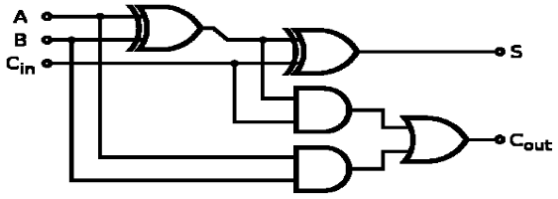

Figure 4: Gate-level schematic of the full adder

2.8 Carry Ripple Adder

The carry ripple adder (CRA) is constructed by cascading seven full-adder (FA) cells, where each cell propagates its carry-out to the carry-in of the next stage. This structure is used to accumulate the final partial-product results produced by the array. For the target bit-width in this design, the 7-FA ripple topology offers a compact and power-efficient summation implementation with straightforward timing behavior. A gate-level representation of the CRA is shown in Fig. 5.

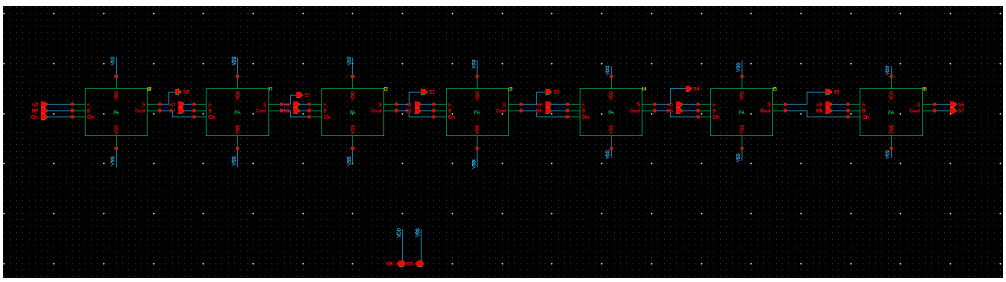

Figure 5: Carry Ripple Adder Gate Level Schematic

2.9 SuperBuffer

Each output bit of the multiplier is driven by a six-stage tapered superbuffer operating from the 1.8 V I/O supply. Because the core logic cannot directly drive the large pad capacitances, a geometric chain of increasingly larger inverters is used to provide the necessary current

while minimizing delay. The sizing follows a roughly constant taper factor between stages, consistent with logical-effort design methodology.

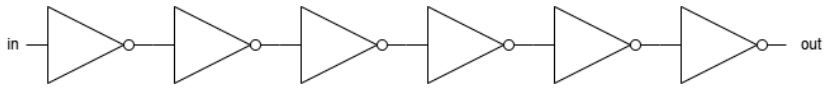

Figure 6: SuperBuffer Gate Level Schematic

The first stage is minimally sized at $W_p/W_n = 640/320\,\text{nm}$ to avoid excessive loading on the low-to-high level shifter output. Subsequent stages increase in width to $840/640\,\text{nm}$, $2560/1280\,\text{nm}$, and $8920/4920\,\text{nm}$, providing progressively greater drive strength. The final stage is significantly upsized to $144000/72000\,\text{nm}$ in order to meet the required $1.8\,\text{V}$ output slew and to drive the large off-chip load associated with pad connections.
This five-stage taper provides:

- near-optimal delay through the buffer chain,
- reduced short-circuit and dynamic power by maintaining consistent stage effort,
- isolation between the low-voltage core and the high-capacitance I/O domain,
- clean, full-swing $1.8\,\text{V}$ output transitions suitable for external interfacing.

3 Latency

The latency of the designed circuit is **3** clock cycles.

In this project, latency refers to the number of additional cycles needed for the product of the two input values to appear at the output pins after the inputs are applied. Because the multiplier requires synchronous output, registers are placed in the output stage and this alone results in a baseline latency of one cycle.

To achieve the project's goal of power-optimization, the design adds two internal pipeline stages to the multiplier core, increasing the total latency to three cycles. These added stages reduce the amount of combinational logic between registers, which helps the circuit maintain timing as the supply voltage is lowered. Reducing V_{DD} decreases dynamic power but increases gate delay, so shorter logic paths are essential for reliable operation under these conditions.

By introducing these pipeline stages, the design maintains stable and predictable output behavior while enabling the voltage scaling needed for power reduction. The chosen latency of three cycles therefore represents a deliberate balance between maintaining functional correctness and achieving the desired power savings.

4 Power Reduction Strategy

The power consumed by the multiplier can be reduced through multiple strategies at the gate level and the architectural level. Section 4 discusses those strategies.

4.1 Multiplier Design Choice

In general, one can conceptualize the function of an array multiplier circuit into three distinct stages: partial-product generation, partial-product accumulation, and final addition. Due to the decision to incorporate two internal pipeline stages, each stage of this multiplier design was separated by a set of registers, meaning that each stage could be optimized independently of each other. Much of the following discussion regarding the multiplier stages comes from chapter 11.4 of the second edition of "Digital Integrated Circuits: A Design Perspective" by Rabaey, Chandrakasan, and Nikolic, as that chapter inspired many of the architectural decisions made for the multiplier.

4.1.1 Partial-Product Generation Stage

The Partial-Product Generation stage encompasses the first stage of the multiplier circuit, wherein each bit of one input is multiplied with each bit of the other input. In it's simplest form, this results from the logical AND of the multiplicand with each bit of the multiplier. For an M x N multiplier (where M and N represent the number of bits in the first and second input, respectively), this stage produces M*N partial products. Other strategies, such as Booth recoding, have been proposed to reduce the number of partial products created, but often at the cost of additional logical complexity. In this project, it was decided to use a 4 x 4 array of AND gates to generate the partial products, as it proved to be the simplest in terms of logical complexity and layout while also keeping the critical path through this stage at one NAND gate and one inverter.

4.1.2 Partial-Product Accumulation Stage

The Partial-Product Accumulation stage encompasses the second stage of the multiplier, wherein some or all of the partial products are summed together in preparation for the final addition step. Usually, this stage contains 1-bit half-adders and full-adders, but the choice in design of this stage determines both the number and arrangement of those adder cells.

The most straightforward implementation is that of an array multiplier. In this design, M-1 N-bit adders are used to sum all M*N partial products, producing the final product as its result. The benefit of this design is it is very layout efficient, as the use of (M-1)*N adder cells usually allows for a compact rectangular arrangement. However, the computation of this design's critical path is not straightforward, and there are multiple paths that can serve as the critical path. This makes delay minimization non-trivial, since it requires a reduction in delay along all possible critical paths, not just one.

Another implementation is that of a carry-save multiplier. A carry-save architecture differs from the array architecture in that it provides a different propagation path for the carry-out

bits and utilizes a vector-merging adder. While requiring one adder more than the array architecture, the carry-save architecture has the benefit of both having one unique critical path and making the delay of that path shorter than that of the array architecture.

A third implementation (and the one used for this multiplier) is that of a Wallace Tree multiplier. In this design, half and full adders are used to "compress" the number of partial products before performing the final addition. In the case of a 4 x 4 multiplier, this results in the use of 6 adder cells in this stage, substantially fewer than the other implementations. This decrease in the number of adders would allow for much more flexibility in the delay of this stage, thus allowing more room for potential power reduction. As such, this architecture was ultimately chosen.

4.1.3 Final Addition Stage

The Final Addition stage is where all remaining partial products (if any) are summed together to obtain the final product. The design of this stage depends on the design chosen for the Partial-Product Accumulation stage. For the sake of brevity, the following discussion of the final addition stage is applicable only to the Wallace Tree architecture.

If a multiplier uses the Wallace Tree architecture described in section 4.1.2, the final addition stage will need to sum 13 partial products. This can be done with a 7-bit adder. The various adder design possibilities are discussed in section 4.2, but the chosen design was that of a 7-bit carry-ripple adder, as it presented both the simplest implementation logically and was the lowest-power option among the designs discussed in section 4.2.

4.2 Adder Design Choice

4.2.1 Carry-Ripple

A chain of full adders where each stage waits for the previous carry before producing its own sum and carry. It is the simplest and smallest adder, giving low power and area but the slowest delay because the carry must ripple through all bits.

4.2.2 Carry-Lookahead

Uses generate/propagate logic to predict carries in parallel instead of waiting for ripple. By computing group carries quickly, it achieves much shorter delay than RCA, at the cost of more complex logic and higher area/power.

4.2.3 Carry-Select

Splits the adder into blocks and precomputes the sum of each block twice, once assuming carry = 0 and once assuming carry = 1, then uses a mux to pick the correct result when the actual carry arrives. This gives speed close to CLA with simpler logic, but uses more hardware and power than RCA.

As can be seen in the above description of each adder design, the Ripple Carry Adder has the lowest power consumption and so we decided to use the Ripple Carry Adder in our design.

4.3 Transistor Sizing

The sizes of the transistors used in this multiplier were chosen mainly to reduce the parasitic capacitances present in the circuit, as these would all contribute to the multiplier's power dissipation. As such, almost all NMOSes (except for those used in ratioed logic like the DCVSLs or those in the timing critical paths of the full adder) were sized to be minimum width, and all PMOSes were sized to be $\beta = 2$ to get approximately equal low-to-high and high-to-low propagation delays.

4.4 Pipelining

Pipelining refers to the practice of breaking up a logic circuit into multiple sections, each separated by a set of registers. While this causes the latency of the circuit to increase, the operating frequency of the circuit still appears to be the same once outputs start appearing. As such, this technique is usually used to increase the operating frequency of a circuit. However, this technique can also be used to reduce the power of a circuit while keeping the operating frequency of the circuit the same, as it relaxes delay requirements such that each individual section can operate at slower speeds. The allowance of slower operation can be used, in turn, to reduce power consumption through various methods. This allowance of power reduction is what led to the incorporation of pipelining in this multiplier.

4.5 V_{DD} Reduction

Once a circuit has been pipelined, many strategies can be used to reduce its power consumption. The primary strategy used in this design is the reduction of V_{DD}. Since the power dissipated by the circuit is roughly described by

$$\mathrm{P} = f \times V_{DD}^2 \times C_{\mathrm{L}},$$

reducing V_{DD} can lead to substantial power savings. Figure 7 gives an idea of the savings that can be achieved with reductions in V_{DD}. Due to this potential, this team also attempted to reduce V_{DD} as much as possible.

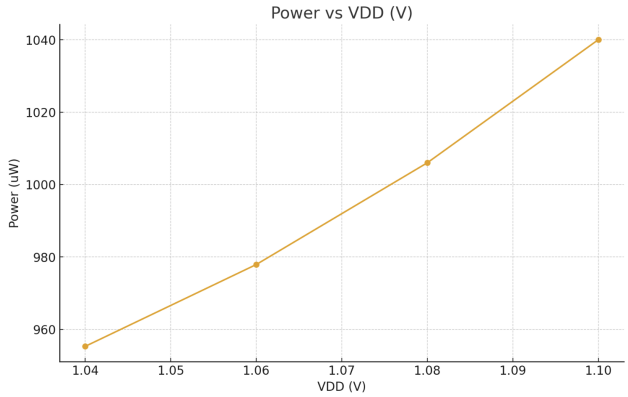

Figure 7: Graph showcasing the relation between reduced V_{DD} and Power consumed by the multiplier

5 Transistor Level Schematic

5.1 High-to-Low DCVSL Level Shifter

Figure 8 shows the transistor-level implementation of the 1.8 V to V_{DD} (High-to-Low) level shifter. The circuit uses a differential DCVSL topology with cross-coupled PMOS devices referenced to the higher voltage domain (VDDIO), and an NMOS differential pair referenced to the core supply (V_{DD}). The complementary input pair (`io`, `io_bar`) ensures correct switching and full-swing regeneration at the reduced core voltage.
PMOS transistors M1/M3 operate from VDDIO and provide regenerative pull-up action, while the NMOS devices M2/M4 ensure that output swing is clamped safely within the V_{DD} domain. The sizing shown in the schematic was selected to provide strong regeneration while limiting contention during cross-coupled switching.

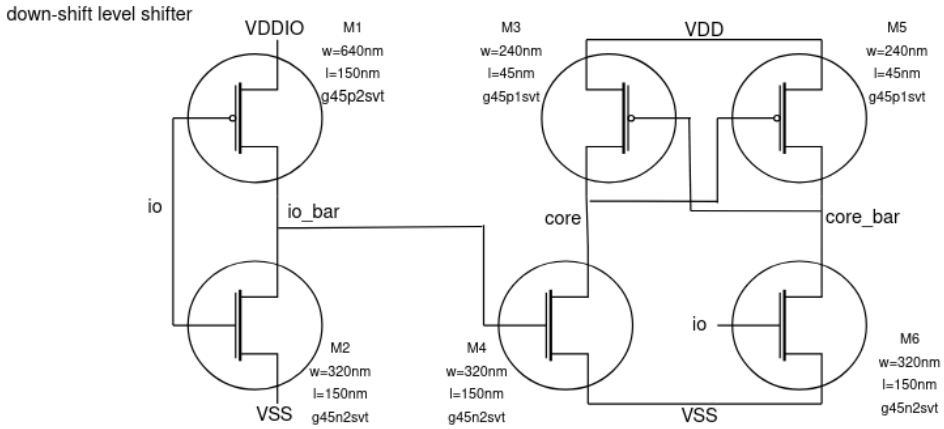

Figure 8: Transistor-level schematic of the High-to-Low (1.8 V $\rightarrow$ V_{DD}) DCVSL level shifter.

5.2 Low-to-High DCVSL Level Shifter

Figure 9 shows the transistor-level implementation of the V_{DD} to 1.8 V (Low-to-High) level shifter used at the output of the core multiplier. This circuit uses a DCVSL-style regenerative latch referenced to the higher-voltage domain (VDDIO). The cross-coupled PMOS devices M3/M5 provide positive feedback to drive the output nodes (`io`, `io_bar`) fully to 1.8 V.
The NMOS input pair M1/M2 is driven from the core domain (V_{DD}) and ensures that the logic levels presented at the latch inputs (`core`, `core_bar`) do not overstress the core-side devices. Devices M4/M6 act as discharge transistors sized to guarantee that the output transitions occur reliably even under process and voltage corners.
The sizes of the up-shift PMOS devices were increased relative to the down-shift network to provide sufficient regeneration strength at the higher supply voltage and to overcome the weaker core input drive.

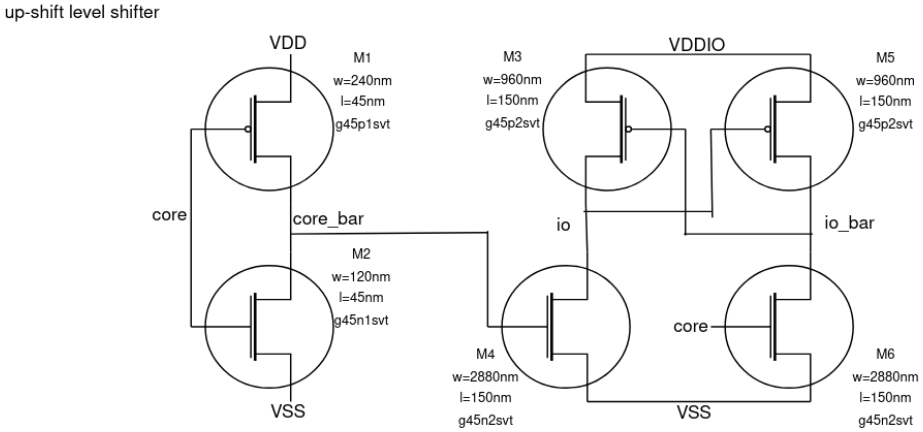

Figure 9: Transistor-level schematic of the Low-to-High ($V_{DD} \rightarrow 1.8\,\text{V}$) DCVSL level shifter.

5.3 FO1 Inverter and Transmission Gate

Figure 10 shows the transistor-level implementation of the FO1 inverter and the transmission gate (TG) used extensively throughout the design. The FO1 inverter consists of a minimum-sized PMOS and NMOS pair (M1 and M2), sized with a 2:1 width ratio to balance rising and falling transition times. This inverter serves as the basic gain and restoration stage within the arithmetic blocks as well as the C2MOS latch.
The transmission gate is implemented using complementary NMOS and PMOS pass devices (M3 and M4). When the enable signal is high and its complement is low, both devices conduct, resulting in low on-resistance and symmetric conduction for logic '0' and logic '1'. When the enable is deasserted, both devices turn off and isolate the input from the output. The sizing of the TG devices matches the FO1 inverter's NMOS/PMOS widths to ensure the propagation delay through the pass network remains dominated by downstream logic rather than by the switch itself.

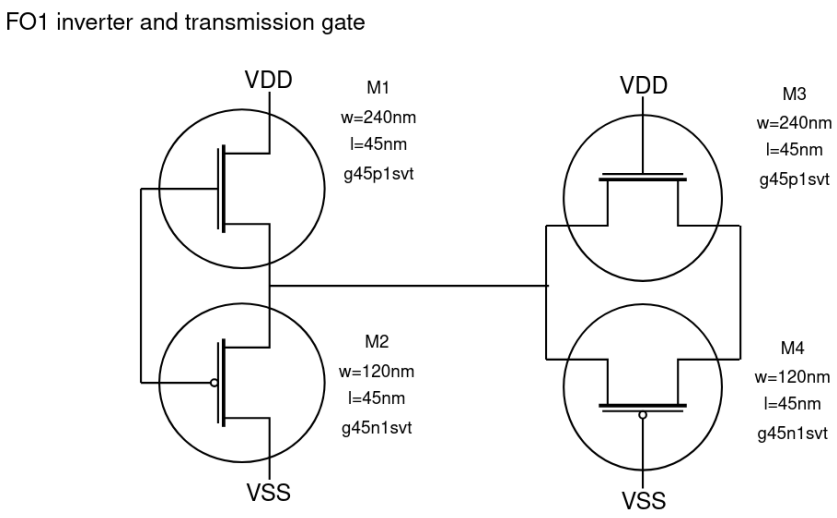

Figure 10: Transistor-level schematic of the FO1 inverter and transmission gate.

5.4 AND Gate

The transistor-level implementation of the AND gate is shown in Fig. 11. The design uses a static CMOS structure in which a two-input NAND gate is followed by a minimum-sized CMOS inverter to restore the non inverted logic level. Transistors M1 and M2 form the parallel PMOS pull-up network, while M3 and M4 implement the series NMOS pull-down stack. This produces the inverted intermediate signal $\overline{A \cdot B}$.

To obtain the final AND output, the NAND stage drives an inverter (M5–M6), which re-establishes proper CMOS logic polarity and provides improved output drive. PMOS devices in both stages are sized approximately $2\times$ their NMOS counterparts to balance rising and falling delays. The resulting AND cell provides clean logic levels and is reused throughout the multiplier's partial-product generation logic.

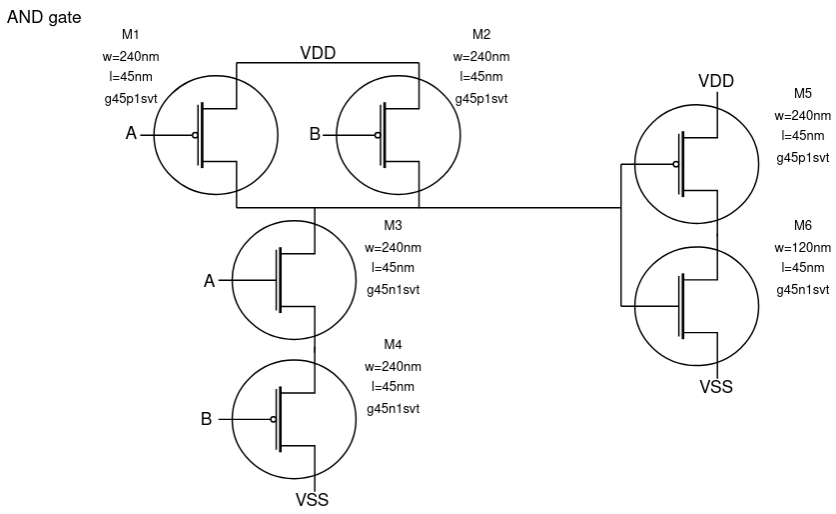

Figure 11: Transistor-level schematic of the AND gate (NAND + inverter).

5.5 Half Adder

Figure 12 shows the transistor-level implementation of the half adder. The circuit generates the partial sum and carry outputs using a static-CMOS XOR network combined with a simple AND stage. The XOR is implemented using complementary pull-up and pull-down transistor pairs arranged to produce $A \oplus B$ with full logic swing and balanced rise/fall times. The AND function that produces the carry bit is implemented using a series-stacked NMOS pair and a parallel PMOS pair.

Device widths for the XOR transistors were chosen to match the delay characteristics of the full adder sum path, ensuring consistent timing within the larger arithmetic datapath. The AND branch uses minimum-sized devices because the carry-out of a half adder does not participate in long carry chains and therefore does not require upsizing.

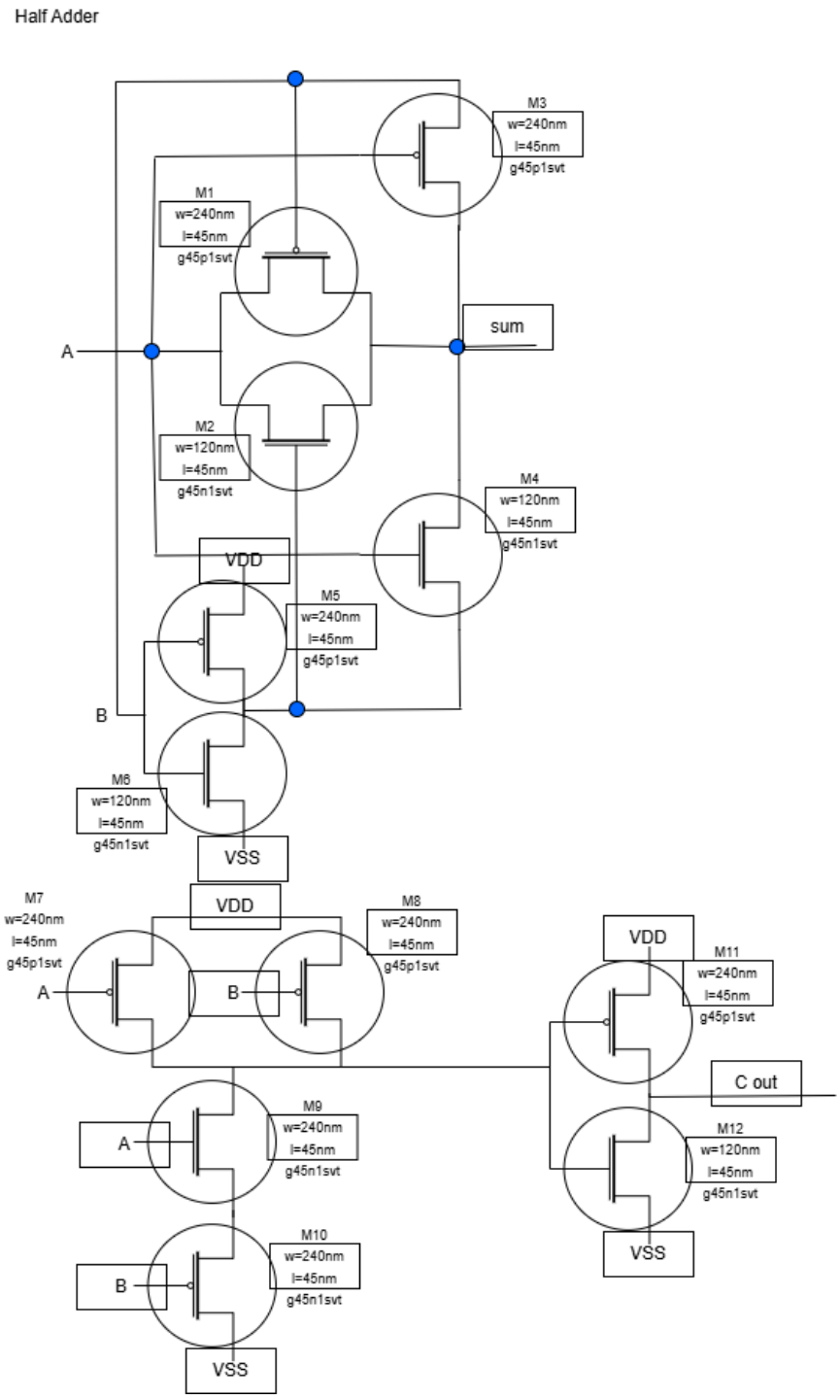

Figure 12: Transistor-level schematic of the half adder.

5.6 Full Adder

Figure 13 shows the transistor-level implementation of the full adder used in the carry-ripple adder stage of the multiplier. The design computes both the sum ($A \oplus B \oplus C_{\text{in}}$) and carry-out

($C_{\text{out}} = AB + C_{\text{in}}(A \oplus B)$) using a combination of transistor-level XOR networks and static CMOS carry-generation logic.

The sum path consists of two cascaded XOR blocks: the first computes $A \oplus B$, and the second XORs this intermediate result with C_{in}. Transistor sizes in the sum chain are kept relatively small since the sum output does not participate in long propagation chains.

The carry-out circuitry is implemented with a propagate–generate structure. The generate term (AB) is produced using a compact static CMOS AND gate, while the propagate term $C_{\text{in}}(A \oplus B)$ uses the intermediate XOR node. Devices in the carry path are slightly upsized relative to the XOR transistors to reduce critical-path delay through the ripple network.

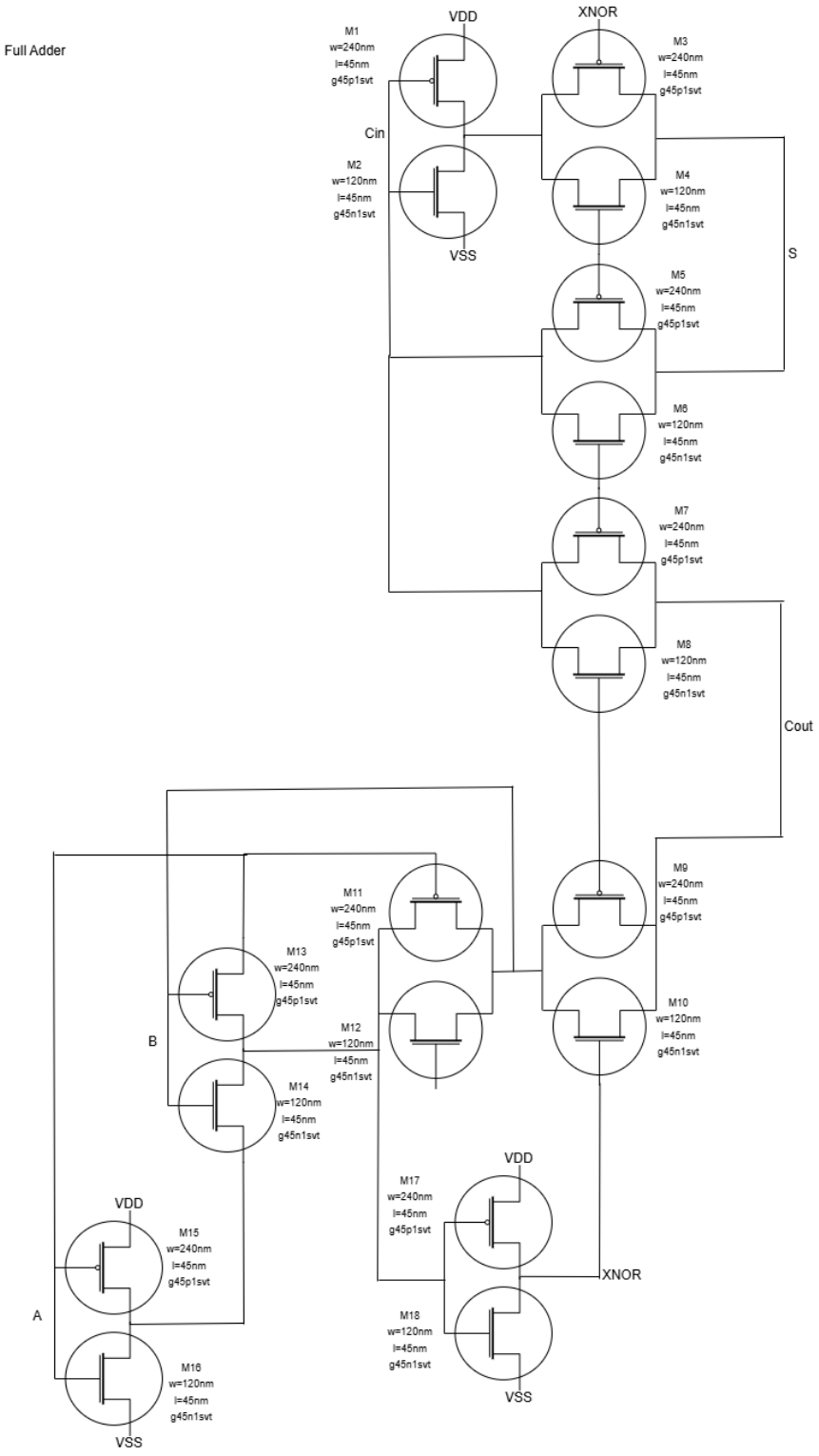

Figure 13: Transistor-level schematic of the full adder used in the carry-ripple chain.

5.7 C2MOS Register

Figure 14 shows the transistor-level implementation of the C2MOS (master–slave) register used at the input and output pipeline stages of the multiplier. The register consists of two level-sensitive latches connected in series. Each latch uses a transmission gate (TG) controlled by complementary clock phases (CLK and $\overline{\text{CLK}}$) followed by a static CMOS inverter that holds the stored value when its TG is off.

When CLK is high, the master latch is transparent and the slave latch holds its previous value. When CLK goes low, the master latch captures its input, and the slave latch becomes transparent, transferring the stored value to the output. This configuration prevents race conditions and ensures robust edge-triggered behavior without requiring explicit pulse generation.

The transmission gates are sized to minimize RC delay when transparent, while the inverter devices use FO1 sizing for adequate drive strength and low static power. The PMOS devices in the latch inverters are approximately 2× wider than their NMOS counterparts to balance rise and fall times.

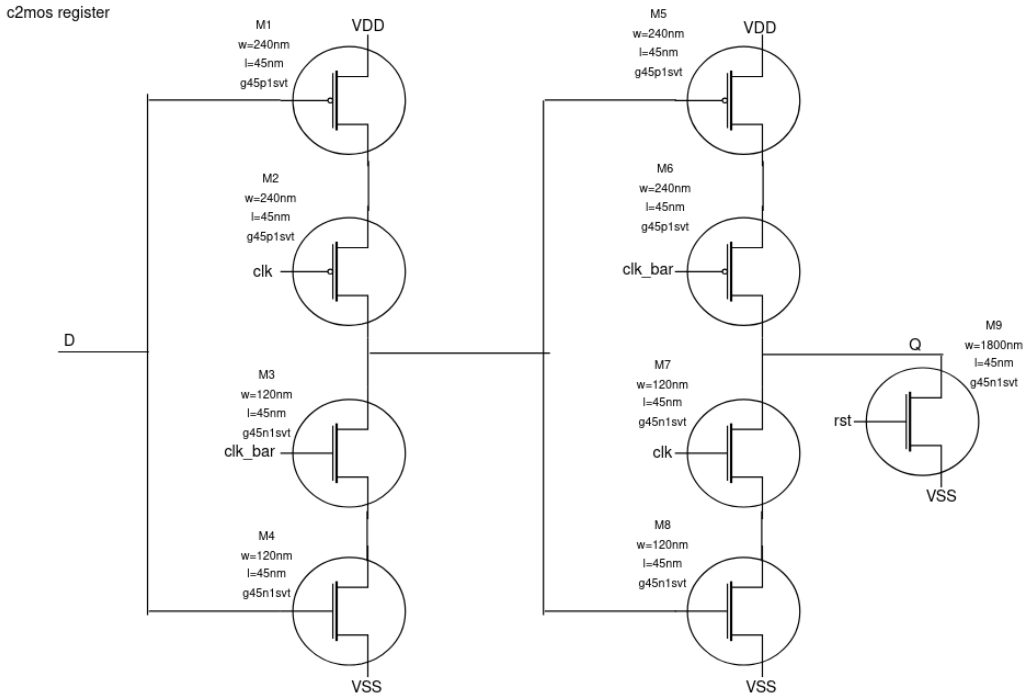

Figure 14: Transistor-level schematic of the C2MOS master–slave register.

5.8 SuperBuffer

Figure 15 shows the transistor-level implementation of the output superbuffer used to drive the 1.8 V product pins $p[7:0]$. This block consists of a tapered chain of inverters, each stage increasing in size to efficiently drive the large capacitive load associated with pad and routing capacitances. The superbuffer is powered from the VDDIO (1.8 V) domain and is preceded by the Low-to-High level shifter, ensuring safe device operation and full-swing output levels.

The first stage uses minimum-sized transistors, while subsequent stages are upsized by a nearly constant geometric factor to minimize the overall propagation delay according to logical effort principles. The final inverter stage contains significantly larger PMOS and NMOS devices to supply the required output current for fast transitions at the chip boundary. This sizing strategy balances delay, area, and dynamic power.

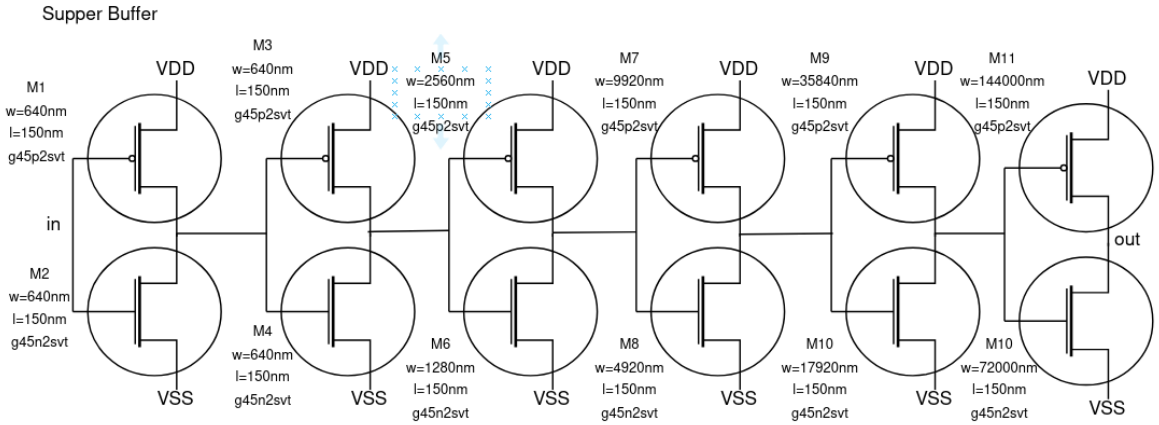

Figure 15: Transistor-level schematic of the multi-stage output superbuffer for 1.8 V pad drive.

5.9 Transistor Sizing Summary

Table 1 and Table 2 summarize the transistor sizing used in the design. Widths (W) are listed in nanometers (nm); all channel lengths are L = 45 nm unless otherwise specified. PMOS devices are generally sized larger than NMOS devices ($\approx 2\times$) to balance rise/fall timing. Critical carry-path and output-driver devices are upsized as indicated to meet delay and drive requirements.

Table 1: Per-cell transistor sizing (W_p/W_n in nm; L = 45 nm, except for SB, L = 150 nm).

Cell	W_p / W_n (nm)	Notes
FO1 inverter	240 / 120	FO1: minimum inverter used for buffering
Transmission gate (TG)	240 / 240	PMOS/NMOS matched for symmetric pass
NAND / AND primitive	240 / 240	NAND stage; inverter follows for AND
Half adder (XOR)	240 / 120	XOR pull-ups / pull-downs
Half adder (carry)	240 / 120	AND/ carry path (min size)
Full adder (sum)	240 / 120	Sum-path kept small for area/power
Full adder (carry)	240 / 120	Carry-path devices
DCVSL H $\rightarrow$ L	640 / 320	Down-shift translator (large PMOS pull-ups)
DCVSL L $\rightarrow$ H	960 / 2880	Up-shift translator (strong PMOS for regen)
C2MOS register	240 / 120	TG: 240/240 for pass devices
Superbuffer (stage 1)	640 / 320	Initial buffer stage
Superbuffer (stage 2)	640 / 320	Initial buffer stage
Superbuffer (stage 3)	2560 / 1280	Intermediate
Superbuffer (stage 4)	9920 / 4920	Large intermediate
Superbuffer (stage 5)	35840 / 17920	Near-final stage
Superbuffer (final)	144000 / 72000	Final output stage (pad drive)

Table 2: Detailed transistor sizing (W in nm; L = 45 nm).

Cell	Device	W (nm)	L (nm)
FO1 inverter	PMOS (pull-up)	240	45
FO1 inverter	NMOS (pull-down)	120	45
Transmission gate	PMOS (pass)	240	45
Transmission gate	NMOS (pass)	240	45
NAND primitive	PMOS (pull-up)	240	45
NAND primitive	NMOS (pull-down stack)	240	45
AND output inverter	PMOS	240	45
AND output inverter	NMOS	120	45
Half adder (XOR)	XOR PMOS	240	45
Half adder (XOR)	XOR NMOS	120	45
Half adder (carry)	Carry NMOS	120	45
Half adder (carry)	Carry PMOS	240	45
Full adder (sum)	Sum-path PMOS	240	45
Full adder (sum)	Sum-path NMOS	120	45
Full adder (carry)	Carry-path PMOS	240	45
Full adder (carry)	Carry-path NMOS	120	45
DCVSL H→L	Pull-up PMOS (VDDIO side)	640	45
DCVSL H→L	Pull-down NMOS (core side)	320	45
DCVSL L→H	Pull-up PMOS (VDDIO side)	960	45
DCVSL L→H	Pull-down NMOS (core/protect)	2880	45
C2MOS register	Inverter PMOS	240	45
C2MOS register	Inverter NMOS	120	45
C2MOS register	TG PMOS	240	45
C2MOS register	TG NMOS	240	45
Superbuffer	Stage 1 PMOS	640	150
Superbuffer	Stage 1 NMOS	320	150
Superbuffer	Stage 2 PMOS	640	150
Superbuffer	Stage 2 NMOS	320	150
Superbuffer	Stage 3 PMOS	2560	150
Superbuffer	Stage 3 NMOS	1280	150
Superbuffer	Stage 4 PMOS	9920	150
Superbuffer	Stage 4 NMOS	4920	150
Superbuffer	Stage 5 PMOS	35840	150
Superbuffer	Stage 5 NMOS	17920	150
Superbuffer	Final PMOS	144000	150
Superbuffer	Final NMOS	72000	150
Other (reset / special)	NMOS	1800	45

6 Results

6.1 Transient Simulations

After the multiplier layout was integrated with the PDN layout, an RC extraction was performed to obtain a netlist for the entire system. This netlist was then tested with both provided test benches at a V_{DD} value of 1.04 V. Figures 16 - 20 show the multiplier outputs for test bench 1 at the chip side bond pads (the oc signals) across all 5 process corners.

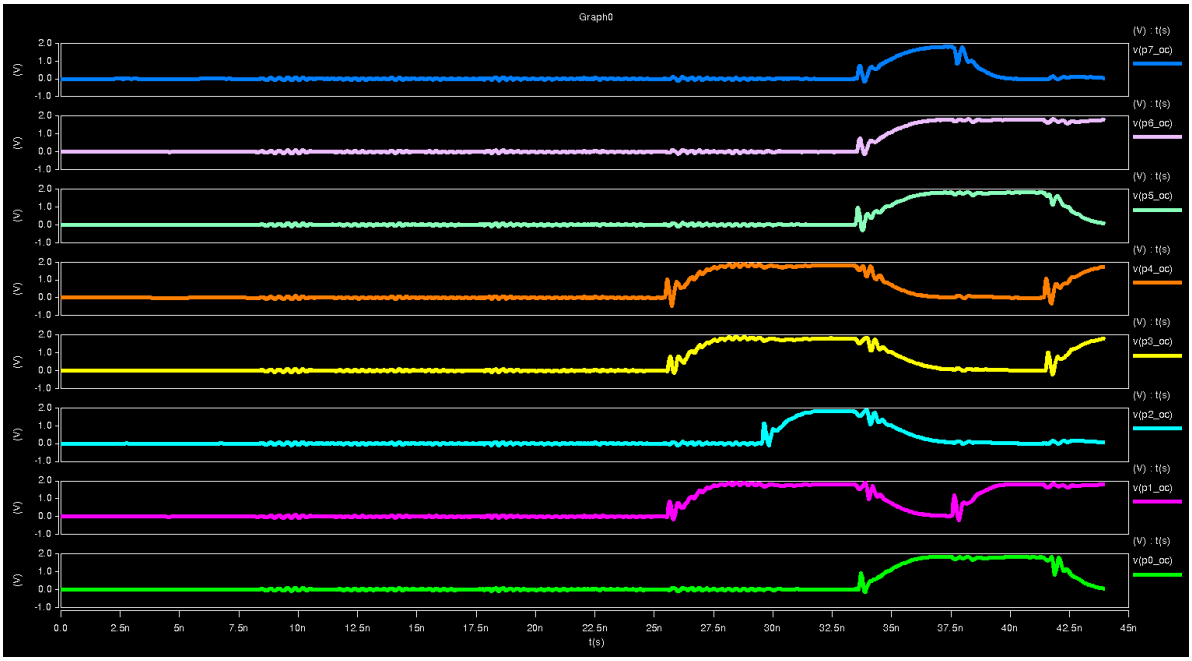

Figure 16: Chip side bond pad output signals for test bench 1 at the TT process corner.

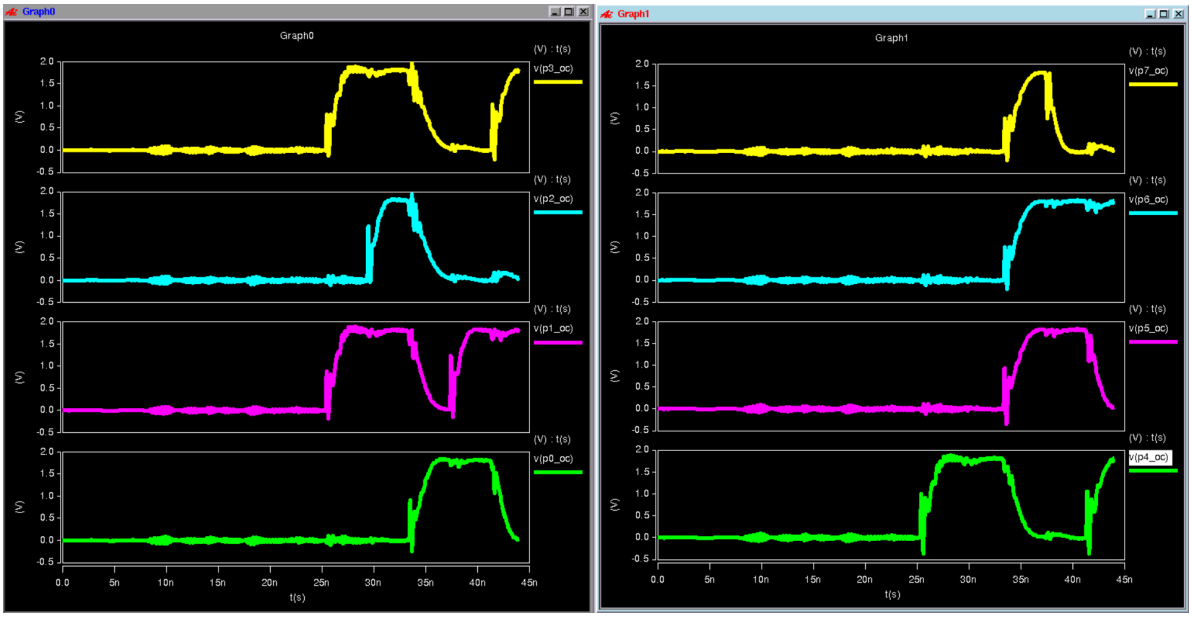

Figure 17: Chip side bond pad output signals for test bench 1 at the FF process corner.

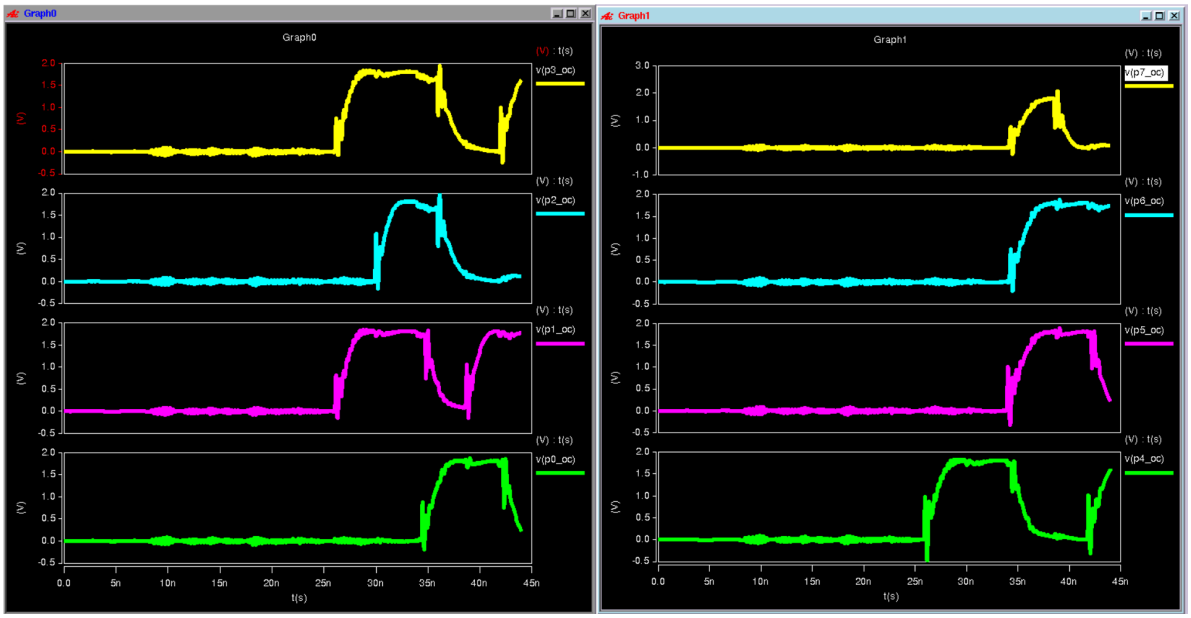

Figure 18: Chip side bond pad output signals for test bench 1 at the SS process corner.

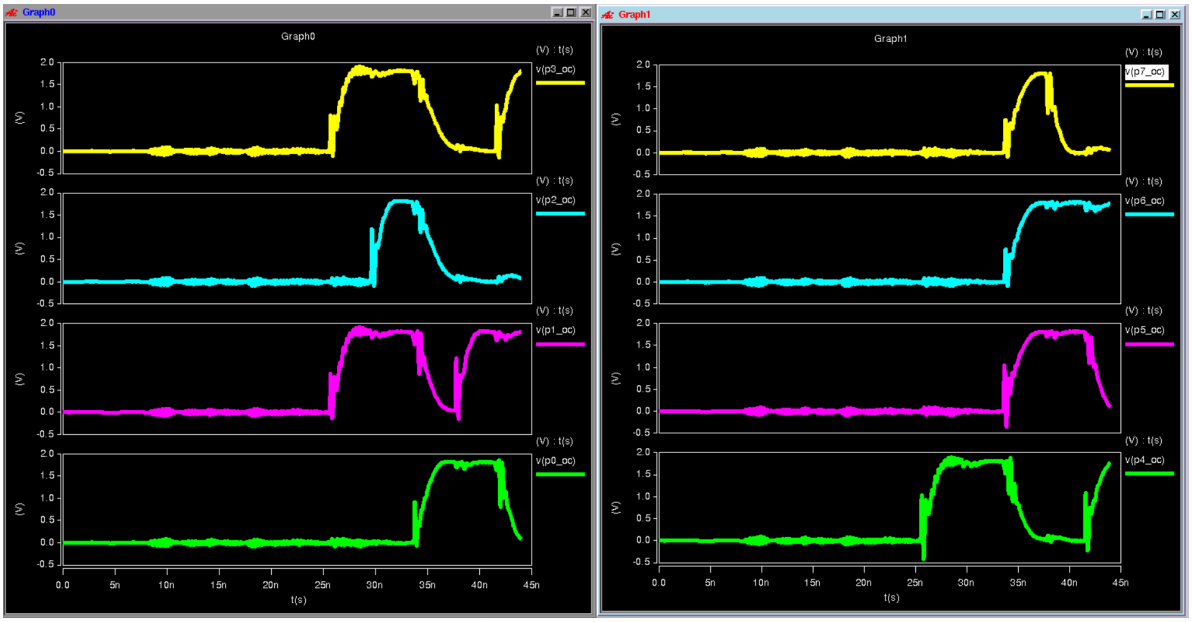

Figure 19: Chip side bond pad output signals for test bench 1 at the SF process corner.

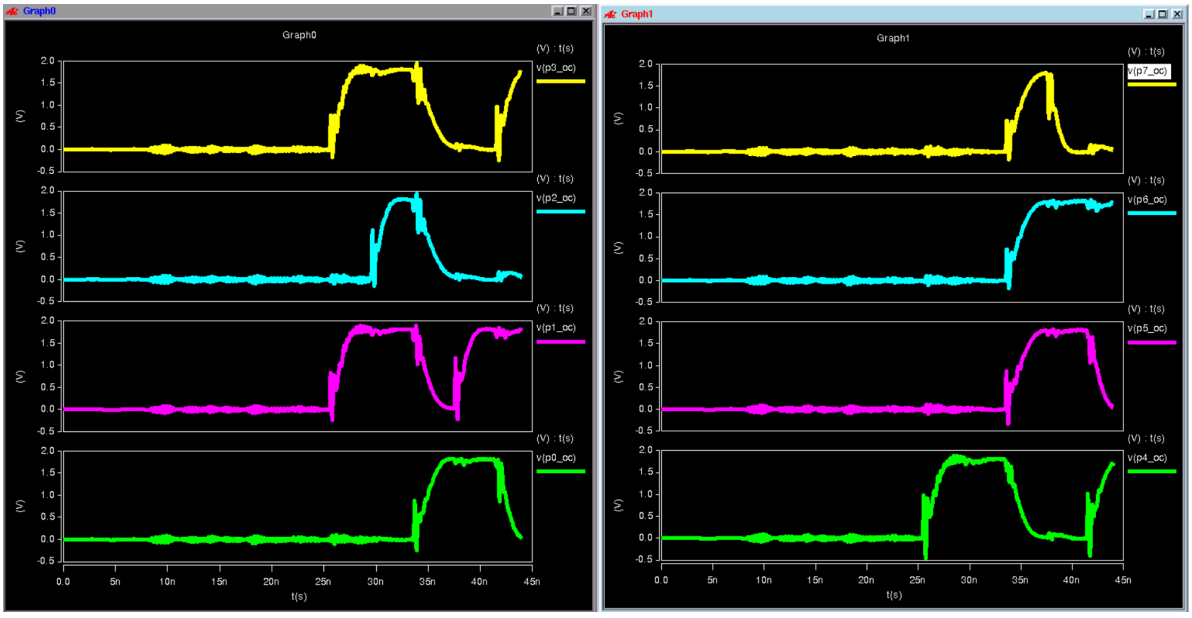

Figure 20: Chip side bond pad output signals for test bench 1 at the FS process corner.

Figures 21 - 25 show the multiplier outputs for test bench 1 at the package output pins (p0 - p7) across all 5 process corners.

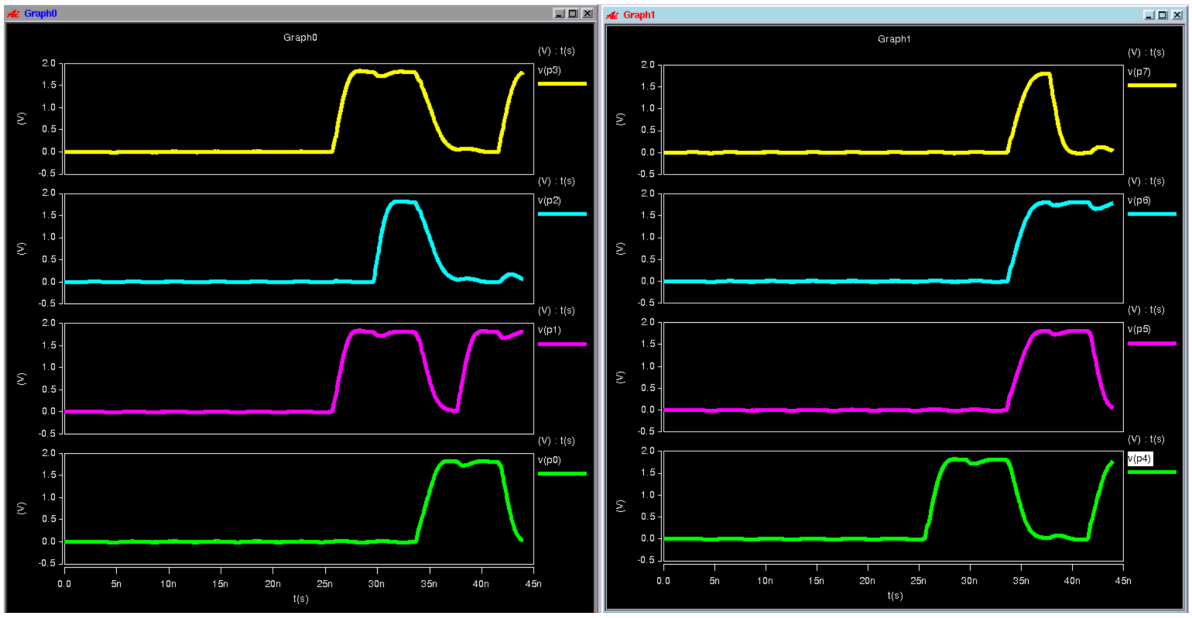

Figure 21: Package output pin signals for test bench 1 at the TT process corner.

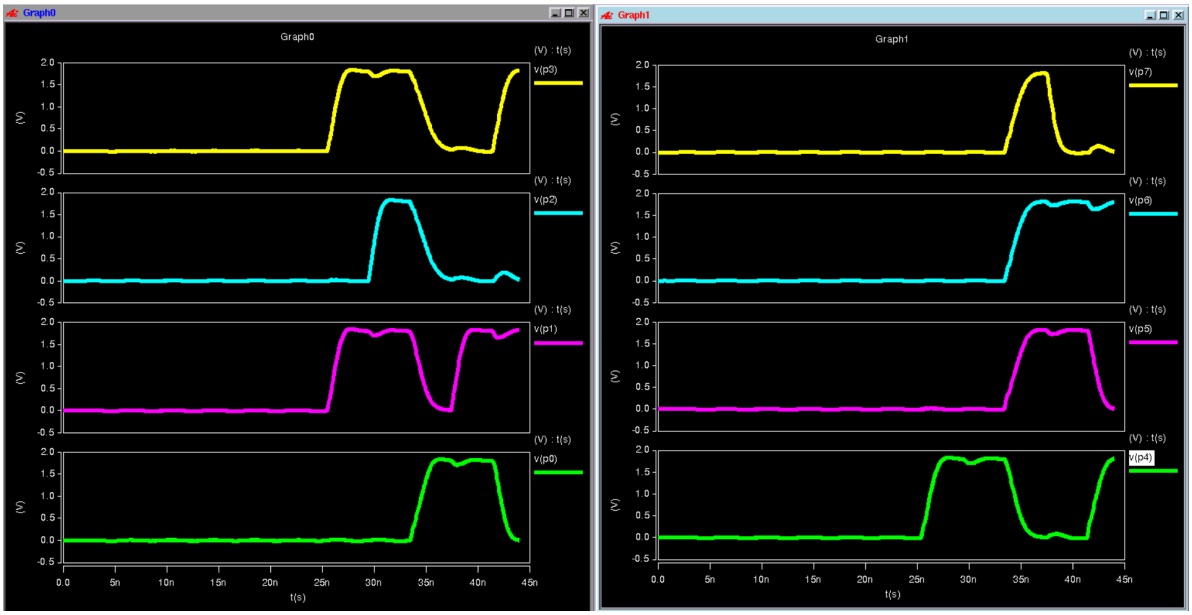

Figure 22: Package output pin signals for test bench 1 at the FF process corner.

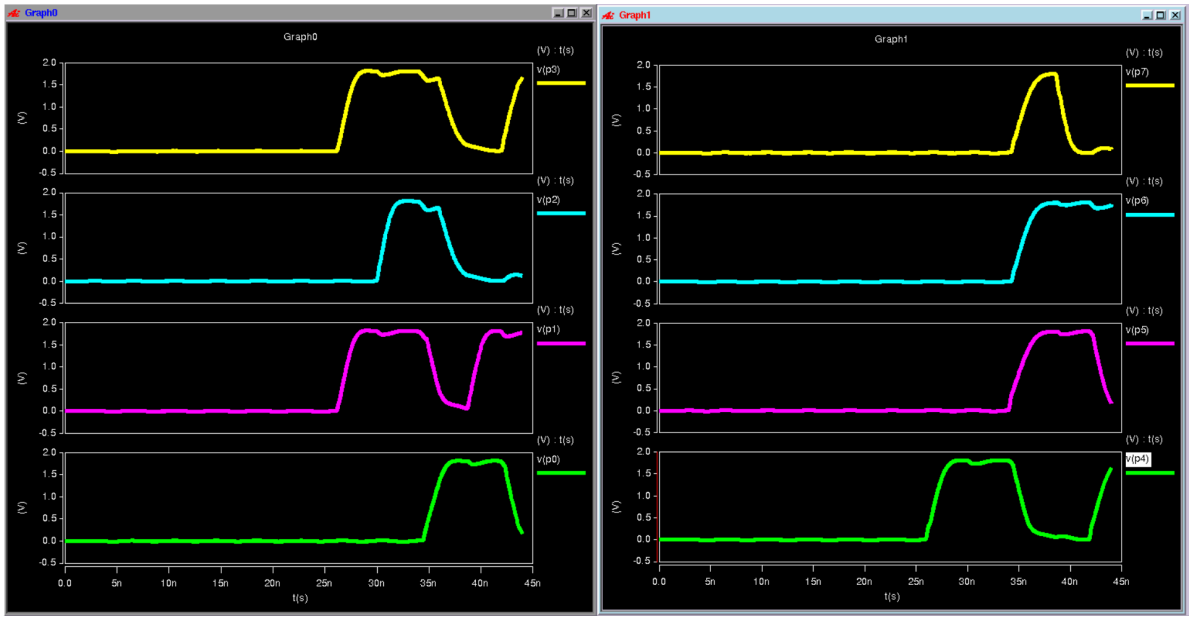

Figure 23: Package output pin signals for test bench 1 at the SS process corner.

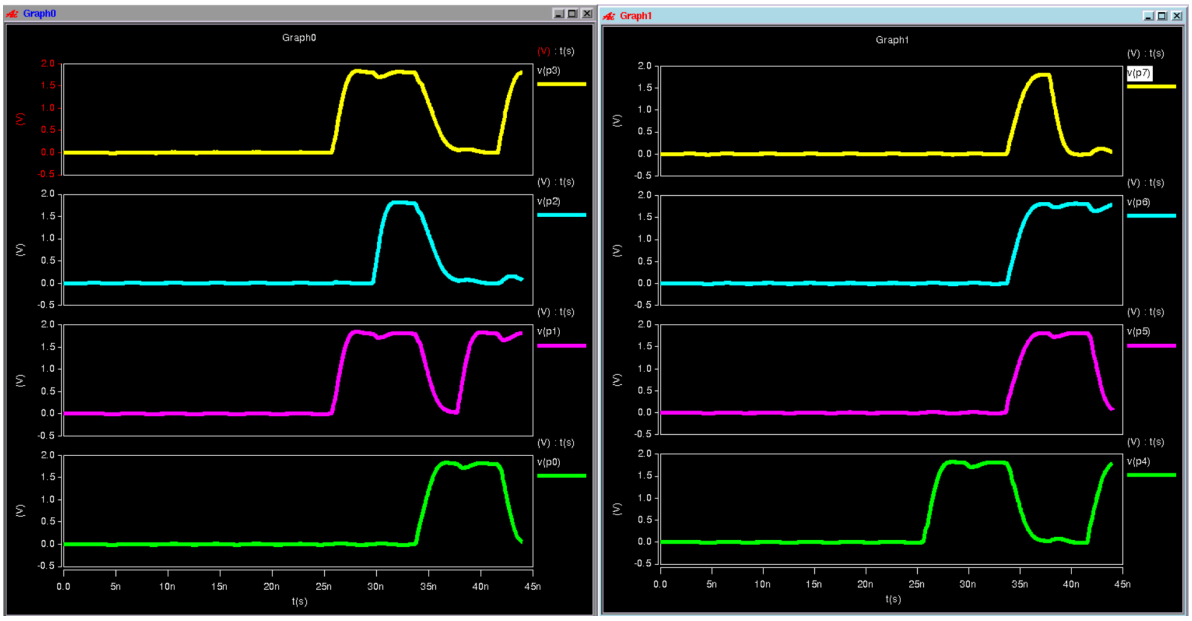

Figure 24: Package output pin signals for test bench 1 at the SF process corner.

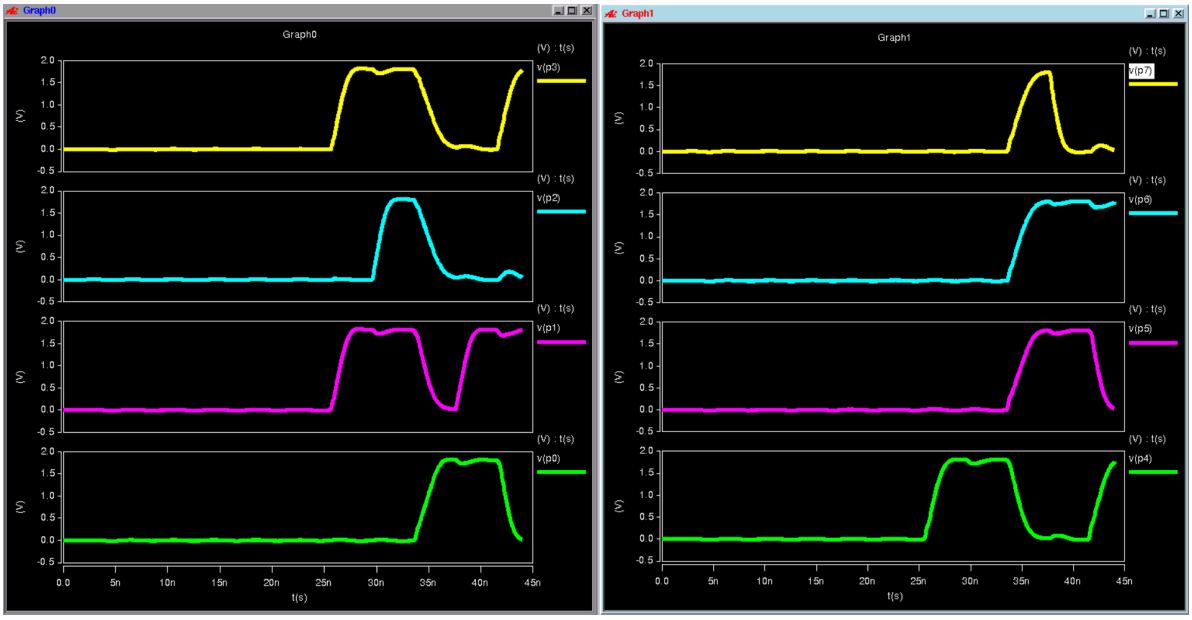

Figure 25: Package output pin signals for test bench 1 at the FS process corner.

Figures 26 - 30 show the multiplier outputs for test bench 2 at the chip side bond pads across all 5 process corners.

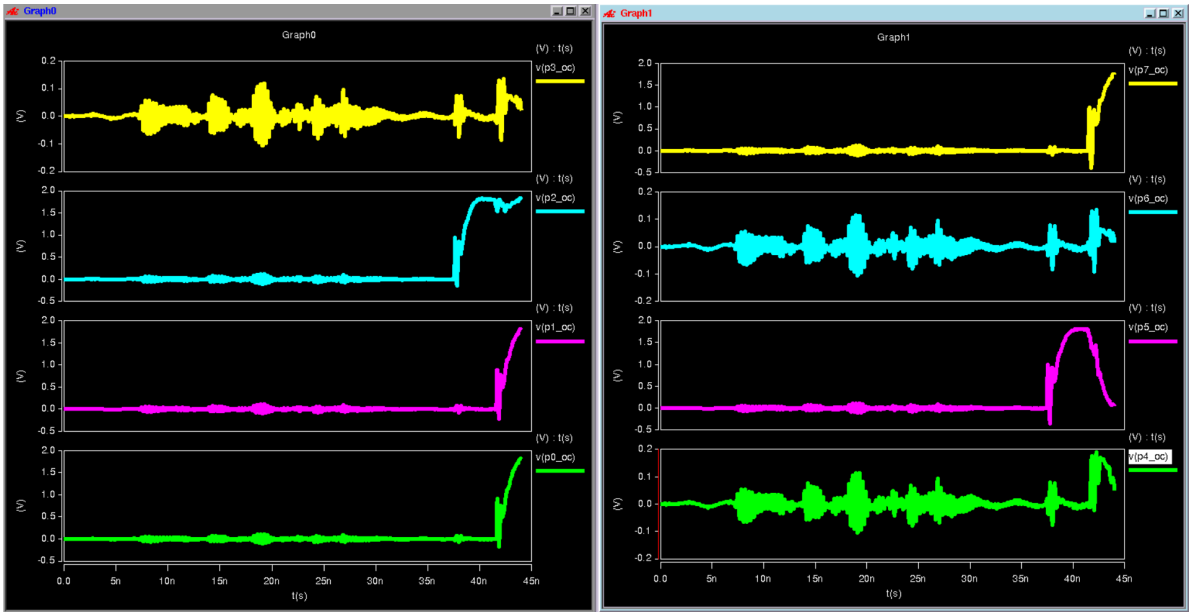

Figure 26: Chip side bond pad output signals for test bench 2 at the TT process corner.

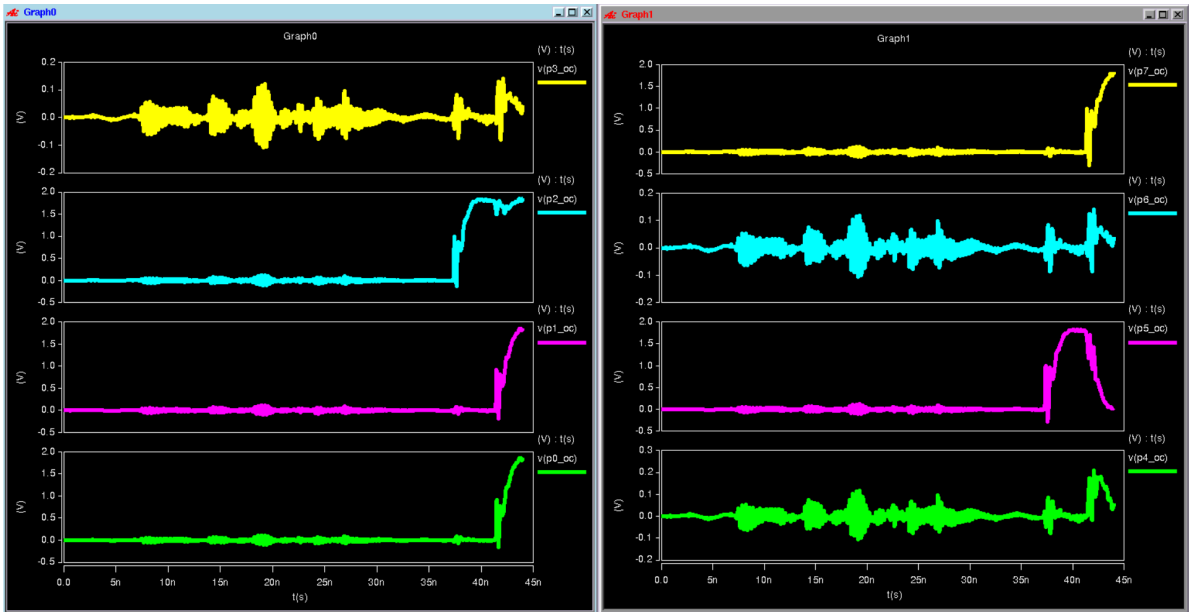

Figure 27: Chip side bond pad output signals for test bench 2 at the FF process corner.

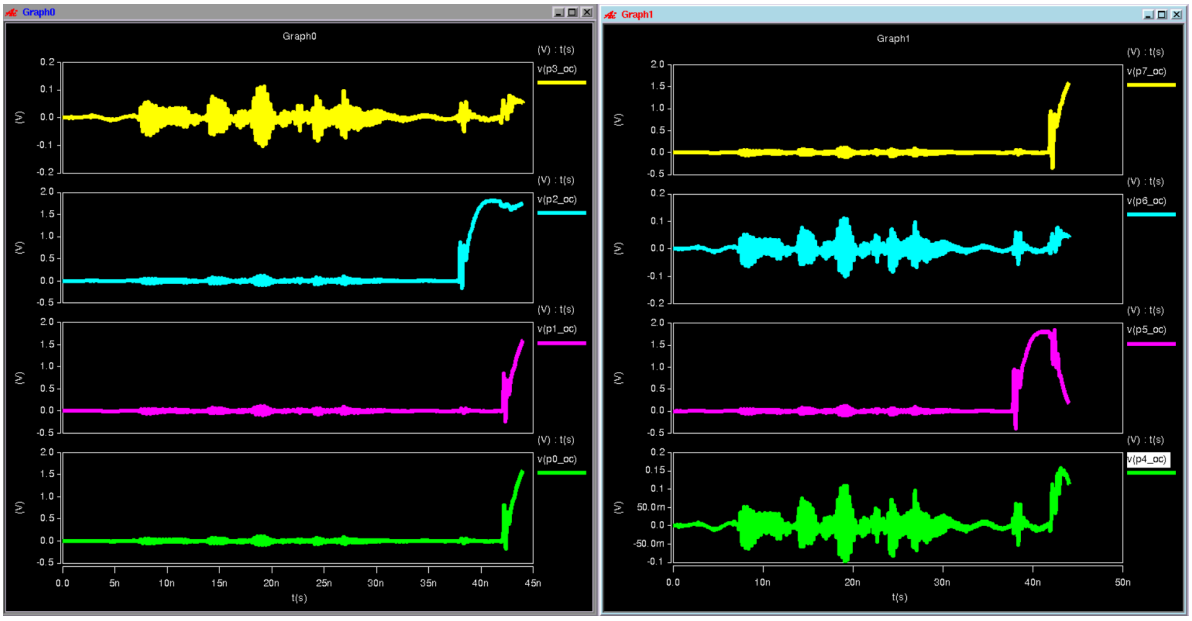

Figure 28: Chip side bond pad output signals for test bench 2 at the SS process corner.

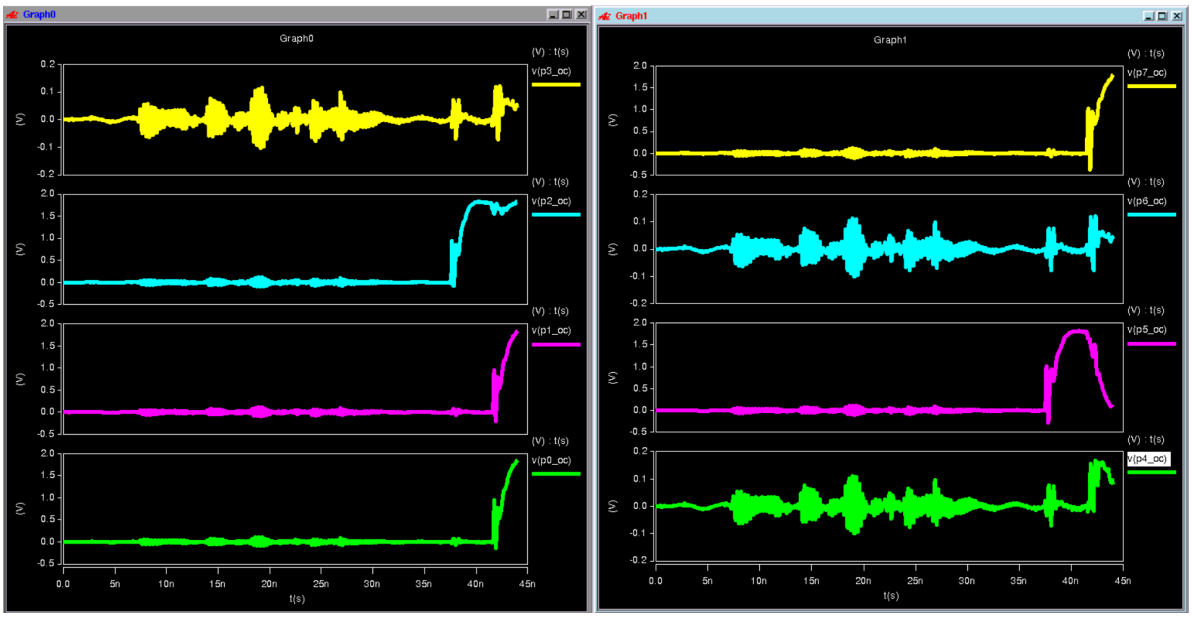

Figure 29: Chip side bond pad output signals for test bench 2 at the SF process corner.

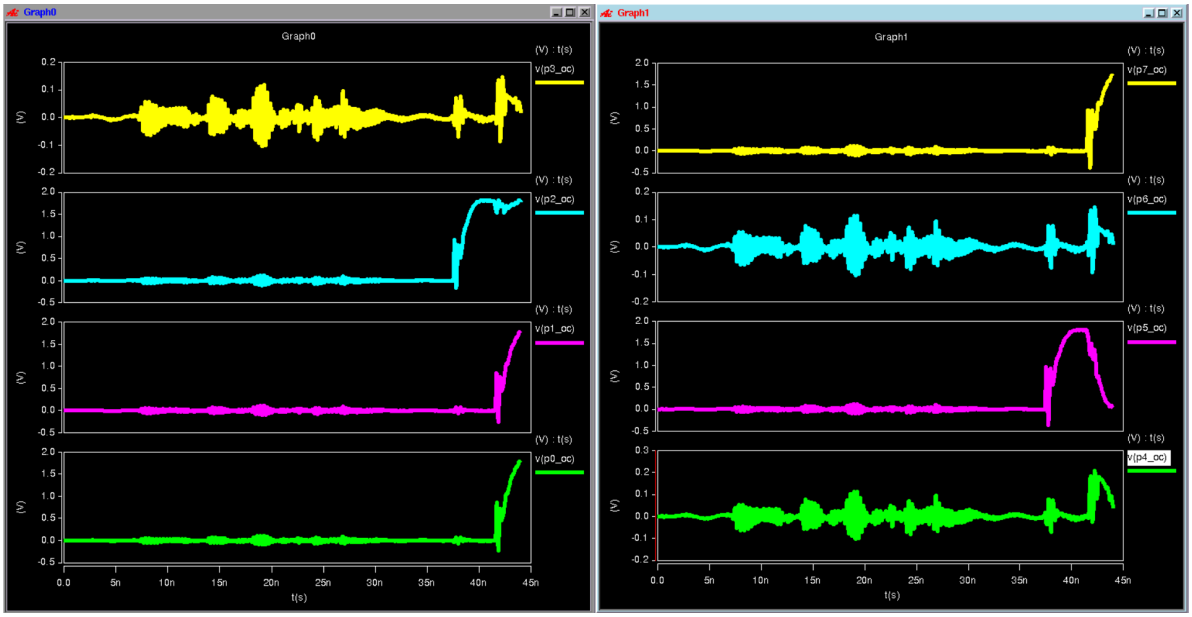

Figure 30: Chip side bond pad output signals for test bench 2 at the FS process corner.

Figures 31 - 35 show the multiplier outputs for test bench 1 at the package output pins across all 5 process corners.

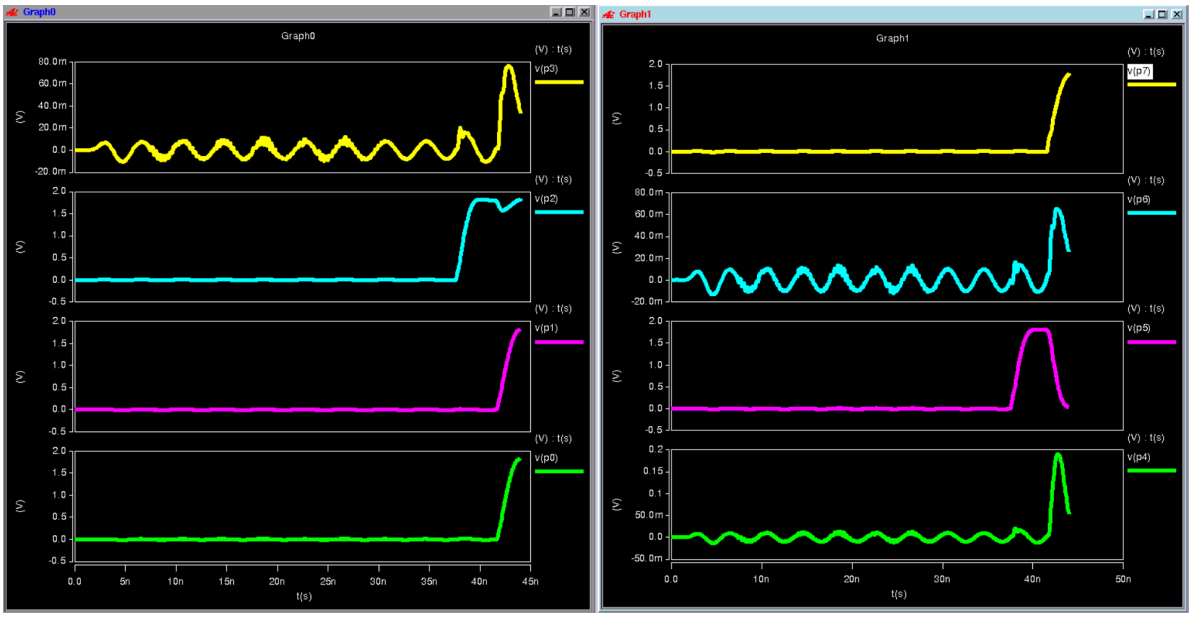

Figure 31: Package output pin signals for test bench 2 at the TT process corner.

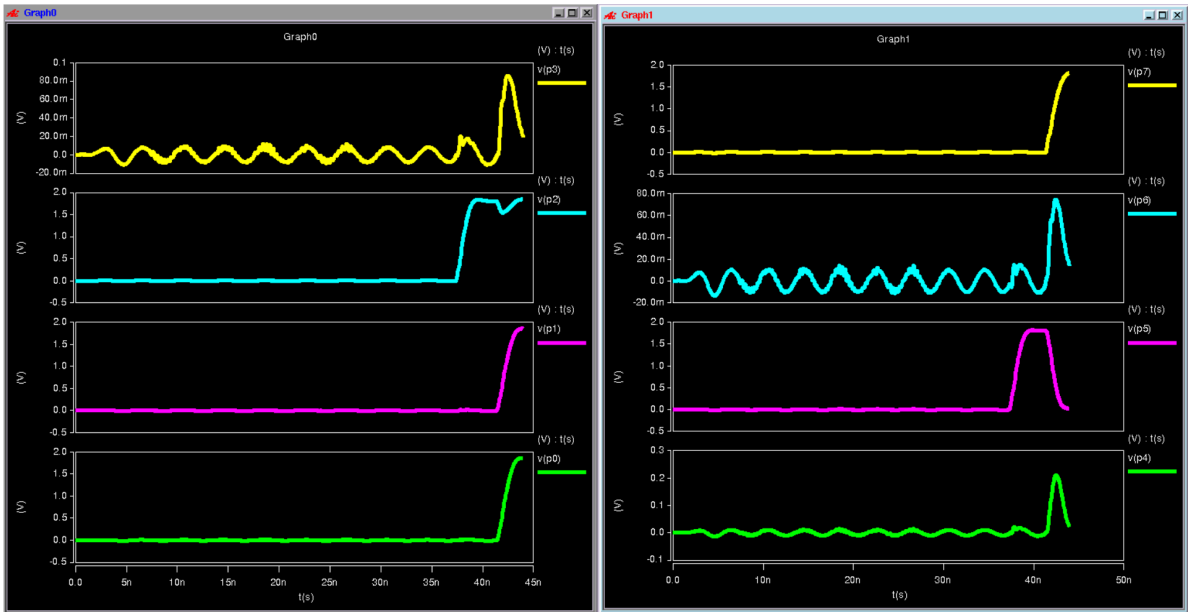

Figure 32: Package output pin signals for test bench 2 at the FF process corner.

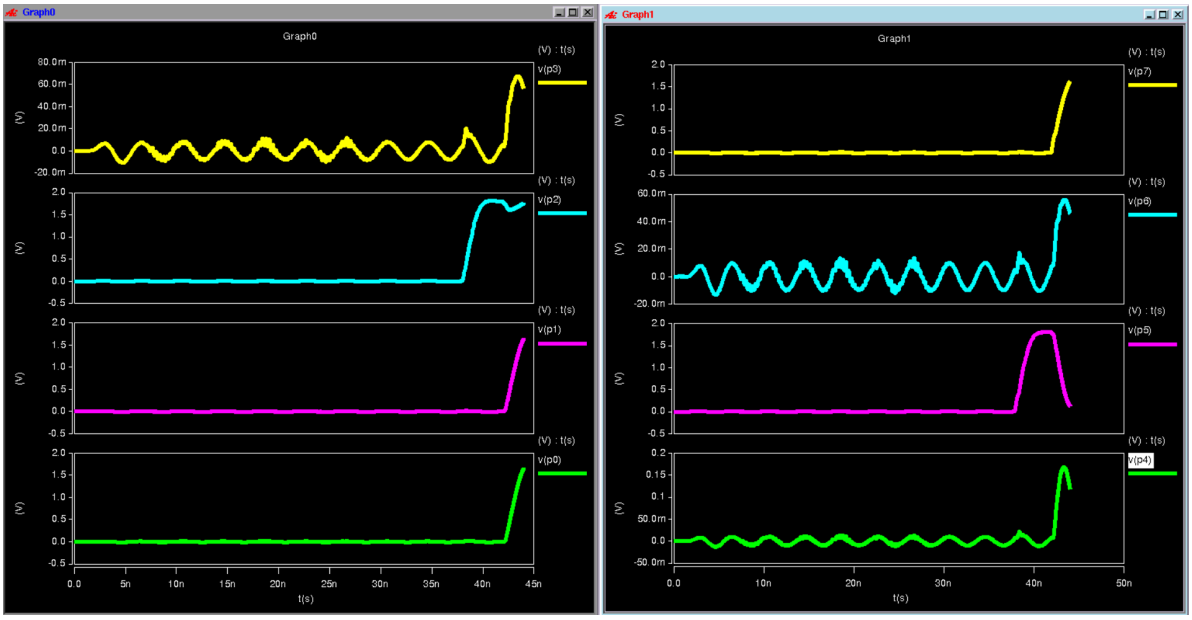

Figure 33: Package output pin signals for test bench 2 at the SS process corner.

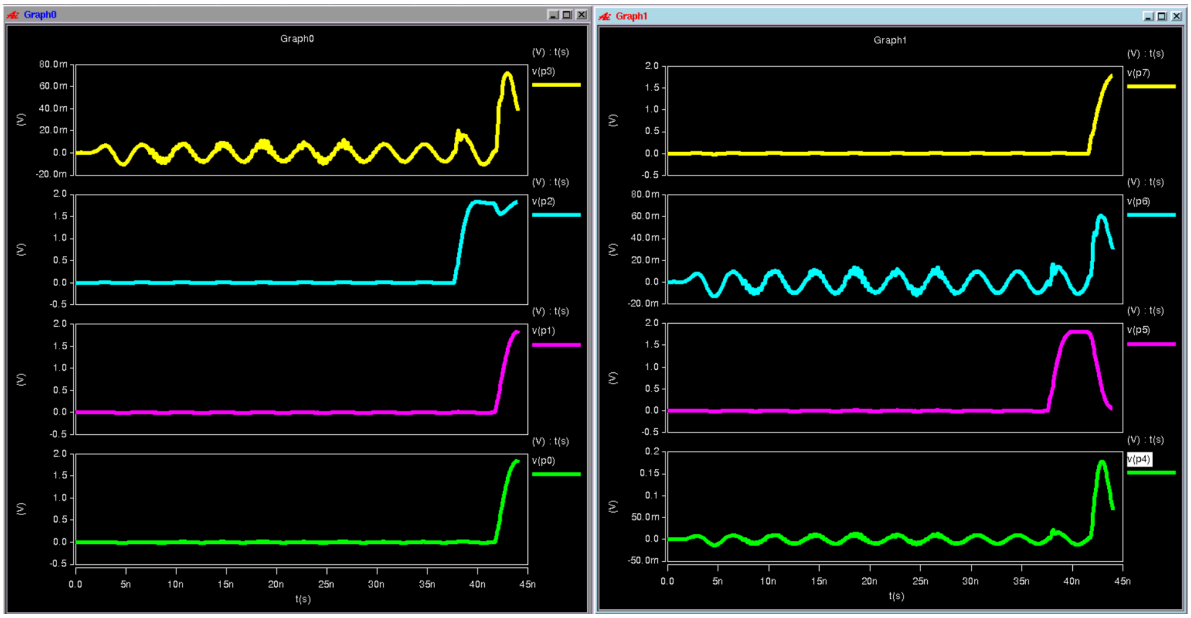

Figure 34: Package output pin signals for test bench 2 at the SF process corner.

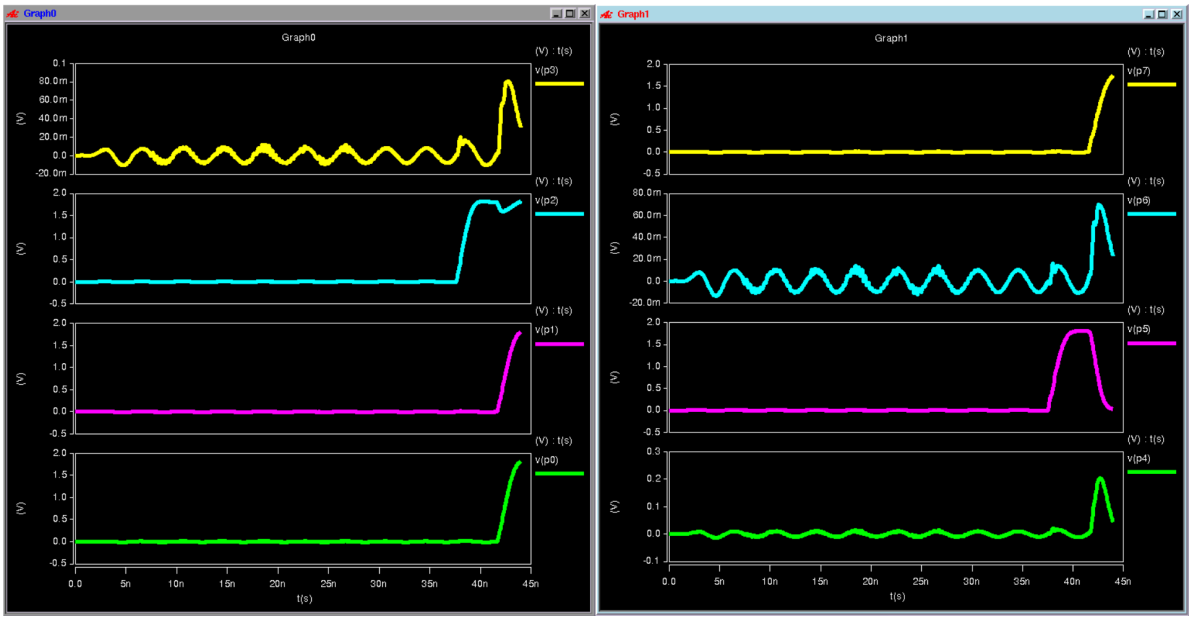

Figure 35: Package output pin signals for test bench 2 at the FS process corner.

As shown in the above figures, the output signals are able to swing between 0.1VDDIO and 0.9VDDIO in all 5 process corners when V_{DD} is at 1.04 V. 1.04 V was chosen as the operating voltage for V_{DD} because it was the lowest value of V_{DD} for which the output signals were able to achieve this swing, thus minimizing the power consumption of the circuit while maintaining its intended functionality. The actual power dissipation metrics are discussed in section 6.2.

6.2 Power Dissipation

The power dissipation metrics of the circuit were measured using test bench 2, as stated in the project guidelines. The three metrics measured in the test bench are shown in Figure 36.

```
y
 avg_pwr_vdd= -20.6737m  from=   7.4000n     to=  42.0000n
 avg_pwr_vddio=  -5.0435m  from=   7.4000n     to=  42.0000n
 avg_pwr_iload=  19.7184m  from=   7.4000n     to=  42.0000n
 corrected_pwr_vdd=-955.2953u
```

Figure 36: Power measurements from test bench 2 under the TT process corner.

Figure 36 indicates that the multiplier dissipates 955.2953 uW when $V_{DD} = 1.04$ V, which falls in line with the trend predicted in Figure 7.

7 Layout

7.1 Top-Level Layout Overview

Figure 37 shows the top-level layout of the implemented 8-bit multiplier. The view has the upper-level power and ground mesh removed, as required by the project specification, so that device placement and signal routing are clearly visible. All major subblocks of the design are identifiable in this view, including the multiplier core, the output register and associated level shifters, and the final superbuffer bank.

Figure 38 provides the same top-level view with labeled bounding boxes highlighting the three principal regions of the design: (1) the three-stage multiplier array, (2) the output registers and DCVSL level shifters, and (3) the 1.8 V output superbuffers. This helps illustrate the physical dataflow from the 1.8 V input domain through the core logic and back to the 1.8 V output domain.

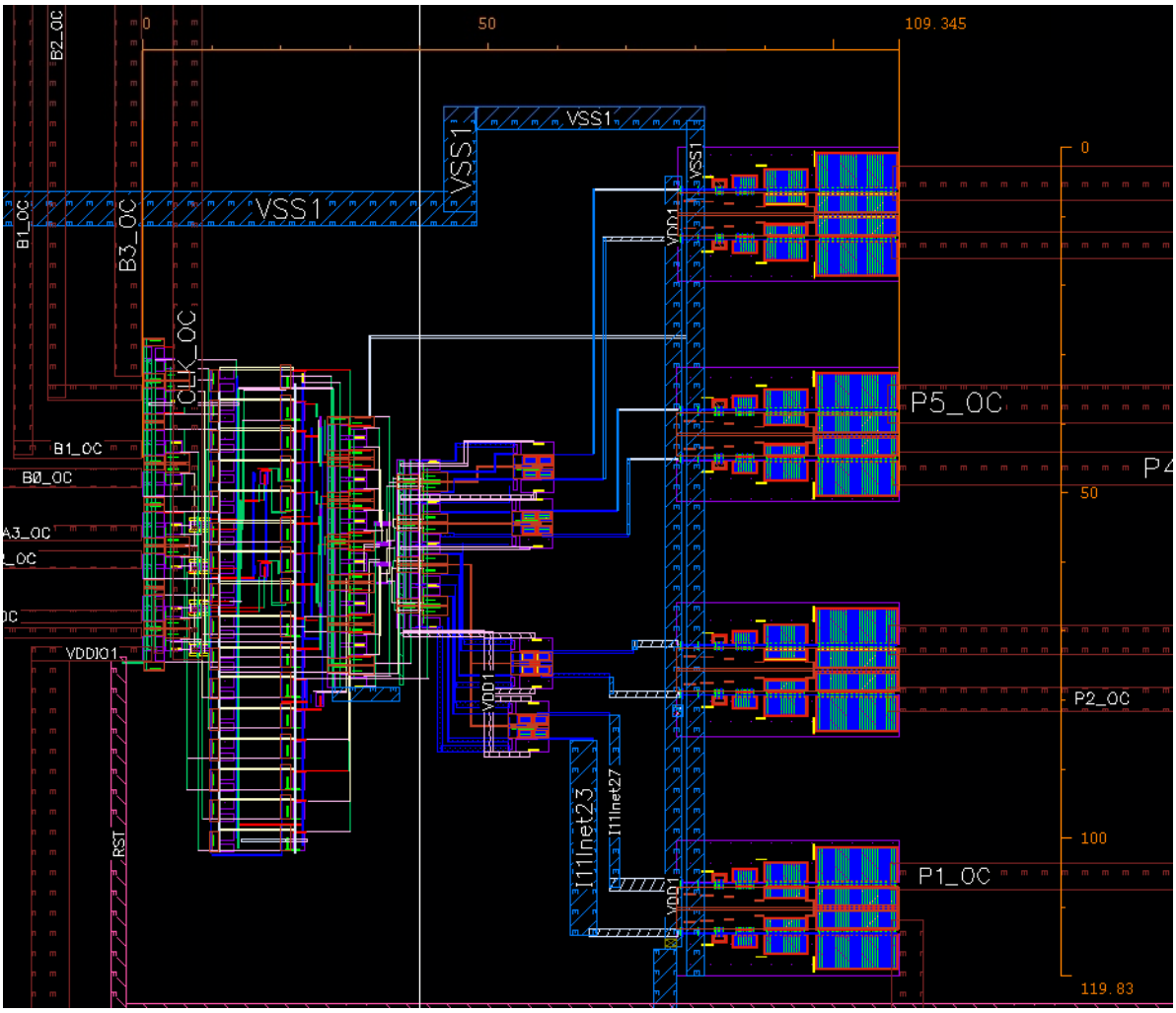

Figure 37: Top-level layout of the multiplier with upper-level P/G mesh removed.

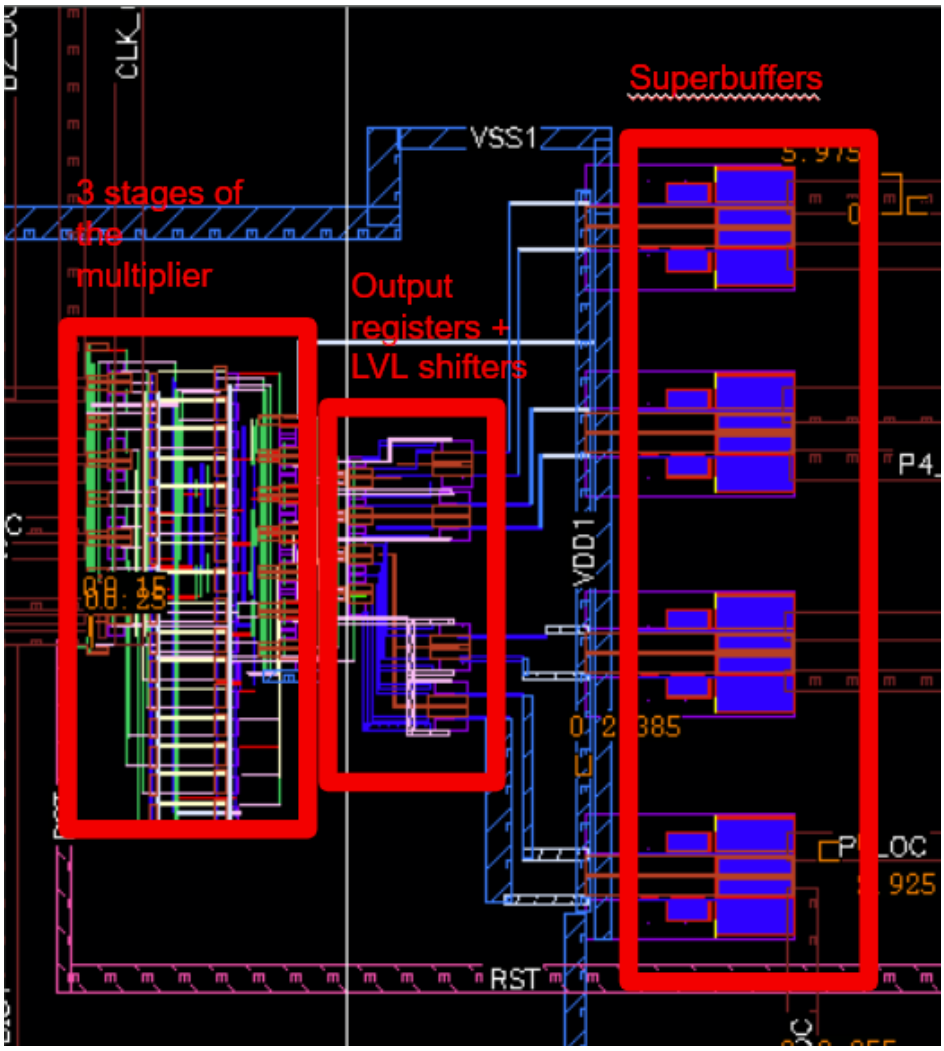

Figure 38: Top-level layout with major functional blocks highlighted and labeled.

7.2 Individual Block Layouts

This section shows the layout views of all custom-designed blocks used in the multiplier. Each layout is exported with upper-level power and ground metals removed, leaving only signal-carrying layers visible (M1–M3). The figures highlight the compact placement and local routing within each cell.

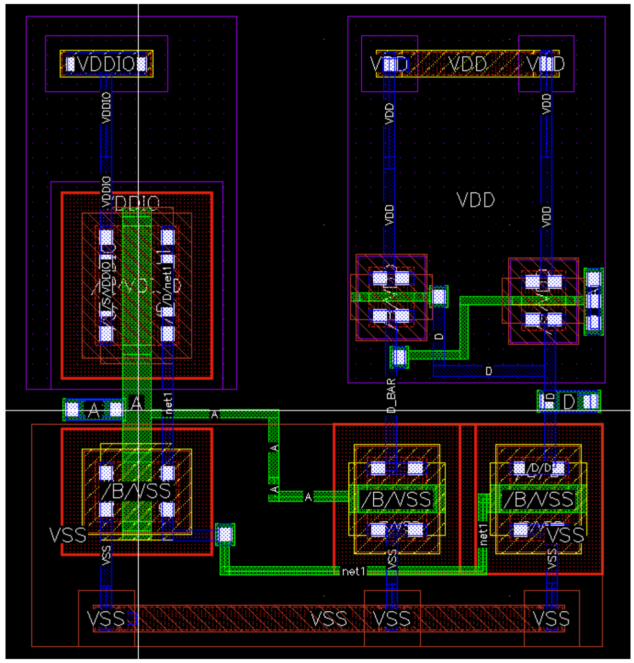

Figure 39: Layout of the 1.8 V → V_{DD} DCVSL level shifter.

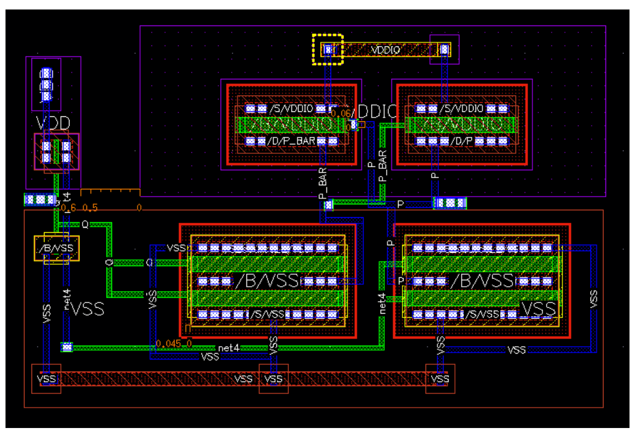

Figure 40: Layout of the V_{DD} → 1.8 V DCVSL level shifter.

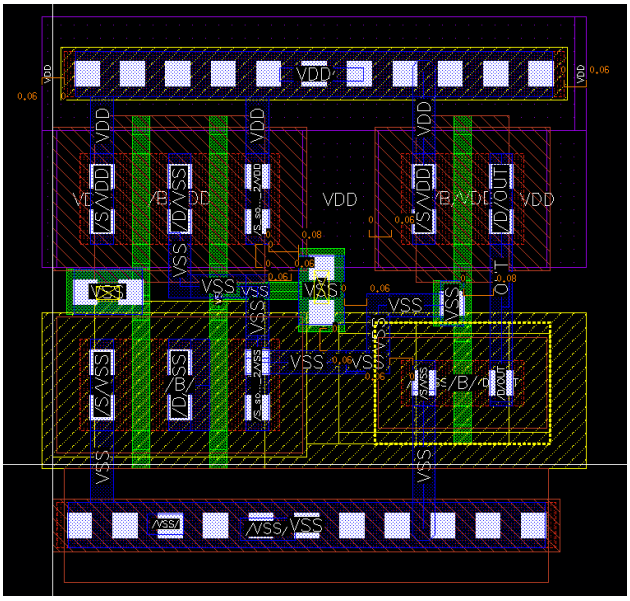

Figure 41: Layout of the AND gate (NAND + inverter structure)

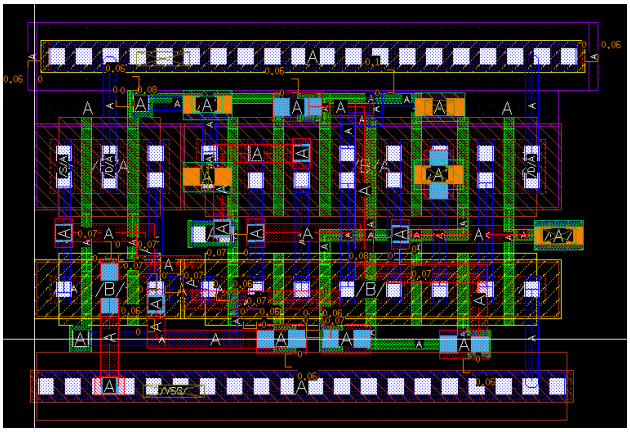

Figure 42: Layout of the full adder used in the CRA chain.

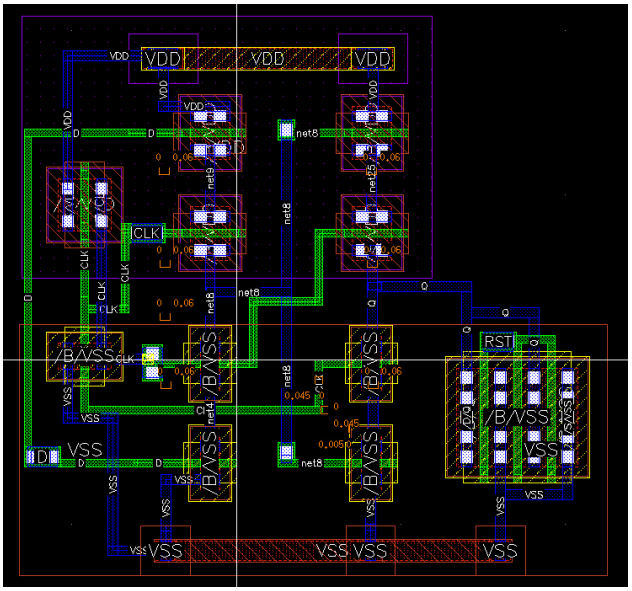

Figure 43: Layout of the C2MOS master–slave register.

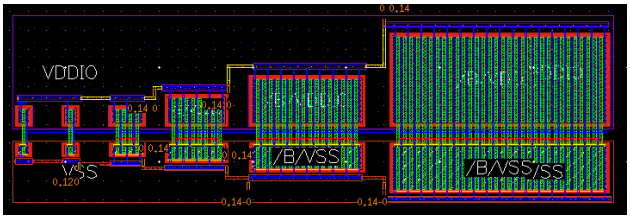

Figure 44: Layout of the tapered output superbuffer.

7.3 Total Area of the Circuitry

The total area of the implemented multiplier was measured from the final top-level layout bounding box. The dimensions of the core (including the multiplier array, output registers, level shifters, and output superbuffers) are:

- **Width:** 109.345 μm
- **Height:** 119.83 μm

The resulting silicon area is:

$$A_{\text{total}} = 109.345\ \mu\text{m} \times 119.83\ \mu\text{m} = 13{,}102.81\ \mu\text{m}^2.$$

This area excludes the upper-level power and ground mesh, as required by the project specification, and reflects only the active signal-routing and device layers.

8 Conclusions

This report demonstrates a successful implementation of a 4-bit Wallace Tree multiplier, utilizing techniques such as pipelining and V_{DD} reduction to reduce the multiplier's power consumption. Pipelining proved to be essential in the reduction of power consumption, as the extra delay margins provided by the technique allowed the circuit to meet timing requirements even as V_{DD} was scaled down. The use of non-static CMOS logic families, such as pass-transistor logic, also aided in the reduction, as it allowed for the usage of fewer transistors than static CMOS, which in turn reduced delay and increased the delay margins each pipeline stage could work with.

However, there are also aspects of the design that could be improved. For example, while most of the circuit used minimum-sized transistors in an attempt to reduce the capacitance driven by V_{DD}, there are likely different sizing combinations that could have worked to bring power consumption even lower while maintaining proper functionality. An example of this can be seen in the full adder circuit, where simulations show that the unloaded delay of the full adder was essentially independent of the internal transistor sizes. As such, the transistor sizes likely could have been increased to better drive any following loads, which would in turn decrease the full adder delay and allow for a slightly larger reduction in V_{DD}. The design also likely could have been improved by implementing other low power design techniques, like clock gating or sleep states. Given more time, the team would have experimented with those techniques to study their impact.

9 Team Effort

The initial division of labor was devised such that Cristian and Madhav would work on the design, simulation, and layout of the half and full adder cells, while Wei and Kriti were in charge of the design, simulation, and layout of the registers, level shifters, and AND gates. Unfortunately, despite multiple attempts to reach Mai Xu, the team was unable to contact them between the first week of the project assignment and the week before the project's due date, so the work was divided under the assumption that Mai could not contribute. From there, Cristian, Wei, and Madhav worked on connecting the cells to form the entire multiplier circuit and integrating it with the PDN. Then, Cristian, Wei, Madhav, and Kriti all contributed to the writing of this report.