```
# EchoPulse_Benchmark_Script.py (Document C5)

import time
import random

# 1. Simulation Constants
PLATFORM_PROFILES = {
    "M0+": 16_000_000,  # Hz
    "M4F": 80_000_000,
    "RV64": 120_000_000,
}

ENCAPS_CYCLES = {
    "M0+": 150_000,
    "M4F": 30_000,
    "RV64": 20_000,
}

DECAPS_CYCLES = {
    "M0+": 130_000,
    "M4F": 25_000,
    "RV64": 18_000,
}

SHA3_CYCLES = {
    "M0+": 20_000,
    "M4F": 4_000,  # Adjusted for plausibility
```

```
    "RV64": 3_000,
}

TRANSITION_COST_PER_SYMBOL = 1  # Simulated cycles
MUTATION_ROW_COST = 64  # Simulated cycles
MUTATION_FULL_COST = {"M0+": 16_000, "M4F": 3_000, "RV64": 2_000} # Adjusted for plausibility

RAM_USAGE_ESTIMATE = 8420  # Bytes

def simulate_delay(cycles, platform):
    if PLATFORM_PROFILES[platform] > 0:
        return cycles / PLATFORM_PROFILES[platform]
    return 0

def simulate_encaps(platform):
    return simulate_delay(ENCAPS_CYCLES[platform], platform)

def simulate_decaps(platform):
    return simulate_delay(DECAPS_CYCLES[platform], platform)

def simulate_sha3(platform):
    return simulate_delay(SHA3_CYCLES[platform], platform)

def simulate_transition(num_symbols, platform):
    return simulate_delay(TRANSITION_COST_PER_SYMBOL * num_symbols, platform)

def simulate_mutation_row(platform):
```

```
    return simulate_delay(MUTATION_ROW_COST, platform)


def simulate_mutation_full(platform):
    return simulate_delay(MUTATION_FULL_COST[platform], platform)


if __name__ == "__main__":
    NUM_CYCLES = 100
    MUTATION_INTERVAL = 10

    for platform in PLATFORM_PROFILES:
        total_encaps_time = 0
        total_decaps_time = 0
        total_mutation_time = 0
        mutation_count = 0

        start_time = time.perf_counter()

        for i in range(NUM_CYCLES):
            encaps_time = simulate_encaps(platform)
            total_encaps_time += encaps_time

            decaps_time = simulate_decaps(platform)
            total_decaps_time += decaps_time

            if (i + 1) % MUTATION_INTERVAL == 0:
                mutation_time = simulate_mutation_row(platform)
                total_mutation_time += mutation_time
```

```
            mutation_count += 1

    end_time = time.perf_counter()
    elapsed_time = end_time - start_time

    avg_encaps_time = (total_encaps_time * 1_000_000) / NUM_CYCLES if NUM_CYCLES > 0 else 0
    avg_decaps_time = (total_decaps_time * 1_000_000) / NUM_CYCLES if NUM_CYCLES > 0 else 0
    avg_mutation_time = (total_mutation_time * 1_000_000) / mutation_count if mutation_count > 0 else 0

    print(f"--- Performance Simulation on {platform} ---")
    print(f"Average Encapsulation Time: {avg_encaps_time:.2f} µs")
    print(f"Average Decapsulation Time: {avg_decaps_time:.2f} µs")
    print(f"Average Mutation (Row) Time: {avg_mutation_time:.2f} µs")
    print(f"Total Mutation Count: {mutation_count}")
    print(f"Estimated RAM Usage: {RAM_USAGE_ESTIMATE} Bytes")
    print(f"Total Simulation Time: {elapsed_time:.2f} seconds")
    print("-" * 40)

# Optional: Matplotlib Bar Chart (requires installation: pip install matplotlib)
try:
    import matplotlib.pyplot as plt

    platforms = list(PLATFORM_PROFILES.keys())
    avg_encaps = [ENCAPS_CYCLES[p] / PLATFORM_PROFILES[p] * 1_000_000 for p in platforms]
    avg_decaps = [DECAPS_CYCLES[p] / PLATFORM_PROFILES[p] * 1_000_000 for p in platforms]
    avg_mutate = [MUTATION_ROW_COST / PLATFORM_PROFILES[p] * 1_000_000 for p in platforms]
```

```
    bar_width = 0.2
    index = range(len(platforms))

    fig, ax = plt.subplots()
    bar1 = ax.bar([i - bar_width for i in index], avg_encaps, bar_width, label='Encapsulation')
    bar2 = ax.bar(index, avg_decaps, bar_width, label='Decapsulation')
    bar3 = ax.bar([i + bar_width for i in index], avg_mutate, bar_width, label='Mutation (Row)')

    ax.set_xlabel('Platform')
    ax.set_ylabel('Average Time (µs)')
    ax.set_title('EchoPulse Simulated Operation Times')
    ax.set_xticks(index)
    ax.set_xticklabels(platforms)
    ax.legend()
    fig.tight_layout()
    plt.savefig("EchoPulse_Benchmark_Chart.png")
    print("Optional: Generated 'EchoPulse_Benchmark_Chart.png'")

except ImportError:
    print("Optional: Matplotlib not found. Skipping chart generation.")
```