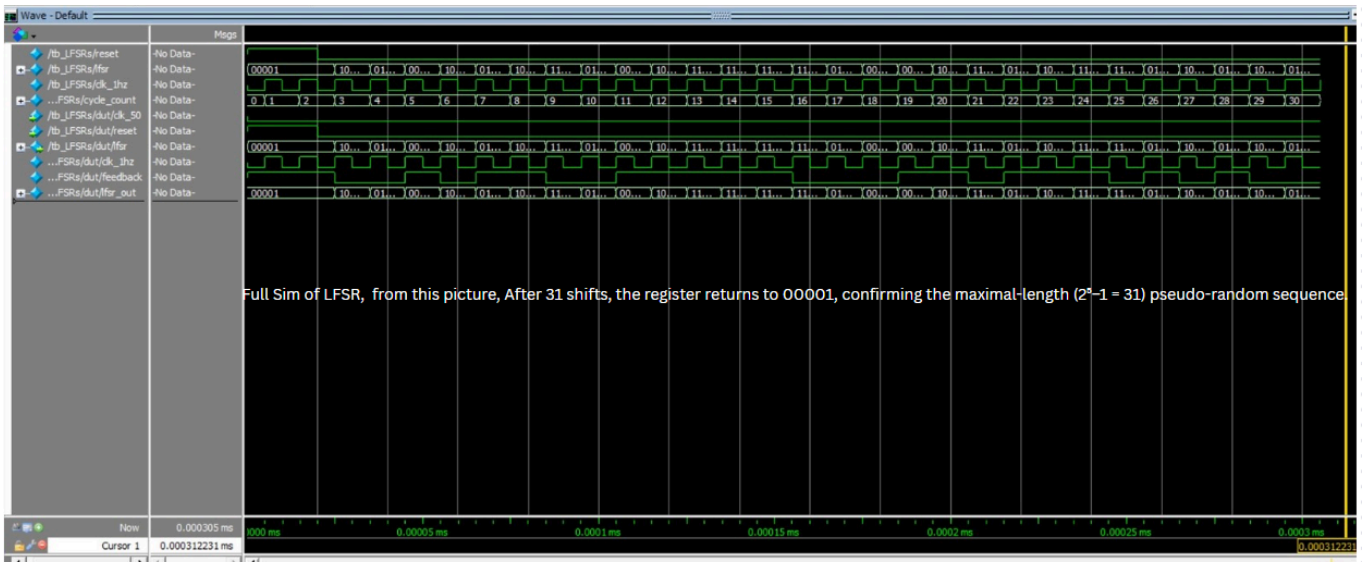
Jacob
Everything Works!

go to “rtl” so get sof file.

LFSR

Simulation:
Full version:



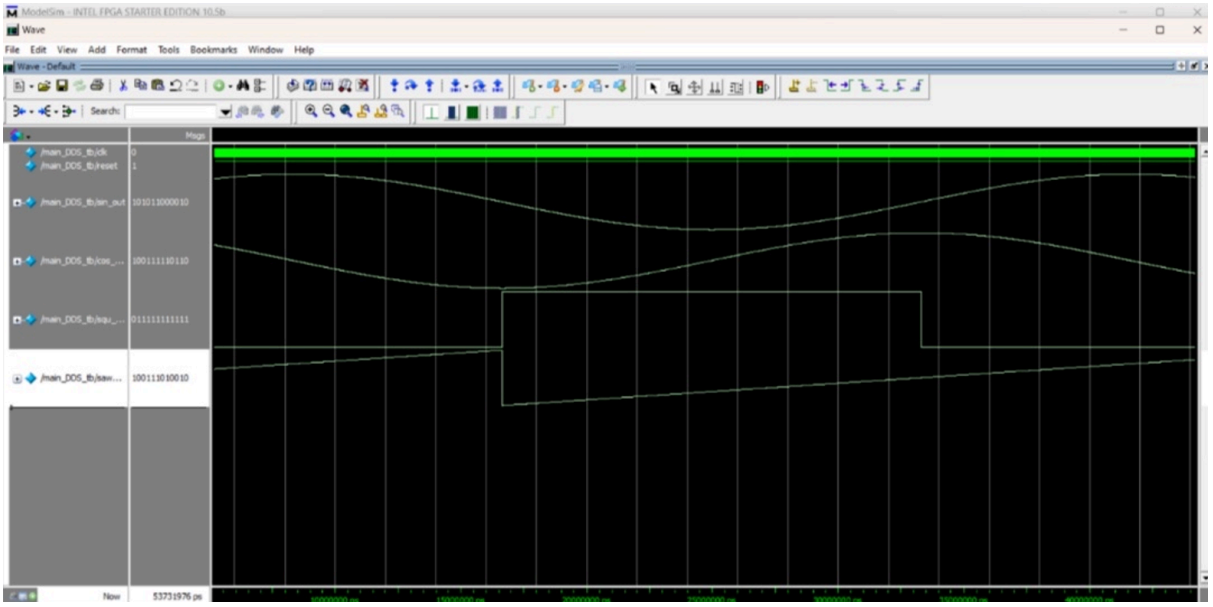
In the simulation waveform, the LFSR begins at state 00001 on the first rising edge of the 1 Hz clock, due to the initialization logic. On each subsequent rising edge, the feedback bit is calculated as the XOR of bit 0 and bit 2 of the current LFSR output. This feedback bit becomes the new most significant bit (MSB), while the remaining

bits shift right. For example, at cycle 1, the state 00001 has a feedback of 1 ⊕ 0 = 1, so the next state becomes 10000. At cycle 2, the state 10000 results in 0 ⊕ 0 = 0, leading to 01000. This process continues: at cycle 3, 01000 produces a feedback of 0 ⊕ 1 = 1, resulting in 10100, and so on. This shifting and feedback process generates a pseudo-random sequence of 5-bit values. After 31 clock cycles, the LFSR returns to its original state of 00001, confirming that it generates a maximal-length sequence of 2^5 – 1 = 31 unique values. The waveform also shows that the internal signal lfsr_out, the output port lfsr, and the feedback bit all behave as expected, making the simulation valid and ready for SignalTap verification.

SignalTap of LSFR:

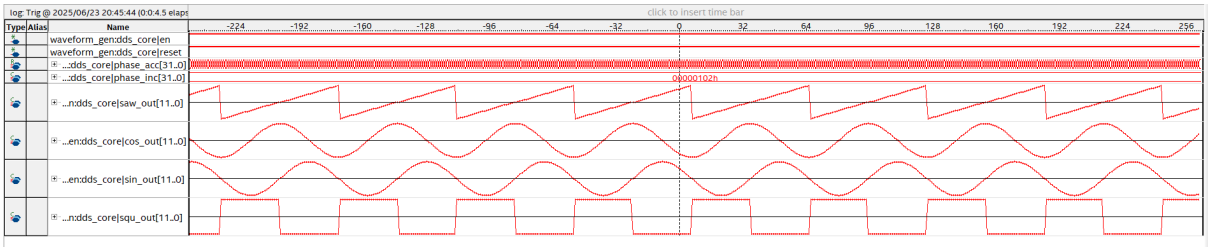
From left to right, the LFSR output clearly cycles through a repeating pseudo-random sequence of 5-bit binary values (e.g., 00001, 10000, 01000, etc.). The signal transitions occur at regular intervals, confirming that the LFSR is clocked properly—likely by a 1 Hz signal as designed. The fact that all values are changing and eventually loop back to the initial value indicates that the LFSR is producing a maximal-length sequence (2^5 – 1 = 31 states).

DDS Simulation:



From the top: sin, cos, squ, saw wave.

SignalTap of DDS:



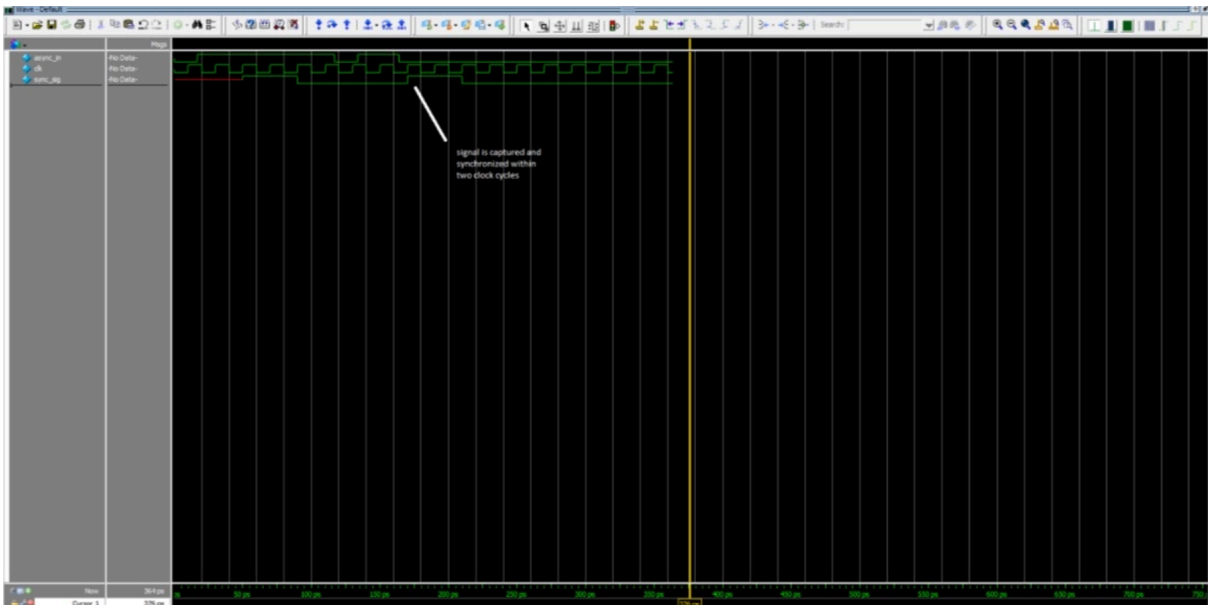
phase_inc[31:0] (top trace) is your 32-bit frequency control word. It remains constant (e.g. 0x00010000), so you're generating a fixed output frequency.

squ_out[11:0] (next trace) is the square-wave output. You can see it hold full-scale for exactly half the phase-accumulator cycle, then switch to zero for the other half—exactly what you expect when you threshold the top bit of the accumulator.

saw_out[11:0] (third trace) is the sawtooth wave. It "ramps" linearly upward one LSB per clock, then instantly wraps back to zero when the accumulator overflows. The sawtooth period matches the square wave's period.

sin_out[11:0] and cos_out[11:0] (bottom two traces) are the sine and cosine lookup-table outputs. They rise and fall in smooth, stepped arcs and are exactly 90° out of phase with each other. Notice each complete sin/cos cycle aligns with one full sawtooth ramp and two square-wave edges.

Clock Synchronizer:



This simulation shows the operation of a two-flip-flop synchronizer used to safely transfer an asynchronous signal (async_in) into the clk domain. The waveform clearly demonstrates that after async_in transitions from low to high, the synchronized output signal (sync_sig) updates two clock cycles later. This delay is expected and confirms that the signal passed through two flip-flops before appearing at the output, effectively mitigating the risk of metastability. The labeled arrow in the waveform points out the moment of synchronization, validating that the design correctly captures and stabilizes the asynchronous input within two clock cycles. This is a standard and reliable technique for clock domain crossing in digital systems.

SignalTap:

This SignalTap capture verifies the operation of a two-flip-flop synchronizer in hardware. The top trace, async_in, represents an asynchronous input signal that transitions from low to high. The bottom trace, sync_sig, shows the synchronized version of that signal within the 50 MHz clock domain. As expected, sync_sig updates exactly two clock ticks after the change in async_in, confirming that the signal passed through two flip-flops before being registered. This behavior ensures

that any metastability from the asynchronous domain is filtered out before the signal is used in the synchronous logic, making the design safe and reliable for clock domain crossing.