FPGA Direct Digital Synthesis DDS

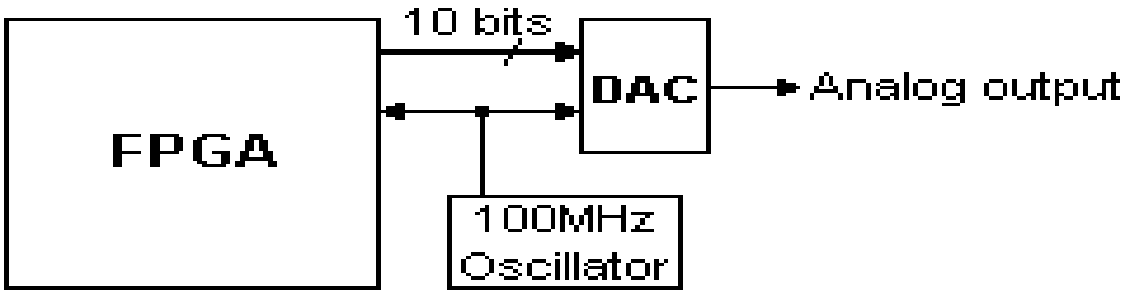
Andrew Hofmaier and Matthew Bajor
andyhof@gmail.com
bajor.matthew@gmail.com

Introduction

- Direct Digital Synthesizer (DDS) Description

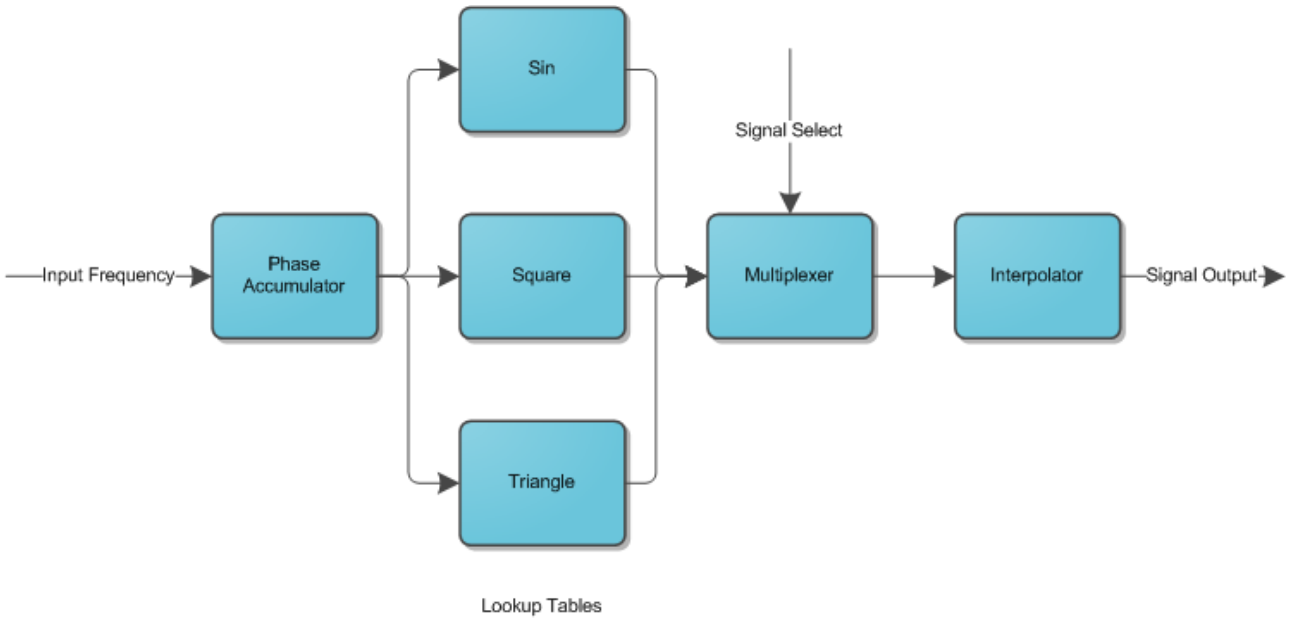
Introduction

- DDS systems allow periodic signal generation
  - Function generators
  - Arbitrary waveform generators



System Components

- Phase Accumulator
- Signal Lookup Tables
- Interpolator



System Components

- Phase Accumulator
- Lookup Tables
- Interpolator
- DAC Output

Phase Accumulator

- Allows the DDS to change frequencies
- Functions as a variable rate counter
  - output <= present value + frequency step
- Frequency value read from the DE-2 switches
  - Supports multipliers from 64x to .125x base frequency at .125x steps
  - Base frequency set at 440 Hz

If the reference frequency is 100 MHz, and the table contains 512 entries, then a complete pass through the table with a phase increment of 1 will take 512 / 100 MHz = 0.00512 ms, so the frequency of the output waveform will be 1/(0.00512 ms) = 195 kHz.

Phase Accumulator Source

```
entity phase_accumulator is
	port  (
		multiplier : in unsigned(9 downto 0);
		clk : in std_logic;
		address : out unsigned(9 downto 0)
	);
end entity;

architecture rtl of phase_accumulator is

signal counter : unsigned(9 downto 0);

begin
	address <= counter;
	process(clk)
	begin
		if( clk='1' and clk'event )then
			counter <= counter + multiplier;
		end if;
	end process;
end rtl;
```

Clock Splitting

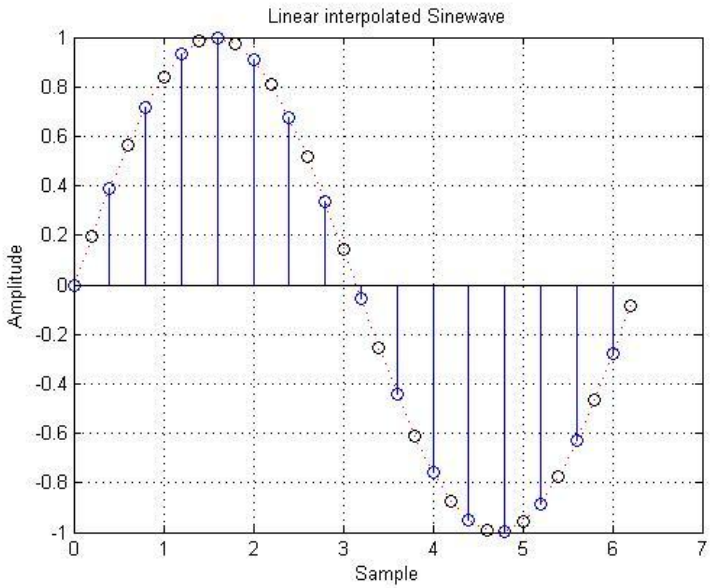
- Used DE2 28Mhz clock chip to drive logic
  - 28 Mhz too fast to allow full range of frequencies
  - Clock split by a factor of 260
  - The 128 sample rate produces an overall reference frequency for an entire period of 840 Hz

Lookup Tables

- One for each of our waves
  - Sine, Triangle, Square, and Noise
- 7 bit addresses, 10 bit data
- Contains one period of each waveform in 128 consecutive slices.
- Used Altera dual-port ROM Megafunction
  - Data files generated using Python script
  - Future work – use folding and symmetries to quarter the size of the lookup ROMs

Interpolator

- Prevents output from ‘jumping’ between samples
  - Uses lowest three bits of address to estimate linearly between adjacent samples
  - Total discrete-valued samples are increased by 2^3



Interpolator Source

```
entity interpolator is
port(
clk : in std_logic;
d0 : in unsigned(9 downto 0);
d1 : in unsigned(9 downto 0);
int_add : in unsigned(2 downto 0);
data_out : out unsigned(9 downto 0)
);
end interpolator;

architecture rtl of interpolator is

signal direction : std_logic;
signal direction_delay_1 : std_logic;
signal diff : unsigned(9 downto 0);
signal step : unsigned(6 downto 0);

begin

cp1 : compare
port map(
        dataa => std_logic_vector(d0),
        datab => std_logic_vector(d1),
        AgB => direction
);

step <= SHIFT_RIGHT(diff, 3)(6 downto 0);

process(clk)
begin
if clk = '1' and clk'event then
        direction_delay_1 <= direction;
        if(direction = '1') then
                diff <= d0 - d1;
        else
                diff <= d1 - d0;
        end if;

        if(direction_delay_1 = '1') then
                data_out <= d0 - (step * int_add);
        else
                data_out <= d0 + (step * int_add);
        end if;

end if;
end process;
end rtl;
```

DAC Output

- We used the onboard VGA Controller DAC to output the sample
  - Required setup and clock signals
  - DAC is run using the 50Mhz clock on the DE2
  - Port VGA-G[9-0] are used for a higher dyanamic range (0->5V)

Conclusions

Conclusions

- System functional – Demonstration
- Generate four possible waveforms with a large array of frequencies

- System performance limited by DAC
  - DAC is non-linear
  - No capacitor on output

Questions?