Clock of Digital System

2013 - 2020

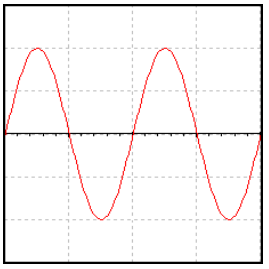
Ando Ki, Ph.D.
(adki@future-ds.com)

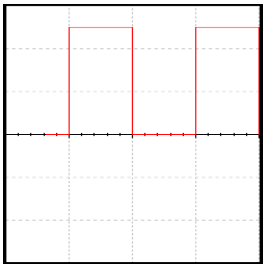
Agenda

- Analog and digital signal
- Switching standard and logic level
- Analog and digital signal: periodic signals
- What is clock in the digital system
- Characteristics of clock
- Clock skew minimization
- How to generate a clock
- How to make clock signal using Verilog-HDL
- Examples
- Projects
- PLL & DLL
- Xilinx DCM: Digital Clock Manager
- DCM usage in general
- Xilinx DCM.v
- Xilinx Spartan-3 DCM
- DCM features and capabilities
- Example: DCM
- How to make clock signals using DCM
- Simulation with Xilinx library
- Example: DCM
- Project; DCM
- Examples
  - ex_clock_at_top
  - ex_dcm_at_top
- Projects
  - prj_clock_module
  - prj_clock_counter
  - prj_clock_parameter
  - prj_dcm_delay

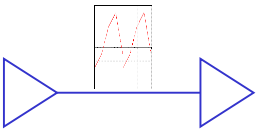
Analog and digital signal

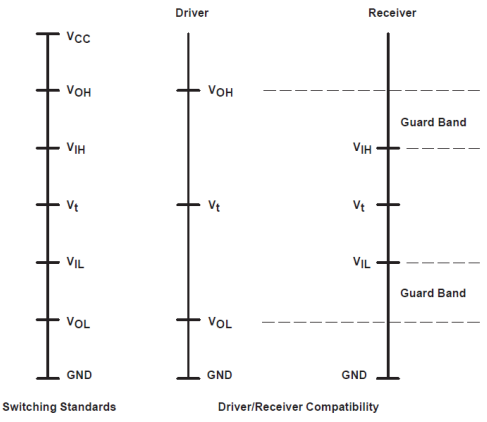
- Analog signal
  - Continuous
  - Infinite range of values
  - More exact values, but more difficult to work with
  - The analog value is continuous and more accurate.
- Digital signal
  - Discrete
  - Finite range of values (2)
  - Not as exact as analog, but easier to work with
  - The digital value is more than adequate for the application and significantly easier to process electronically.





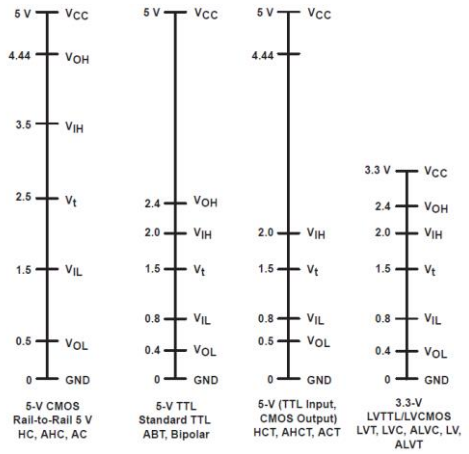
Switching standard and logic level



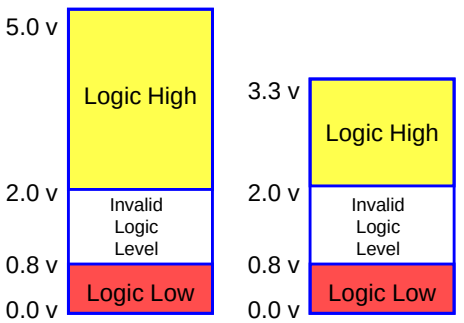


- For receiver
  - VIL: (input low voltage) lower than this will be recognized at low-level (0, L)
  - VIH: (input high voltage) higher than this will be recognized as high-level (1, H)
- For driver
  - VOL: (output low voltage) drive lower than this for low-level
  - VOH: (output high voltage) drive higher than this for high-level
- Purpose of guard band
  - noise immunity (잡음면역)

Switching standard and logic level

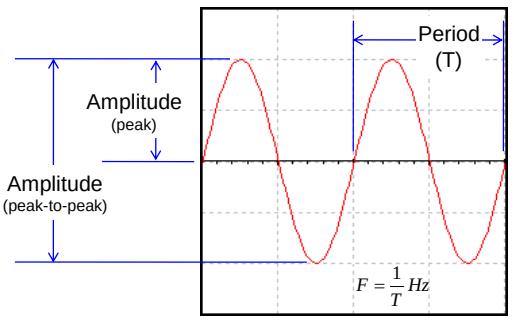


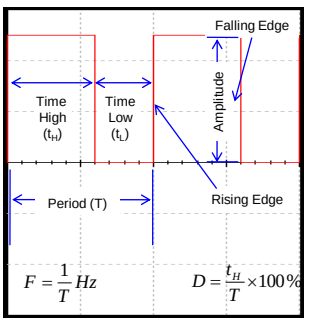
- TTL family switching standard
- Which switching standards are compatible?
  - 5-V TTL and 3.3-V (LVTTL and LVCMOS) have the same switching standards for VOL, VIL, VIH, and VOH. The only difference is VCC.



Analog and digital signal: periodic signals

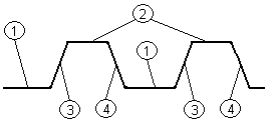
- Analog signal
  - Frequency: F Hertz
  - Amplitude (peak)
  - Amplitude (peak-to-peak)
- Digital
  - Frequency: F Hertz
  - Duty cycle: D %



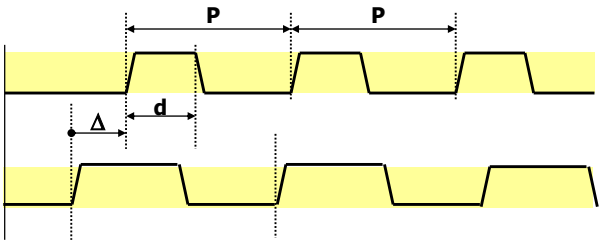


What is clock in the digital system

- 'clock' is a special signal that provides a reference timing.
- The most common clock signal is in the form of a square wave with a 50% duty cycle, usually with a fixed, constant frequency.

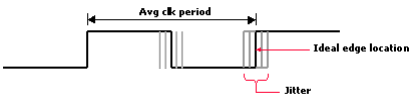


1) low level, (2) high level, (3) rising edge, and (4) falling edge

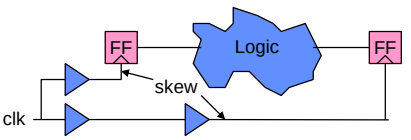


Characteristics of clock

- Clock frequency (clock rate, clock speed)
  - cycles per second
  - 1/P
- Clock duty cycle
  - the ratio of the duration of the event to the total period of a signal.
  - percent of high over the period
  - d/P x 100
- Clock jitter
  - Actual clock signal generator is not ideal so that its period varies in time.



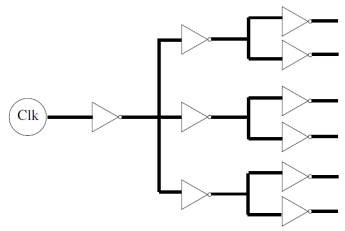
- Clock skew
  - Clock skew is the maximum difference in the arrival time of a clock signal at two different components
  - Skew is just the average delay between two signals.
  - difference between clock delay along different paths

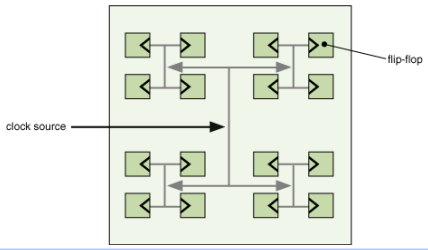


- Voltage level, slew-time/slew-rate, rising time/falling time

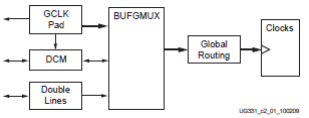
Clock skew minimization

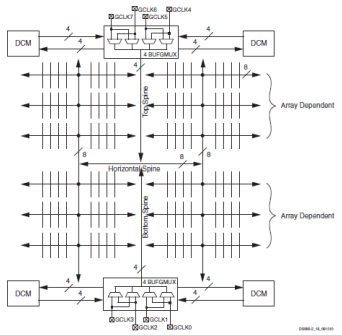
- ASIC (Application Specific IC)
  - Clock tree using buffer and H-tree style routing





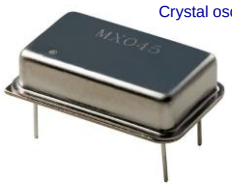
- FPGA
  - Global clock network
    - use global routing through BUFGMUX (**BUFG**)

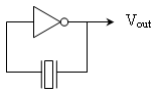




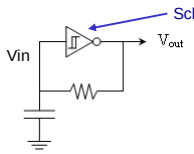
How to generate a clock

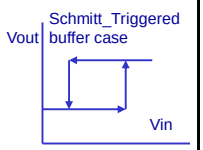
Crystal oscillator

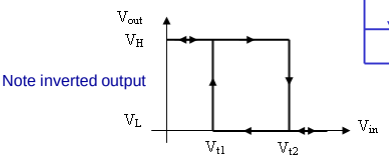


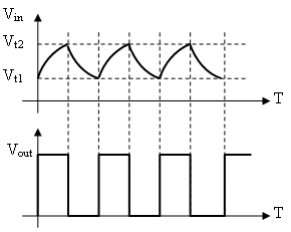


Crystal



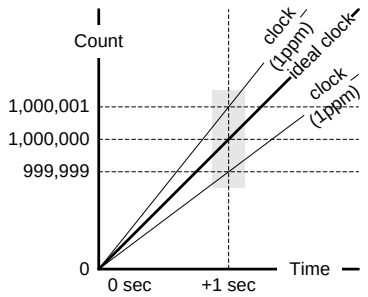




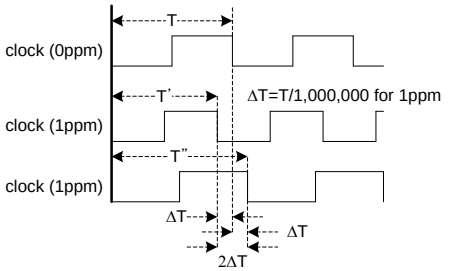


Clock accuracy in PPM

- Clock accuracy
  - a measurement of how a clock actually performs in relation to its ideal performance.
- PPM (Parts Per Million)
  - 1 ppm = $\pm 1/10^6 = \pm 10^{-6} = \pm 10^{-4}\%$

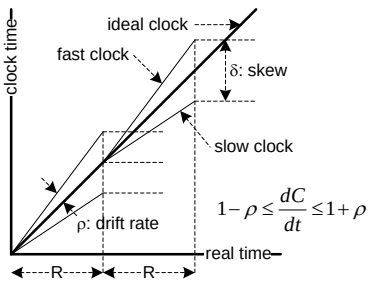


- Example
  - two clocks both with a speed of 1 MHz
  - clock 1: 0ppm (i.e., ideal/perfect clock)
  - clock 2: 1ppm ($\pm$1/1,000,000)
  - two clocks start at the same time from 0

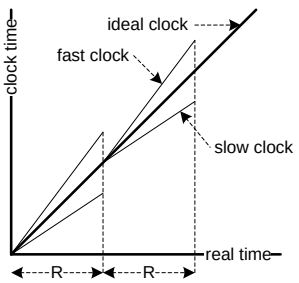


  - 1 second later; count of pulses
    - clock 1: 1,000,000
    - clock 2: 999,999 ~ 1,000,001
      - 1sec*1,000,000/sec*($\pm$1/1,000,000)

Terminologies



$$1-\rho \le \frac{dC}{dt} \le 1+\rho$$

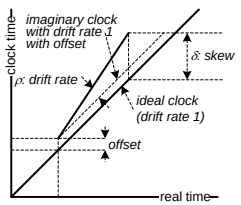


- Terminologies
  - Resynchronization interval R
  - Drift rate ρ
    - Clock drift: Count at different rates. (Different frequency of the oscillator.)

$$\frac{dC}{dt}$$

  - Clock skew δ
    - Clock skew (offset): Difference between time on two clocks.
  - Ideal clock: drifting rate 1

$$\frac{dC}{dt} = 1$$

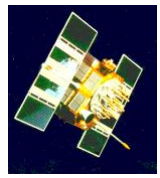


Time accuracy

10,000 sec/day
~2 hours/day

1-10 sec/day

0.5 sec/day



10 nanosec/day
GPS

0.1 nanosec/day
Atomic clock

	Accuracy	Error	Remarks
Typical crystal oscillator	100ppm (0.01%)	8.64 sec/day	86400*0.01%
Watch crystal oscillator	20ppm (0.002%)	1.73 sec/day	86400*0.002%
XO (selected-cut crystal oscillator)	1ppm (0.0001%)	86.4 msec/day	86400*0.0001%
TCXO (temperature compensated X-tal)	0.1ppm		
MCXO (microprocessor compensated X-tal)	0.01ppm		
OCXO (oven controlled X-tal)	0.001ppm	84.6 usec/day	2.6 msec/month
Rubidium atomic	0.000001ppm	84.6 nsec/day	
Cesium atomic	0.0000001ppm	8.64 nsec/day	
Hydrogen atomic	0.00000001ppm	0.864 nsec/day	

How to make clock signal using Verilog-HDL

```
`timescale 1ns/1ns

`ifndef CLK_FREQ
`define CLK_FREQ     100000000 // 100Mhz
`endif

module top ;
  localparam CLK_FREQ=`CLK_FREQ;
  localparam CLK_PERIOD_HALF=1000000000/(CLK_FREQ*2);

  reg clk = 1'b0;

  always #CLK_PERIOD_HALF clk <= ~clk;

  real stamp, delta;
  initial begin
     repeat (10) @ (posedge clk);
     @ (posedge clk) stamp = $time;
     @ (posedge clk) delta = $time - stamp;
     $display("%m clk %f nsec %f Mhz", delta, 1000.0/delta);
     repeat (10) @ (posedge clk);
     $finish(2);
  end

  initial begin
     $dumpfile("wave.vcd");
     $dumpvars(0);
  end

endmodule
```

- This code shows how to make 100Mhz clock signal with 50% duty cycle.
- It reports its period and frequency when it runs.

Use $realtime instead of $time.

Quiz: is it synthesizable code?

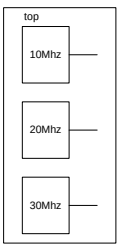
Example: clock at top-level

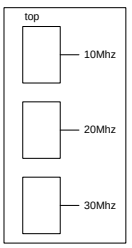
- Run an example code as follows.


```
[user@host] cd $(PROJECT)/codes/clock/ex_clock_at_top/sim/modelsim
[user@host] make
[user@host] gtkwave wave.vcd &
```


Project: clock generation

- Make following three clocks
  - Clock 1:
    - Frequency: 10Mhz
    - Duty cycle: 50%
    - Start with low
  - Clock 3:
    - Frequency: 20Mhz
    - Duty cycle: 40%
    - Start with high
  - Clock 3:
    - Frequency: 30Mhz
    - Duty cycle: 60%
    - Start with low





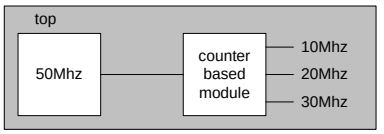
- $(PROJECT)/codes/clock/prj_clock_module
- $(PROJECT)/codes/clock/prj_clock_parameter

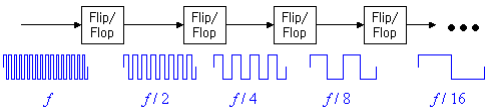
- 1. Simple solution
  - Make three clocks in the top-level

- 2. More advanced solution
  - Make a clock generator module
  - Make three clock generator modules in the top-level
  - The clock generator module takes three parameters
    - Frequency
    - Duty cycle
    - Phase relationship

Project: clock generation using counter

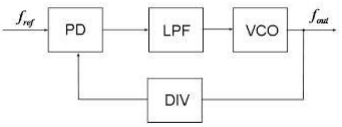
- Make following three clocks from 50Mhz clock input
  - Clock 1:
    - Frequency: 10Mhz
    - Duty cycle: 50%
    - Start with low
  - Clock 3:
    - Frequency: 20Mhz
    - Duty cycle: 40%
    - Start with high
  - Clock 3:
    - Frequency: 30Mhz
    - Duty cycle: 60%
    - Start with low
- $(PROJECT)/codes/clock/prj_clock_counter



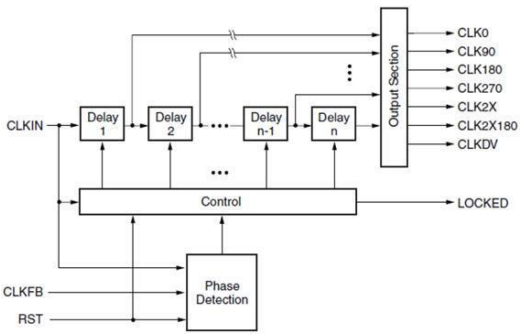


PLL & DLL

- PLL (Phase-locked loop)
  - Phase alignment
  - Frequency multiplication



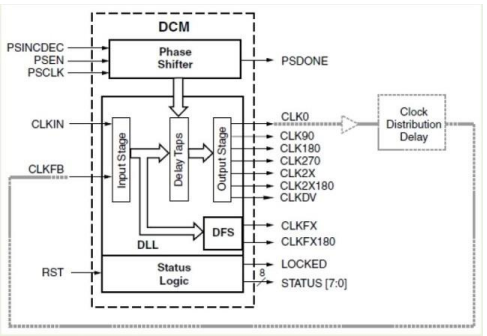
- DLL (Delay-locked loop)
  - CLKIN is used as a reference signal.
  - CLKFB is a feedback signal from the end of the clock distribution network.
  - Clock skew is compensated by correcting phase differences.



Xilinx: PLL, DCM, and MMCM

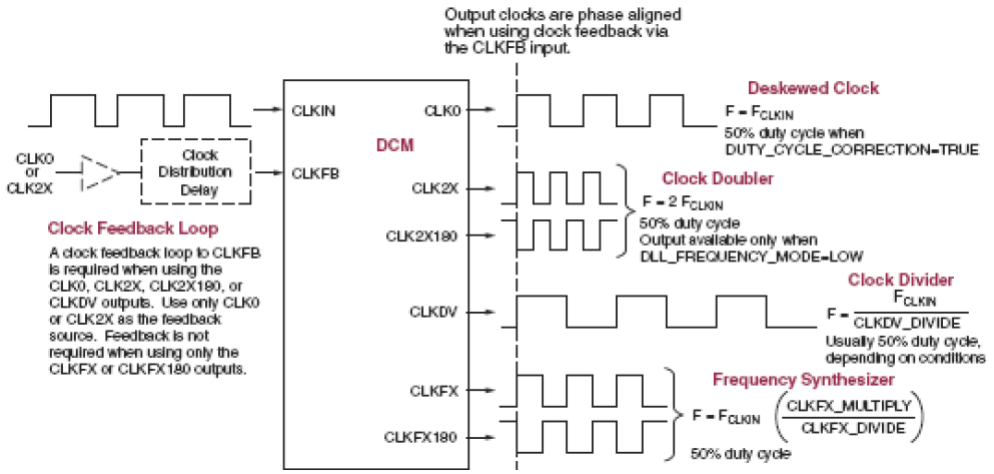
- DCM (Digital Clock Manager)
  - supported up to Spartan-3 and Virtex-4
  - based on DLL (Delayed Locked Loop)
  - de-skew a clock, generate different phases of the clock
  - dynamic change phase
  - 2x clock, clock division
- PLL (Phase Locked Loop)
  - in addition to DCM, supported up to Spartan-6 and Virtex-5
  - a superset of DCM
  - more precise frequency generation
  - generate multiple different frequencies at the same time
- MMCM (Mixed Mode Clock Manager)
  - supported from Virtex-6
  - a superset of PLL

Xilinx DCM: Digital Clock Manager



- Delay Lock Loop
  - uses feedback to deskew clock
- Digital Freq Synth
  - Generates clocks
- Phase Shifter
  - Adjusts phase relationships of output clocks

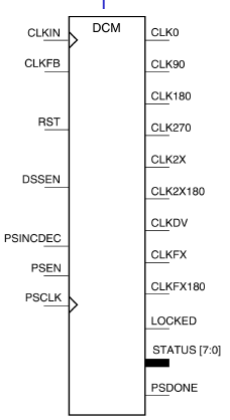
DCM usage in general



Xilinx DCM.v

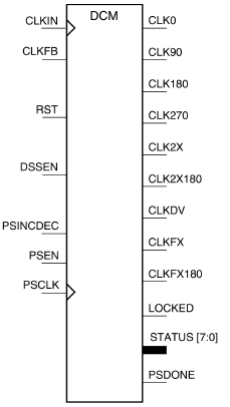
C:\Xilinx\12.4\ISE_DS\ISE\verilog\src\unisims\DCM.v

```
`timescale  1 ps / 1 ps

module DCM (CLK0, CLK180, CLK270, CLK2X, CLK2X180, CLK90,
            CLKDV, CLKFX, CLKFX180, LOCKED, PSDONE, STATUS,
            CLKFB, CLKIN, DSSEN, PSCLK, PSEN, PSINCDEC, RST);
parameter real      CLKDV_DIVIDE = 2.0;
parameter integer CLKFX_DIVIDE = 1;
parameter integer CLKFX_MULTIPLY = 4;
parameter           CLKIN_DIVIDE_BY_2 = "FALSE";
parameter real      CLKIN_PERIOD = 10.0;                    // non-simulatable
parameter           CLKOUT_PHASE_SHIFT = "NONE";
parameter           CLK_FEEDBACK = "1X";
parameter           DESKEW_ADJUST = "SYSTEM_SYNCHRONOUS";   // non-simulatable
parameter           DFS_FREQUENCY_MODE = "LOW";
parameter           DLL_FREQUENCY_MODE = "LOW";
parameter           DSS_MODE = "NONE";                      // non-simulatable
parameter           DUTY_CYCLE_CORRECTION = "TRUE";
parameter           FACTORY_JF = 16'hC080;                  // non-simulatable
parameter integer PHASE_SHIFT = 0;
parameter           SIM_MODE = "SAFE";
parameter           STARTUP_WAIT = "FALSE";                 // non-simulatable

input CLKFB, CLKIN, DSSEN;
input PSCLK, PSEN, PSINCDEC, RST;

output CLK0, CLK180, CLK270, CLK2X, CLK2X180, CLK90;
output CLKDV, CLKFX, CLKFX180, LOCKED, PSDONE;
output [7:0] STATUS;
.... ...

endmodule
```



Xilinx Spartan-3 DCM

```
// Verilog Instantiation Template
// DCM: Digital Clock Manager Circuit for Spartan-3
DCM #( ... ... )
DCM_inst (
.CLK0      (CLK0), // 0 degree DCM CLK output
.CLK180    (CLK180), // 180 degree DCM CLK output
.CLK270    (CLK270), // 270 degree DCM CLK output
.CLK2X     (CLK2X), // 2X DCM CLK output
.CLK2X180  (CLK2X180), // 2X, 180 degree DCM CLK out
.CLK90     (CLK90), // 90 degree DCM CLK output
.CLKDV     (CLKDV), // Divided DCM CLK out (CLKDV_DIVIDE)
.CLKFX     (CLKFX), // DCM CLK synthesis out (M/D)
.CLKFX180  (CLKFX180), // 180 degree CLK synthesis out
.LOCKED    (LOCKED), // DCM LOCK status output
.PSDONE    (PSDONE), // Dynamic phase adjust done output
.STATUS    (STATUS), // 8-bit DCM status bits output
.CLKFB     (CLKFB), // DCM clock feedback
.CLKIN     (CLKIN), // Clock input (from IBUFG, BUFG or DCM)
.PSCLK     (PSCLK), // Dynamic phase adjust clock input
.PSEN      (PSEN), // Dynamic phase adjust enable input
.PSINCDEC  (PSINCDEC), // Dynamic phase adjust increment/decrement
.RST       (RST) // DCM asynchronous reset input
);
// End of DCM_inst instantiation
```

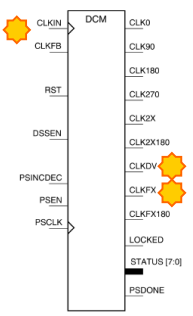


DCM features and capabilities

Feature	Description	DCM Signals
Digital Clock Managers (DCMs) per device	• 4, except in XC3S50 • 2 in XC3S50	All
Digital Frequency Synthesizer (DFS) Input Frequency Range*	1 MHz to 280 MHz	CLKIN
Delay-Locked Loop (DLL) Input Frequency Range*	18 MHz to 280 MHz**	CLKIN
Clock Input Sources	• Global buffer input pad • Global buffer output • General-purpose I/O (no deskew) • Internal logic (no deskew)	CLKIN
Frequency Synthesizer Output	Multiply CLKIN by the fraction (M/D) where M={2..32}, D={1..32}	• CLKFX • CLKFX180
Clock Divider Output	Divide CLKIN by 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 9, 10, 11, 12, 13, 14, 15, or 16	CLKDV
Clock Doubler Output	Multiply CLKIN frequency by 2	• CLK2X • CLK2X180
Clock Conditioning, Duty-Cycle Correction	Always provided on most outputs. Optional on CLK0, CLK90, CLK180, CLK270. 50% duty cycle ±150 to 400 ps*	All
Quadrant Phase Shift Outputs	0° (no phase shift), 90° (¼ period), 180° (½ period), 270° (¾ period)	• CLK0 • CLK90 • CLK180 • CLK270
Half-period Phase Shift Outputs	Output pairs with 0° and 180° phase shift, Ideal for DDR applications	• CLK0, CLK180 • CLK2X, CLK2X180 • CLKFX, CLKFX180
Dynamic or Fixed Phase Shift resolution	Down to 1/256th of a clock period (or ~30 to 60 ps)*	All
Number of Clock Outputs to General-purpose Interconnect	Up to all 9	All
Number of Clock Outputs to Global Clock Network	Any 4 of 9	All
Number of Clock Outputs to Output Pins	Up to all 9	All

$$F_{CLKDV} = F_{CLKIN} \times {}^{1}/_{N_{DIV}}$$

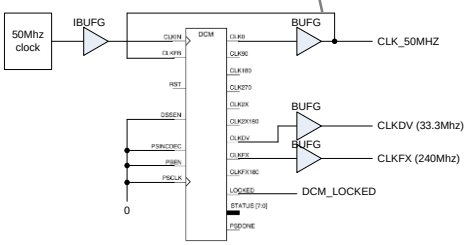
$$F_{CLKFX} = F_{CLKIN} \times {}^{N_{FXMUL}}/_{N_{FXDIV}}$$



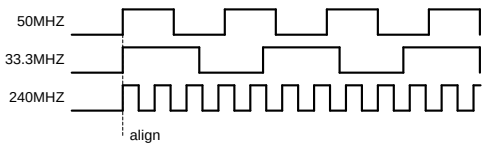
Ref: Using Digital Clock Managers (DCMs) in Spartan-3 FPGAs, XAPP462, Xilinx, 2006.

Example: DCM

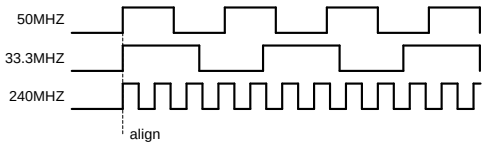
Feedback to compensate phase delay (skew) between CLKIN and CLK_50MHZ



Phase relationship



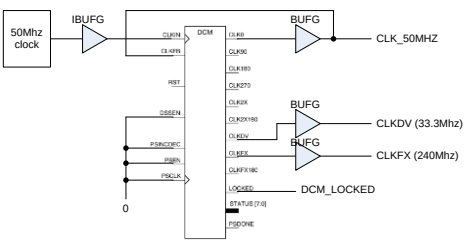
What to make



How to make clock signals using DCM

```
`timescale 1ns/1ps
module top ;
  reg osc_clk = 1'b0;
  always #10 osc_clk <= ~osc_clk; // 50Mhz
  ... ..,
  IBUFG u_ibufg(.I(osc_clk), .O(CLKin));

  DCM #(.CLKFX_MULTIPLY( 24   ),.CLKFX_DIVIDE  (  5   )
          ,.CLKDV_DIVIDE  (  1.5 ),.CLKIN_PERIOD  ( 20.0 ) )
 u_dcm (
      .CLKIN    ( CLKin        )
    , .CLKFB    ( CLK0         )
    , .DSSEN    ( 1'b0         )
    , .RST      ( RST          )
    , .PSEN     ( 1'b0         )
    , .PSINCDEC ( 1'b0         )
    , .PSCLK    ( 1'b0         )
    , .CLK0     ( CLKdcm_0     ) // 50Mhz
    , .CLK90    ( CLKdcm_90    )
    , .CLK180   ( CLKdcm_180   )
    , .CLK270   ( CLKdcm_270   )
    , .CLK2X    ( CLKdcm_2X    ) // 100Mhz
    , .CLK2X180 ( CLKdcm_2X180 )
    , .CLKDV    ( CLKdcm_dv    ) // 33.3Mhz
    , .CLKFX    ( CLKdcm_fx    ) // 240Mhz
    , .CLKFX180 ( CLKdcm_fx180 )
    , .STATUS   ( STATUS       )
    , .LOCKED   ( DCM_LOCKED   )
    , .PSDONE   ( PSDONE       )
 );
```

- This code shows how to make 240Mhz and 33.3Mhz from 50Mhz using DCM




```
  BUFG u_bufg   (.I(CLKdcm_0 ), .O(CLK0 ));
  BUFG u_bufg_dv(.I(CLKdcm_dv), .O(CLKDV));
  BUFG u_bufg_fx(.I(CLKdcm_fx), .O(CLKFX));
... ...
endmodule
```

Simulation with Xilinx library

How to use Xilinx Verilog library with ModelSim

```
+libext+.v
-y $XILINX/verilog/src
-y $XILINX/verilog/src/unisims
-y $XILINX/verilog/src/XilinxCoreLib
  $XILINX/verilog/src/glbl.v

+incdir+../../bench/verilog
      ../../bench/verilog/top.v
```

modelsim.args

```
all: vlib compile simulate

vlib:
      if [ -d work ]; then /bin/rm -rf work; fi
      (vlib work || exit -1) 2>&1 | tee compile.log

compile:
      (vlog -lint -work work\
            -f modelsim.args || exit -1) 2>&1 | tee compile.log

simulate: compile
      vsim -novopt -c -do "run -all; quit"\
            work.top
            work.glbl
```

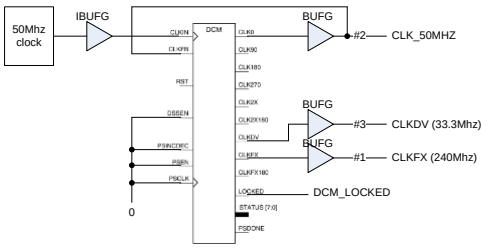
Makefile

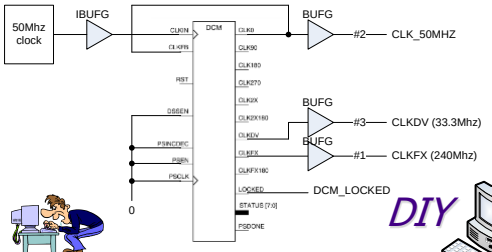
Example: DCM clock at top-level

- Run an example code as follows.


```
[user@host] cd $(PROJECT)/codes/clock/ex_dcm_at_top/sim/modelsim
[user@host] make
[user@host] gtkwave wave.vcd &
```

Project: clock generation

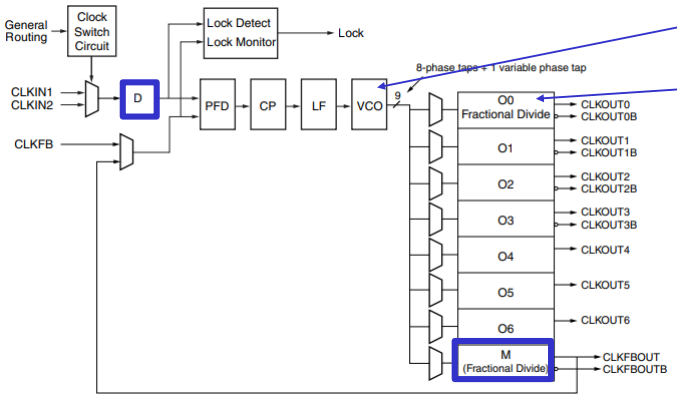




DIY

- $(PROJECT)/codes/clock/prj_dcm_delay
  - make this directory
- Note followings
  - Feedback changed
  - Wires has delays
- Make the design work and terminate by itself.
  - The code given will not work and you need to modify some.
  - Tip: read DCM manual carefully.
    - XAPP462.pdf
- Explain the results with waveform
  - what have been affected by delay
  - what have been affected by moving feedback signal.

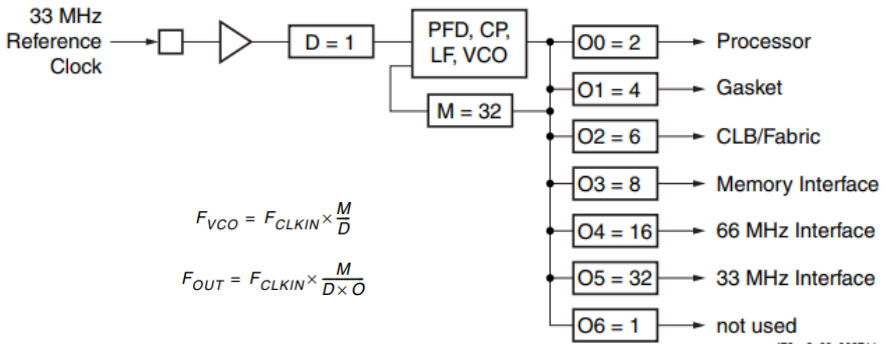
MMCM



$$F_{VCO} = F_{CLKIN} \times \frac{M}{D}$$

$$F_{OUT} = F_{CLKIN} \times \frac{M}{D \times O}$$

MMCM example



$$F_{VCO} = F_{CLKIN} \times \frac{M}{D}$$

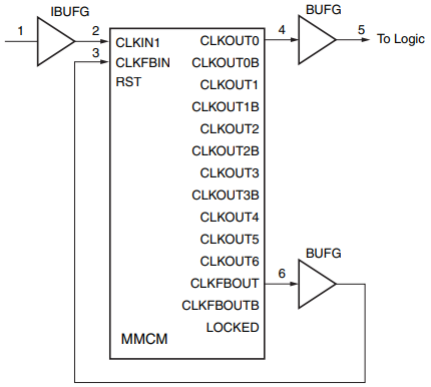
$$F_{OUT} = F_{CLKIN} \times \frac{M}{D \times O}$$

MMCM attributes

Attribute	Type	Allowed Values	Default	Description
CLKOUT[1:6]_DIVIDE	Integer	1 to 128	1	Specifies the amount to divide the associated CLKOUT clock output if a different frequency is desired. This number in combination with the CLKFBOUT_MULT_F and DIVCLK_DIVIDE values will determine the output frequency.
CLKOUT[0]_DIVIDE_F(2)	Integer or Real	1 to 128 or 2.000 to 128.000 in increments of 0.125	1	
CLKOUT[0:6]_PHASE	Real	–360.000 to 360.000 in increments of 1/56 the F_{VCO} and/or increments depending on CLKOUT_DIVIDE.	0.0	Allows specification of the output phase relationship of the associated CLKOUT clock output in number of degrees offset (that is, 90 indicates a 90° or ¼ cycle offset phase offset while 180 indicates a 180° offset or ½ cycle phase offset).
CLKOUT[0:6]_ DUTY_CYCLE	Real	0.01 to 0.99	0.50	Specifies the Duty Cycle of the associated CLKOUT clock output in percentage (that is, 0.50 will generate a 50% duty cycle).
CLKFBOUT_MULT_F(2)	Integer or Real	2 to 64 or 2.000 to 64.000 in increments of 0.125	5	Specifies the amount to multiply all CLKOUT clock outputs if a different frequency is desired. This number, in combination with the associated CLKOUT#_DIVIDE value and DIVCLK_DIVIDE value, will determine the output frequency.
DIVCLK_DIVIDE	Integer	1 to 106	1	Specifies the division ratio for all output clocks with respect to the input clock. Effectively divides the CLKIN going into the PFD.
CLKFBOUT_PHASE	Real	0.00 to 360.00	0.0	Specifies the phase offset in degrees of the clock feedback output. Shifting the feedback clock results in a negative phase shift of all output clocks to the MMCM.

$$F_{OUT} = F_{CLKIN} \times \frac{M}{D \times O}$$

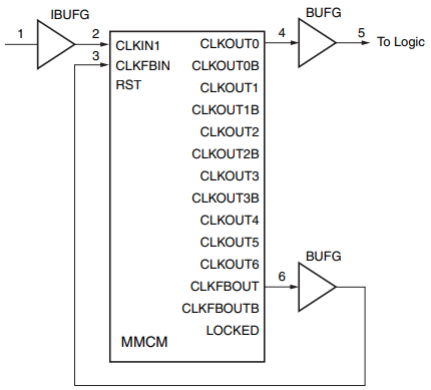
MMCM example



MMCM example

```
BUFG BUFG_CLKOUT0  ( .I( SYS_CLK_CLKOUT0 ), .O( CLKOUT0 ));
BUFG BUFG_CLKOUT1  ( .I( SYS_CLK_CLKOUT1 ), .O( CLKOUT1 ));
BUFG BUFG_CLKOUT2  ( .I( SYS_CLK_CLKOUT2 ), .O( CLKOUT2 ));
BUFG BUFG_CLKOUT3  ( .I( SYS_CLK_CLKOUT3 ), .O( CLKOUT3 ));
BUFG BUFG_CLKOUT4  ( .I( SYS_CLK_CLKOUT4 ), .O( CLKOUT4 ));

localparam real CLK_IN_PERIOD_NS = 1000.0/(INPUT_CLOCK_FREQ/1_000_000.0);
localparam      MUL  =  1_000/(INPUT_CLOCK_FREQ/1_000_000);// 18.0  // 5~64
localparam      DIV  = ((INPUT_CLOCK_FREQ/1_000_000)*MUL)/(CLKOUT0_FREQ/1_000_000); // 1.0~128.0
localparam real CLK_MUL  = MUL
          , CLK0_DIV = DIV;
localparam      CLK1_DIV = ((INPUT_CLOCK_FREQ/1_000_000)*CLK_MUL)/(CLKOUT1_FREQ/1_000_000) //1~128
          , CLK2_DIV = ((INPUT_CLOCK_FREQ/1_000_000)*CLK_MUL)/(CLKOUT2_FREQ/1_000_000)
          , CLK3_DIV = ((INPUT_CLOCK_FREQ/1_000_000)*CLK_MUL)/(CLKOUT3_FREQ/1_000_000)
          , CLK4_DIV = ((INPUT_CLOCK_FREQ/1_000_000)*CLK_MUL)/(CLKOUT4_FREQ/1_000_000);

MMCME2_BASE #(.BANDWIDTH("OPTIMIZED"), // Jitter programming (OPTIMIZED, HIGH, LOW)
          .CLKFBOUT_MULT_F(CLK_MUL), // Multiply value for all CLKOUT (2.000-64.000).
          .CLKFBOUT_PHASE (0.0), // Phase offset in degrees of CLKFB (-360.000-360.000).
          .CLKIN1_PERIOD  (CLK_IN_PERIOD_NS), // Input clock period in ns to ps resolution (i.e. 33.333 is 30 MHz).
          // CLKOUT0_DIVIDE - CLKOUT6_DIVIDE: Divide amount for each CLKOUT (1-128)
          .CLKOUT1_DIVIDE  (CLK1_DIV),
          . ...
          // CLKOUT0_DUTY_CYCLE - CLKOUT6_DUTY_CYCLE: Duty cycle for each CLKOUT (0.01-0.99).
          .CLKOUT0_DUTY_CYCLE(0.5),
          ....
          )
```

MMCM example



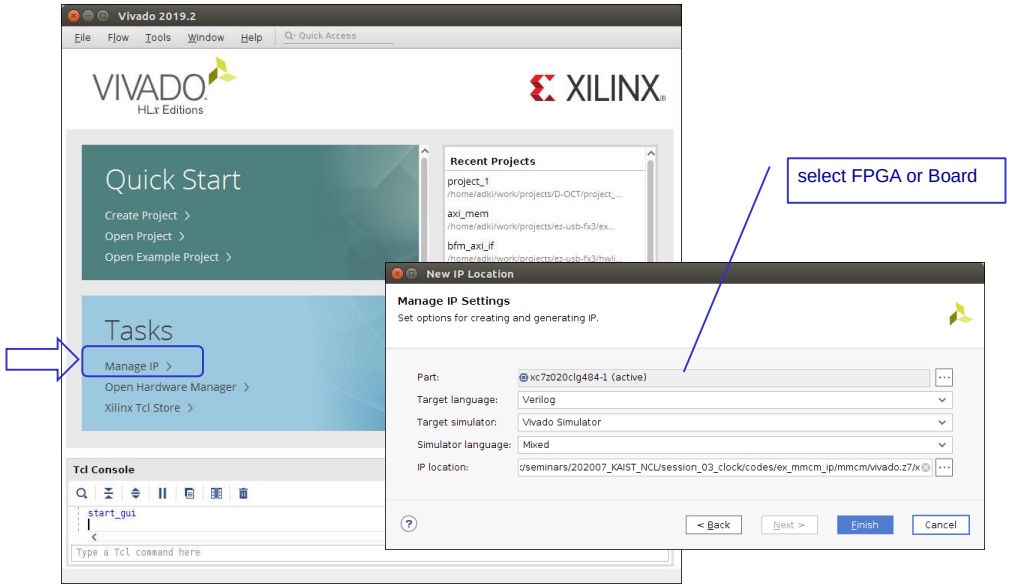

```
u_SYS_CLK (
        // Clock Outputs: 1-bit (each) output: User configurable clock outputs
        .CLKOUT0 (SYS_CLK_CLKOUT0), // 1-bit output: CLKOUT0
        .CLKOUT0B(      ), // 1-bit output: Inverted CLKOUT0
        .CLKOUT1 (SYS_CLK_CLKOUT1), // 1-bit output: CLKOUT1
        .CLKOUT1B(      ), // 1-bit output: Inverted CLKOUT1
        .CLKOUT2 (SYS_CLK_CLKOUT2), // 1-bit output: CLKOUT2
        .CLKOUT2B(      ), // 1-bit output: Inverted CLKOUT2
        .CLKOUT3 (SYS_CLK_CLKOUT3), // 1-bit output: CLKOUT3
        .CLKOUT3B(      ), // 1-bit output: Inverted CLKOUT3
        .CLKOUT4 (SYS_CLK_CLKOUT4), // 1-bit output: CLKOUT4
        .CLKOUT5 (      ), // 1-bit output: CLKOUT5
        .CLKOUT6 (      ), // 1-bit output: CLKOUT6
        // Feedback Clocks: 1-bit (each) output: Clock feedback ports
        .CLKFBOUT (CLKFBOUT), // 1-bit output: Feedback clock
        .CLKFBOUTB(      ), // 1-bit output: Inverted CLKFBOUT
        // Status Port: 1-bit (each) output: MMCM status ports
        .LOCKED   (LOCKED), // 1-bit output: LOCK
        // Clock Input: 1-bit (each) input: Clock input
        .CLKIN1   (CLK_IN    ), // 1-bit input: Clock
        // Control Ports: 1-bit (each) input: MMCM control ports
        .PWRDWN   (1'b0), // 1-bit input: Power-down
        .RST      (RESET), // 1-bit input: Reset
        // Feedback Clocks: 1-bit (each) input: Clock feedback ports
        .CLKFBIN  (CLKFBIN) // 1-bit input: Feedback clock
);
```


Example: MMCM

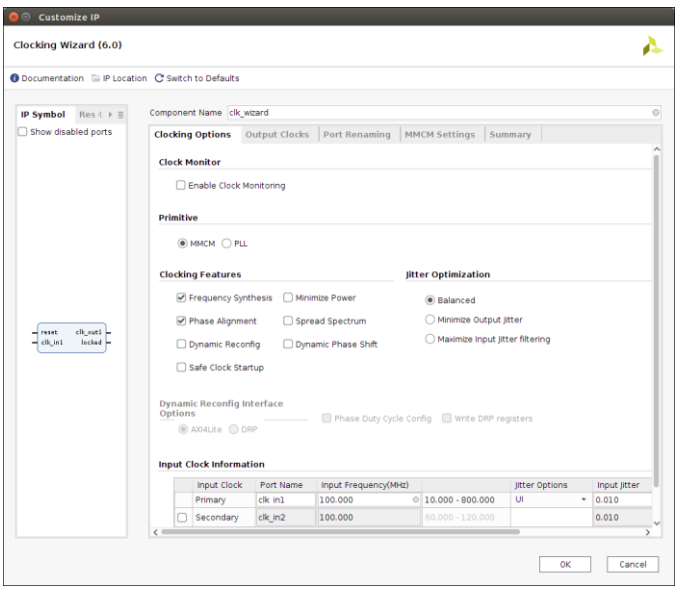
- Run an example code as follows.


```
[user@host] cd $(PROJECT)/codes/clock/ex_mmcm_rtl
[user@host] make
[user@host] gtkwave wave.vcd &
```

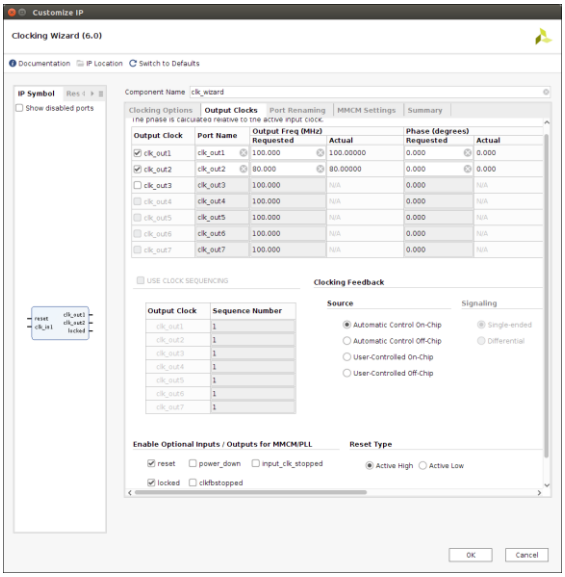
Vivado IP manager



Vivado Clocking Wizard



Vivado Clocking Wizard



Example: Clock Wizard

- Run an example code as follows.


```
[user@host] cd $(PROJECT)/codes/clock/ex_mmcm_ip/sim/xsim
[user@host] make
[user@host] gtkwave wave.vcd &
```

References

- 7 Series FPGAs Clocking Resources User Guide UG472
- Using Digital Clock Managers (DCMs) in Spartan-3 FPGAs, XAPP462, Xilinx, 2006.
- Spartan-3 Libraries Guide for HDL Designs (page 30, DCM), UG607, Xilinx 2011.
- Spartan-3 Generation FPGA User Guide, UG331 (v1.8), Xilinx, June 13, 2011