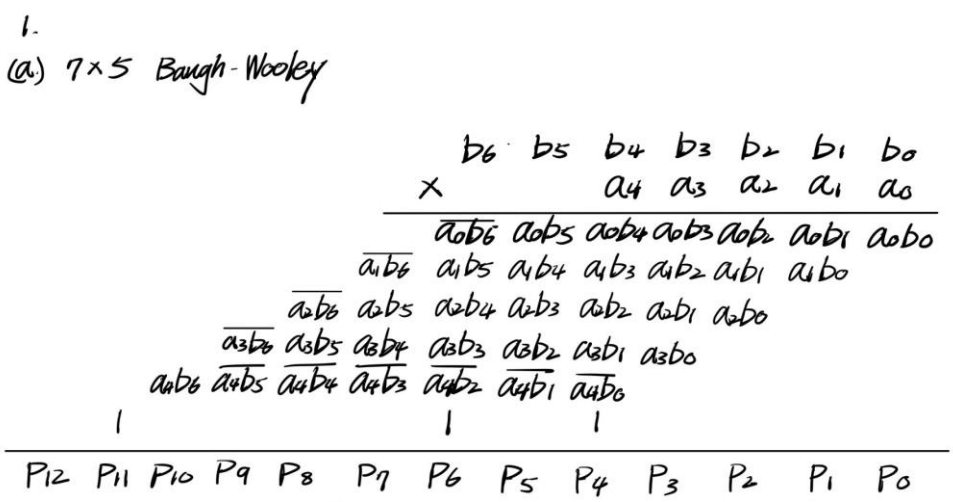
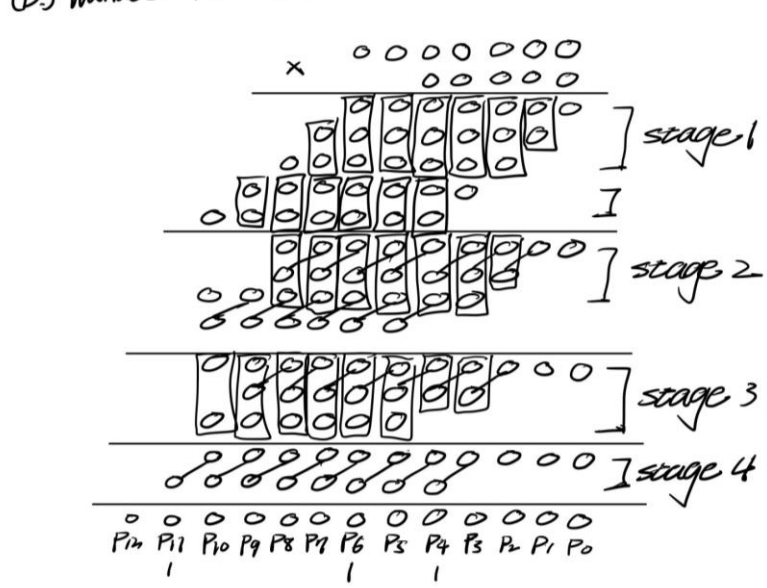
DSPIC HW4

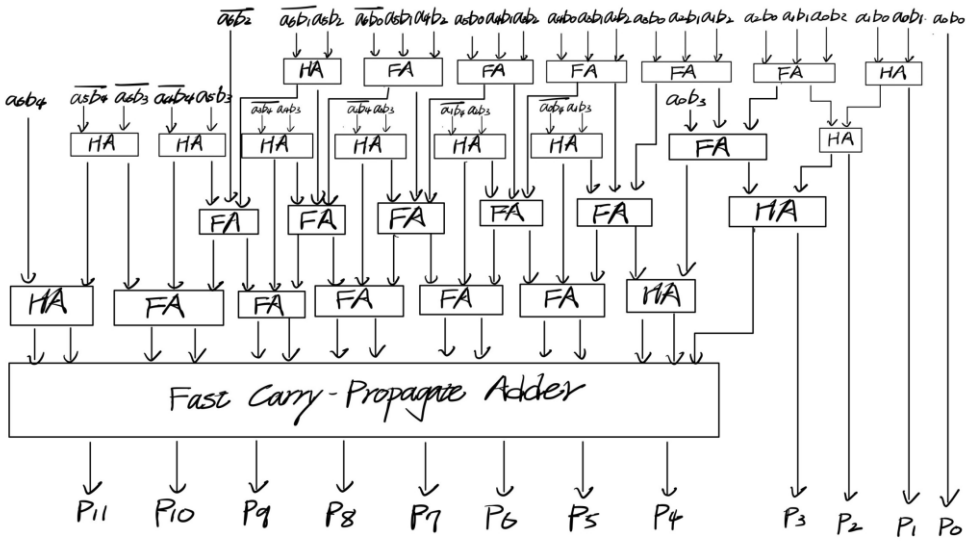
111061545 陳揚哲

1. Please design an 7x5 **Baugh-Wooley** multiplier based on Wallace-tree carry-save-addition (CSA) architecture.
   (a) Derive the long multiplication chart of an 7x5 **Baugh-Wooley** multiplier (similar to Page 29 and 34 of Chapter 9).
   As the figure below. We only need to care about bit 0 to bit 11 (total 12 bits).



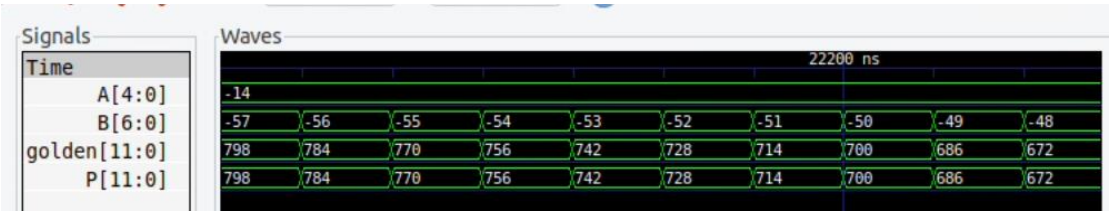
   (b) Plot architecture of the 7x5 Wallace-Tree multiplier (similar to Page 35 of Chapter 9) based on full adder and half adder.





(c) Simulate the architecture by coding the 7x5 **Baugh-Wooley** multiplier in RTL Verilog HDL language. (Please use more than 20 sets of inputs to verify your design. Four combinations of negative and positive signs must all be included.)

I use total 3780 dataset in my simulation, which a is from 1 to 2^5-1, b is from 1 to 2^7-1. From the simulation log, all the test result is correct. Here is part of simulation waveform, we can find out all result is correct in xsim.log.



2. Please design a low-pass digital FIR filter with the following target specifications

(a) Design a low-pass digital FIR filter with the smallest number of floating-point coefficients using C/C++ program. You can use Matlab tool functions to calculate the filter coefficients.

$20 \cdot \log(\delta_s) < -40\text{dB}$ stopband attenuation (-40dB)

$f_p = 1.5\text{kHz}$ passband edge frequency

$F_s = 8\text{kHz}$ sampling frequency

$\Delta f = 0.5\text{kHz}$ transition width of the filter

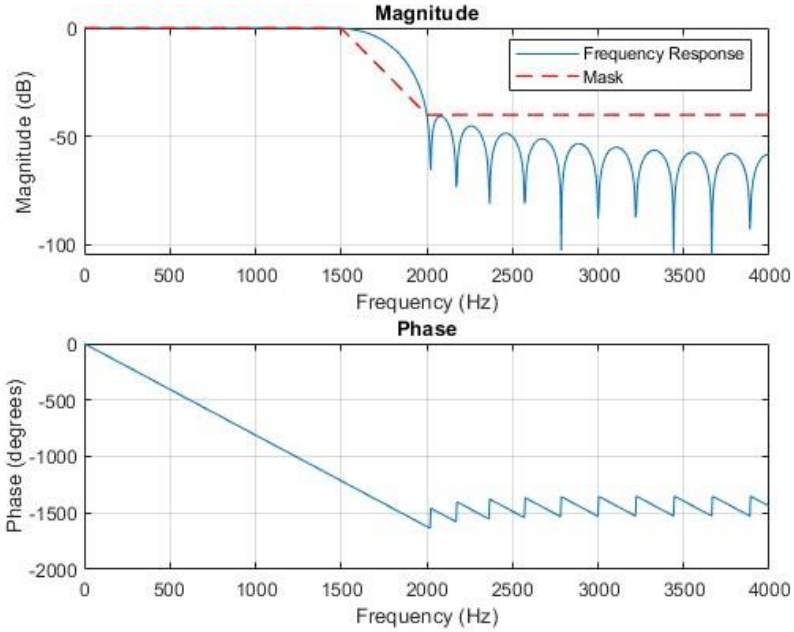
(i) List the floating-point coefficients of the digital FIR filter.

To find the coefficients of the FIR filter, we should specify what type of the FIR we want to apply. Here, I choose Kaiser window function to design our FIR filter. From the specification above, we specify the

parameter needed for designing FIR filter. Fs = 8000 Hz, Passband Edge frequency 1500 Hz, Transition Width = 500 Hz, delta = 0.01 (to make Attenuation = -40dB). We can design the filter, and the coefficients are listed.

```
hh =

  Columns 1 through 11

   -0.0010   -0.0036    0.0000    0.0065    0.0032   -0.0088   -0.0093    0.0090    0.0184   -0.0047   -0.0297

  Columns 12 through 22

   -0.0071    0.0417    0.0316   -0.0524   -0.0848    0.0598    0.3108    0.4375    0.3108    0.0598   -0.0848

  Columns 23 through 33

   -0.0524    0.0316    0.0417   -0.0071   -0.0297   -0.0047    0.0184    0.0090   -0.0093   -0.0088    0.0032

  Columns 34 through 37

    0.0065    0.0000   -0.0036   -0.0010
```

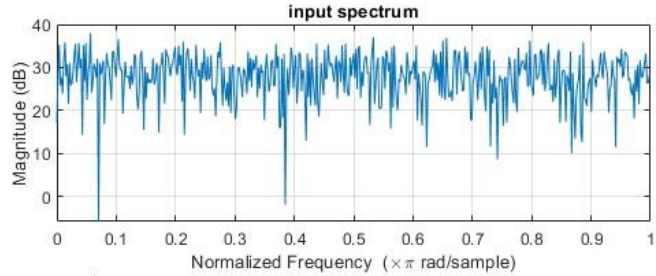
(ii) Plot the frequency response of the digital FIR filter and the frequency mask of the target filter specifications (using Matlab).

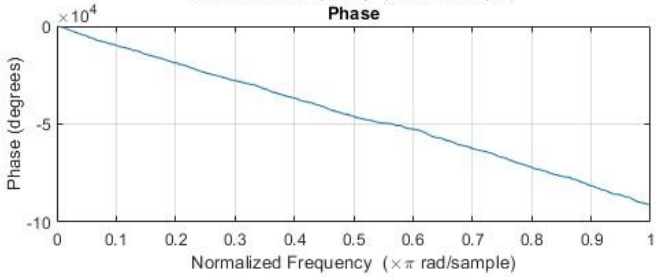


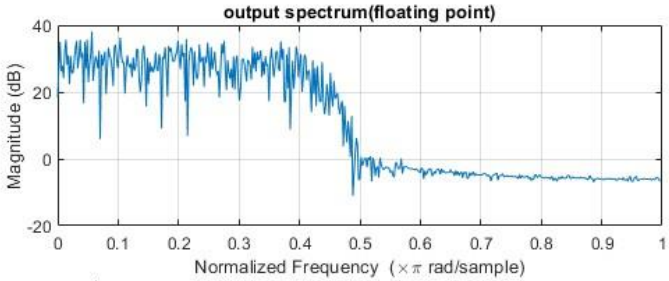
(b) Fixed-point simulation: determine the word-length of input signals and filter coefficients of this low-pass digital FIR filter using C/C++ program.

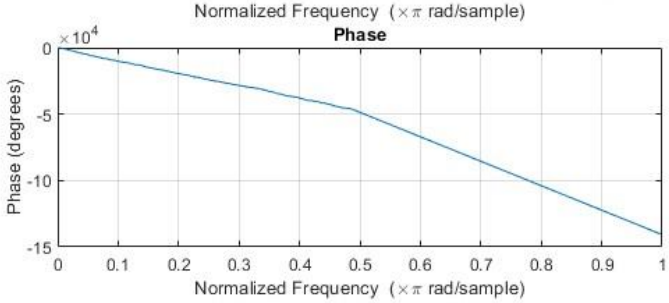
(i) Randomly generate test patterns of floating-point signed real-value signals with average power of 1. Please use enough number of test patterns to perform the target FIR processing. Please analyze the output signal spectral and verify that the output signals are filtered by the target filter specifications. You can plot the spectrum of the input signal spectrum/output signal spectrum along with the target filter specifications of frequency response in a figure.

To generate our test patterns, I use random function in C++. I randomly generate 1024 signals ranging from -500 to 499. If we want to let the average power of the signal equal to 1, we should sum up the average power of each signal and divide by numbers of data. We can express it in mathematic function $\textbf{average power} = \frac{\sum_n x^2}{n}$. After that, we only need let each of the original signals divided by roots of the average power. We save this signal and plot the input spectrum in Matlab. Then, we perform the FIR in C++. I create an array to simulate the shift register behavior. By doing so, we can easily perform FIR calculation. After calculation, we analyze the output signal spectral and verify that the output signals are filtered by the target filter. (passband edge = 0.375 π , stopband edge = 0.5 π in normalized frequency)



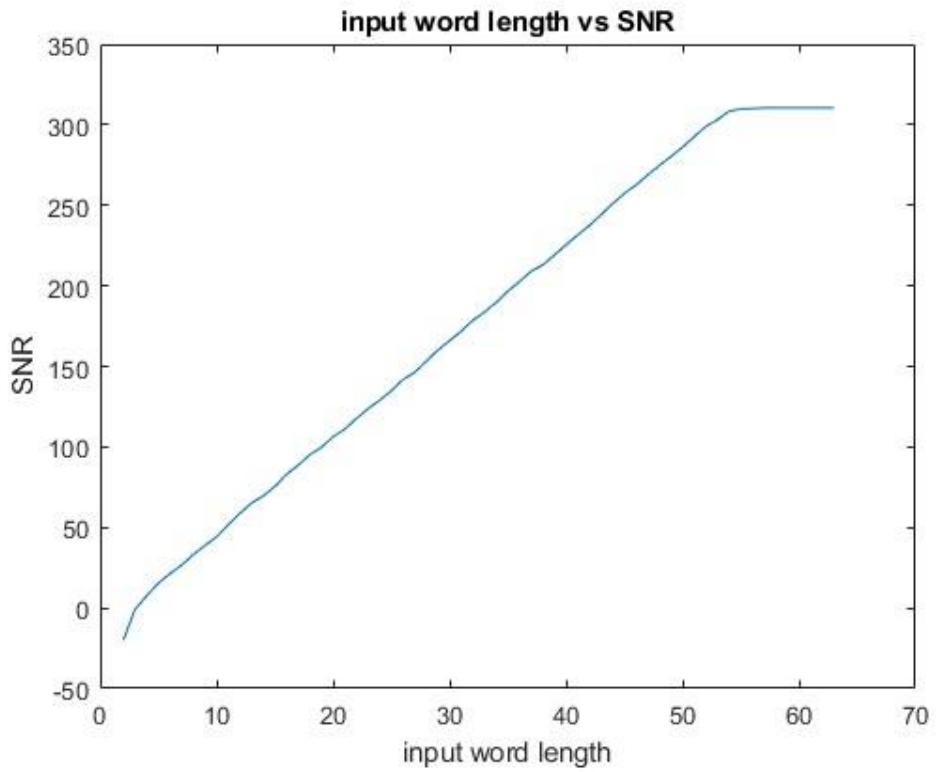




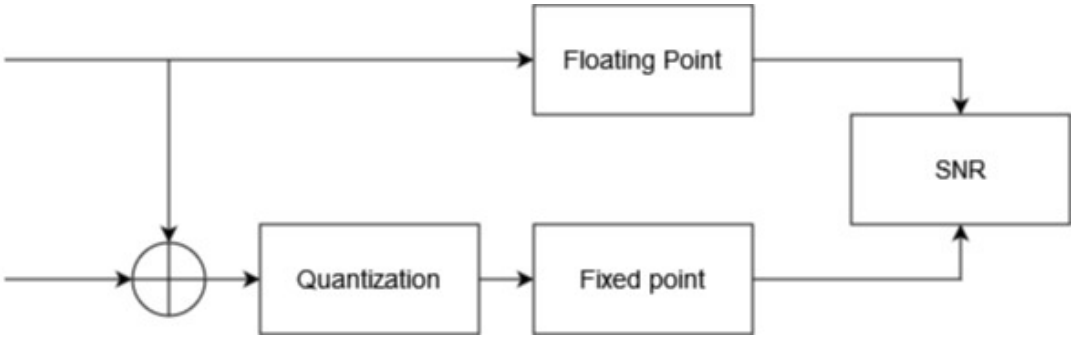


(ii) Perform the fixed-point simulations based on the output SNR metric. Plot the figure of output SNR versus input word length and the figure of output SNR versus word-length of the multiplier-and-accumulator (MAC) to determine the word-length of the digital FIR filter.

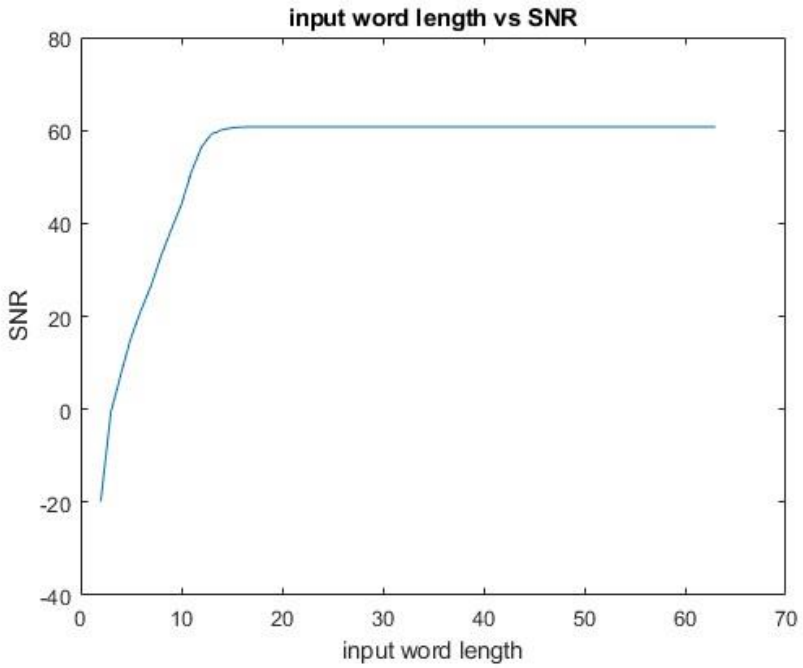
Before we perform the experiment to choose the input word length and MAC word length. Since our calculation is based on double data type in C++, this is very precise calculation. Therefore, it's hard to find the saturation in our experiment. We need to use very large input word length to reach the saturation point. In input word length vs SNR, we assume MAC word length = 64.



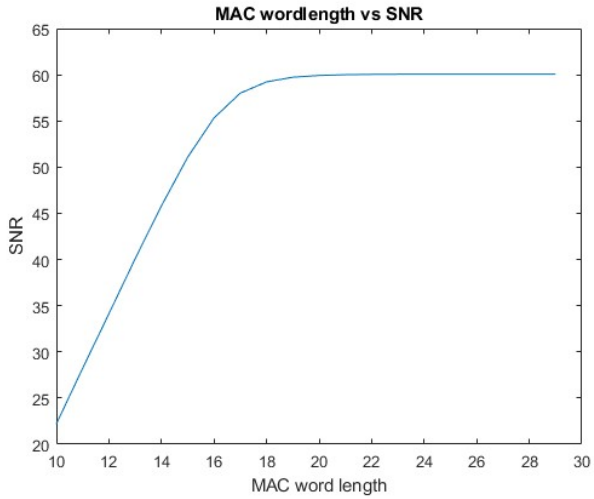
However, in our experiment, we add the Gaussian noise (as channel noise) to input signal for noise resource, the result SNR will increase until saturation. The noise magnitude is about $\frac{1}{100}$ of the original signal.



Thus, I add the noise about $\frac{1}{100}$ magnitude of the input signal. Once we add the noise and perform the input word length test and MAC word length test, we can get the ideal saturation curve. To find the ideal word length, I gradually increase the input word length from 1 to 64, while fix MAC word length = 64. Compare the floating point result to fixed point result, we can plot the curve as below. From this curve, we choose the input word length = 14.

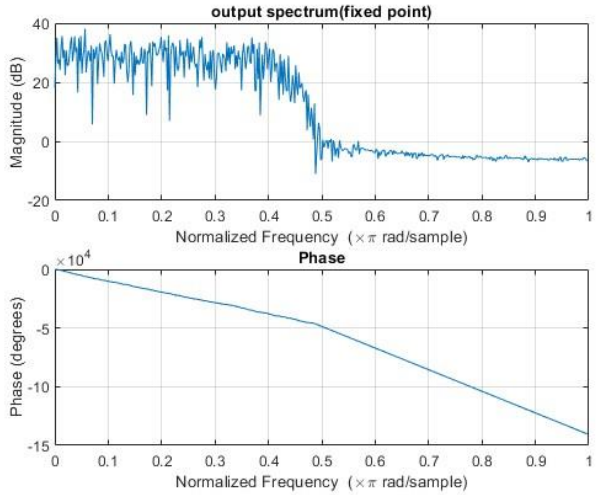


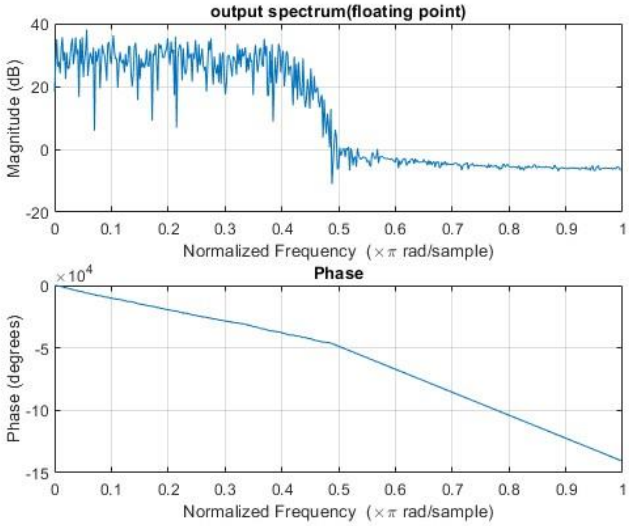
Then, to find the MAC word length, we fix the input word length = 14, gradually increase MAC word length. However, since the input word length is fix as 14. Too large MAC word length won't take effect. Therefore, we only change the MAC word length from 10 to 29.



Compare the floating point result to fixed point result, we choose the MAC word length = 20.

After we choose the input word length = 14, output word length = 20. We can compare the result of floating point and fixed point. The result shows that fixed point and floating point has similar effect.



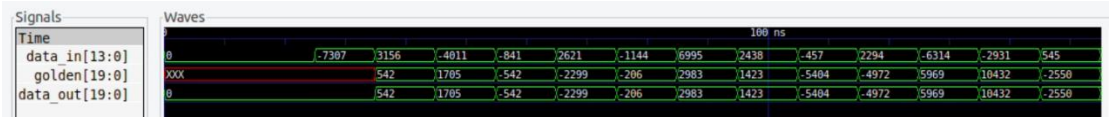


(c) Design this low-pass digital FIR filter by using Verilog HDL according to word-length determined in the fixed-point simulations. You can use the Xilinx Vivado or ISE software to design and synthesize the HDL code with arbitrary FPGA device. (You can also use an available and legally licensed standard cell library to simulate the corresponding results to FPGA.)

(i) Simulate your Verilog HDL code with the same input pattern in the fixed-point simulation. Compare the output results of the fixed-point program and Verilog HDL simulations. (output waveforms)

We use quantized coefficient and quantized input signal to generate output. We can compare the Verilog result and the C++ result, to see

whether our design is reasonable. We retrieve part of waveform, we can find out the output signal is equal to the golden data. To watch all of the output signal, the simulation log (xsim.log) has total 1024 results, all of them are equal to golden data.




```
[Correct] [Pattern       1019] Golden answer:   341844, Your answer:   341844
[Correct] [Pattern       1020] Golden answer:   231583, Your answer:   231583
[Correct] [Pattern       1021] Golden answer:   -25645, Your answer:   -25645
[Correct] [Pattern       1022] Golden answer:  -275282, Your answer:  -275282
[Correct] [Pattern       1023] Golden answer:  -360234, Your answer:  -360234
```

(ii) List the circuit speed and FPGA resource numbers of your digital Filter design. Please provide synthesis and implementation reports of your design synthesis and implementation reports of your design from the tools.

Resource usage:

Name ^ 1	Slice LUTs (53200)	Slice Registers (106400)	DSPs (220)	Bonded IOB (125)	BUFGCTRL (32)
N fir	313	518	35	36	1

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	313	0	0	53200	0.59
LUT as Logic	313	0	0	53200	0.59
LUT as Memory	0	0	0	17400	0.00
Slice Registers	518	0	0	106400	0.49
Register as Flip Flop	518	0	0	106400	0.49
Register as Latch	0	0	0	106400	0.00
F7 Muxes	0	0	0	26600	0.00
F8 Muxes	0	0	0	13300	0.00

2. Memory

Site Type	Used	Fixed	Prohibited	Available	Util%
Block RAM Tile	0	0	0	140	0.00
RAMB36/FIFO*	0	0	0	140	0.00
RAMB18	0	0	0	280	0.00

* Note: Each Block RAM Tile only has one FIFO logic available and therefore

3. DSP

Site Type	Used	Fixed	Prohibited	Available	Util%
DSPs	35	0	0	220	15.91
DSP48E1 only	35				

4. IO and GT Specific

Site Type	Used	Fixed	Prohibited	Available	Util%
Bonded IOB	36	0	0	125	28.80
Bonded IPADs	0	0	0	2	0.00
Bonded IOPADs	0	0	0	130	0.00
PHY_CONTROL	0	0	0	4	0.00
PHASER_REF	0	0	0	4	0.00
OUT_FIFO	0	0	0	16	0.00
IN_FIFO	0	0	0	16	0.00
IDELAYCTRL	0	0	0	4	0.00
IBUFDS	0	0	0	121	0.00
PHASER_OUT/PHASER_OUT_PHY	0	0	0	16	0.00
PHASER_IN/PHASER_IN_PHY	0	0	0	16	0.00
IDELAYE2/IDELAYE2_FINEDELAY	0	0	0	200	0.00
ILOGIC	0	0	0	125	0.00
OLOGIC	0	0	0	125	0.00

5. Clocking

Site Type	Used	Fixed	Prohibited	Available	Util%
BUFGCTRL	1	0	0	32	3.13
BUFIO	0	0	0	16	0.00
MMCME2_ADV	0	0	0	4	0.00
PLLE2_ADV	0	0	0	4	0.00
BUFMRCE	0	0	0	8	0.00
BUFHCE	0	0	0	72	0.00
BUFR	0	0	0	16	0.00

Clock (10 ns and 5 ns):

Since the MAC is synthesis as DSP in the FPGA, and the speed of DSP on the FPGA is very high. Therefore, the maximum delay path isn't the calculation, but the shift register. We can find out this situation in the timing reports.

A. Clock = 10 ns

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	8.731 ns	Worst Hold Slack (WHS):	0.152 ns	Worst Pulse Width Slack (WPWS):	
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TF	
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	
Total Number of Endpoints:	504	Total Number of Endpoints:	504	Total Number of Endpoints:	

All user specified timing constraints are met.

```
Max Delay Paths
--------------------------------------------------------------------------------
Slack (MET) :             8.731ns  (required time - arrival time)
  Source:                 inputbuffer_reg[9][13]/C
                            (rising edge-triggered cell FDCE clocked by clk  {rise@0.000ns fall@5.000ns period=10.000ns})
  Destination:            inputbuffer_reg[10][13]/D
                            (rising edge-triggered cell FDCE clocked by clk  {rise@0.000ns fall@5.000ns period=10.000ns})
  Path Group:             clk
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            10.000ns  (clk rise@10.000ns - clk rise@0.000ns)
  Data Path Delay:        0.869ns  (logic 0.478ns (55.006%)  route 0.391ns (44.994%))
  Logic Levels:           0
  Clock Path Skew:        -0.145ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    2.128ns = ( 12.128 - 10.000 )
    Source Clock Delay      (SCD):    2.456ns
    Clock Pessimism Removal (CPR):    0.184ns
  Clock Uncertainty:      0.035ns  ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE
    Total System Jitter     (TSJ):    0.071ns
    Total Input Jitter      (TIJ):    0.000ns
    Discrete Jitter          (DJ):    0.000ns
    Phase Error              (PE):    0.000ns
```

B. Clock = 5 ns

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	3.731 ns	Worst Hold Slack (WHS):	0.152 ns	Worst Pulse Width Slack (WPWS):	2.000
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	504	Total Number of Endpoints:	504	Total Number of Endpoints:	519

All user specified timing constraints are met.

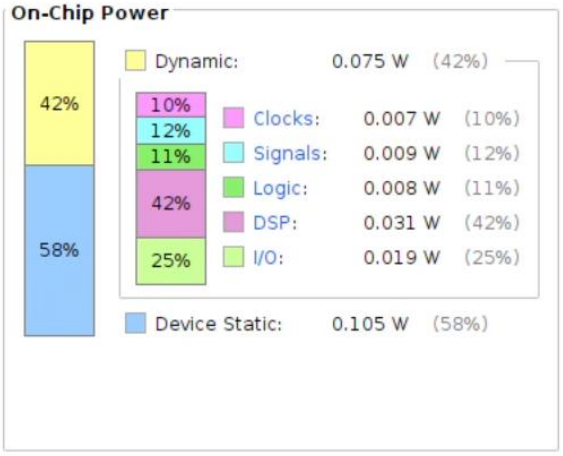
```
Max Delay Paths
--------------------------------------------------------------------------------
Slack (MET) :             3.731ns  (required time - arrival time)
  Source:                 inputbuffer_reg[9][13]/C
                            (rising edge-triggered cell FDCE clocked by clk  {rise@0.000ns fall@2.500ns period=5.000ns})
  Destination:            inputbuffer_reg[10][13]/D
                            (rising edge-triggered cell FDCE clocked by clk  {rise@0.000ns fall@2.500ns period=5.000ns})
  Path Group:             clk
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            5.000ns  (clk rise@5.000ns - clk rise@0.000ns)
  Data Path Delay:        0.869ns  (logic 0.478ns (55.006%)  route 0.391ns (44.994%))
  Logic Levels:           0
  Clock Path Skew:        -0.145ns (DCD - SCD + CPR)
    Destination Clock Delay (DCD):    2.128ns = ( 7.128 - 5.000 )
    Source Clock Delay      (SCD):    2.456ns
    Clock Pessimism Removal (CPR):    0.184ns
  Clock Uncertainty:      0.035ns  ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE
    Total System Jitter     (TSJ):    0.071ns
    Total Input Jitter      (TIJ):    0.000ns
    Discrete Jitter          (DJ):    0.000ns
    Phase Error              (PE):    0.000ns
```

Power:

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.181 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**27.1°C**
Thermal Margin:	57.9°C (4.9 W)
Effective θJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity



Schematic:

RTL Schematic:

Synthesis Schematic: