이 글은 혹시나 노심초사 하는 마음으로 작성하는 것이며, 자신의 방법이 있다면 그대로 해도 좋다. 하지만 결과는 같이 나와야할 것이다.

1. 세팅

이 글과 함께보내는 defaut\_constraint.tcl을 다시 업데이트 하기 바란다. 이번에는 진짜 확실하고, 아예 외부 공급 클락할 칩의 데이터 시트를 기반으로 짠 constraints다.

혹시나 모르는 사람이 있을까봐 말한다.

나는 리눅스에 Front\_End 파일만 왔다갔다 하므로 그 기준으로 이야기하겠다.

1) 자기 운영체제에서 Front\_End/Design\_Compiler/tcl/ 디렉토리에 defautl\_constraint를 리플레이스 하든 복붙을 하든 지금 파일을 사용한다.

2) Front\_End 압축

3) ftp 이용 - 자기 홈서버에 Front\_End.zip이나 압축 파일 확장자를 보낸다 (나는 Front\_End.zip으로 압축)

4) 서버 접속

5) 자기 홈 디렉에 unzip Front\_End.zip 입력

6) cd Front\_End/Design\_Compiler

이제 DC로 클락 최적화를 할 구간이다.

2. constraints 적용

자기 파일 .tcl 을 들어가자 ex) vim ./tcl/MPR121/mpr121\_controller/mpr121\_controller.tcl

나는 float adder이므로 그 기준으로 이야기하겠다.

vim ./tcl/ADS1292/ADS1292\_Filter/float\_adder/float\_adder.tcl

열면 아마 내용은 다 똑같을 것이다. 자기 기준에 맞춰서 생각하길 바란다

# Script file for constraining float\_adder

set design "float\_adder"

set dir "ADS1292/ADS1292\_Filter/float\_adder/"

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo " "

echo " ${design}.tcl "

echo " "

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

set\_svf "${svf\_path}${dir}${t\_w\_path}${design}.svf"

define\_design\_lib khu\_sensor\_lib -path ./lib

analyze -format verilog "${src\_path}ADS1292/ADS1292\_Filter/Float/${design}.v" -lib khu\_sensor\_lib

elaborate float\_adder -lib khu\_sensor\_lib

current\_design $design

# The link command locates the reference for each cell in the design.

link

source "${tcl\_path}default\_constraints.tcl"

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo " "

echo " Apply ${design}\_constraints.tcl "

echo " "

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

set clk\_main\_period 10

create\_clock -name $clk\_name -period $clk\_main\_period [get\_ports i\_CLK]

set\_dont\_touch\_network [get\_clocks clk]

set\_input\_delay -max 0.31 [all\_inputs] -clock $clk\_name

set\_output\_delay -max 0.14 [all\_outputs] -clock $clk\_name

remove\_input\_delay [get\_ports i\_CLK]

remove\_input\_delay [get\_ports i\_RST]

set\_cost\_priority -delay

set\_fix\_hold [get\_clocks clk]

set timing\_disable\_clock\_gating\_checks true

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo " "

echo " compile\_ultra ${design} "

echo " "

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

compile\_ultra -no\_autoungroup -incremental

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

echo " "

echo " write ${design} output file "

echo " "

echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

change\_names -rules verilog -hierarchy -verbose

write\_file -format verilog -hierarchy -output "${netlist\_path}${dir}${t\_w\_path}${design}.vg"

write\_sdf "${db\_path}${dir}${t\_w\_path}${design}.sdf"

write\_sdc "${db\_path}${dir}${t\_w\_path}${design}.sdc"

write\_parasitics -output "${db\_path}${dir}${t\_w\_path}${design}\_parasitics"

source "${tcl\_path}report.tcl"

이렇게 나오는데 여기서 빨간색을 친 부분만 수정을 할 것이다.

이전과 다른점은 이제 클락 올릴때 자기 파일의 clk\_main\_period 옆에 있는 숫자를 바꾼다. (50MHz - 20 100MHz - 10)

특히

create\_clock -name $clk\_name -period $clk\_main\_period [get\_ports i\_CLK]

set\_dont\_touch\_network [get\_clocks clk]

이거 두개는 무슨 일 이 있더라도 지우면 안된다.

내가 하는 방법을 설명하겠다.

1) 일단 clk\_main\_period 를 한계인 10으로 맞춘다.

2) 그리고 특별한 constraints 없이 돌린다.

3) check\_timing, report\_clock, report\_clock -skew, report\_clock\_gating, report\_clock\_gating\_check, report\_attribute, report\_constraint -all\_violators -verbose, report\_timing -delay\_value max, report\_timing -delay\_value min

이렇게 확인한다. 이게 좀 많아 보일 수 있는데 확실하게 해야된다는 마음에 체크를 하는거다. 종류가 많아 보일 수도 있지만 설명을 할테니 읽고 이해 하길 바라고 이 글이 도움 되고 무의미 하지 않았으면 좋겠다.

1) check\_timing

constrained 되지 않는 timing path나 clock-gate logic이 있는지 알려준다.

clock-gate logic은 많이 나올텐데 쉽게 말해 간단한 예로 clk이랑 enable신호랑 and로 묶여서 그 결과를 클락으로 쓴다면 clock-gate logic이 있는 것이다. 하지만 우리는 지금은 retiming 하지 않으므로 그런 건 없을 거라 추정된다. 여기서 주의깊게 봐야할건 warning, error 시그널이고 이게 없어지도록 수정한다.

2) report\_clock

클락의 정보를 알려준다. 이 회로가 어떤 클락 주파수에 움직이게 합성되었는지 알려주는데 이를 통해 지정한 클락 주파수, 돈터치 네트워크, fix hold 가 제대로 먹고 있는지 알려준다.

A screenshot of a cell phone

Description automatically generated

보면 나는 10ns 주기의 50% duty cycle, dont touch network, 그리고 이따 설명할 fix hold가 제대로 먹고 있는것이 확인 된다. 클락 공급 포트도 i\_CLK로 잘 되있는게 나온다.

3) report\_clock -skew

클락의 딜레이를 알려준다. 이거는 본인 constraints 짤때는 클락 관련해서 주파수 조절하는거 말고는 건드리지 말아라. 어차피 지금 CTS는 ideal 한 클락임을 가정하고 해야 결과가 잘나오고 이후 p&R 단계에서 wire에 의한 RC extraction을 알아서 뽑아 줄거고 그 결과를 이용해서 다시한번 front end를 최적화 시키고 싶으면 그 파일을 constraint로 지정하면 된다. (이것을 Back-annotation)이라 한다. 결론은 걍 보기만 해라 지금 보여주는 화면이 나와야한다. 아무것도 건드리지 않았다면,

A screenshot of a cell phone

Description automatically generated

4) report\_clock\_gating, report\_clock\_gating\_check

위에서 말했듯 clock-gate logic이 있는지 알려준다. 대신 이건 더 자세히 알려주는 거다.

A picture containing sitting

Description automatically generated

이렇게 Number of clock gating element가 0이고 check가 None이면 자기 constraints에

set timing\_disable\_clock\_gating\_checks true

이 커맨드를 집어 넣어라. 그러면 컴파일 속도가 올라가는게 체감이 될 것이다.

5) report\_attribute

뜻과 가치 성질을 보여준다. 이는 클락과 리셋의 network가 잘 먹었는지 확인하려 보는것이다.

물론 클락 리셋 말고도 자기가 설정한 특정 셀, 특정 핀, 특정 포트의 constraint가 잘 들어갔는지도 확인 할 수 있다.



클락은 drc\_disabled, dont touch network 리셋은 dont touch , ideal이 잘 되어있다.

혹시나 궁금해 하는 사람이 있을까봐 적는다

dont touch network는 set\_dont\_touch 의 성질 부여랑은 다른 것이고 이가 의미 하는것은 버퍼를 아웃풋에 즉 back쪽으로 아무것도 안단다는것이다.

ideal network는 forwar, back 둘다 buffer를 안단다.

여기까진 나의 constraints가 잘 들어갔나 확인하는 단계고 이제부터 그 들어갔는데 문제가 일어났나? 라는 것을 파악하기 위한 단계이다

report\_constraint -all\_violators -verbose, , report\_timing -delay\_value max

6) report\_constraint -all\_violators -verbose

종합으로 다 알려주는 명령어다. 자기가 지정한 constraints를 컴파일 했을 때 충족 못한 것을 알려준다. 걍 보다보면 종합적으로 알려주는데 대신 이게 큰 기준이다. 여기 있는것만 고치면 다 고친 것이다. area, power는 우리의 고려대상이 아니고 timing만 고려하고 클락을 올렸을 시 가장 많이 오류가 나는게 timing적인 문제이다. 만약 아무 문제가 없다면 밑의 사진 처럼 나온다.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Met 는 meet의 과거 형으로 충족했다는 뜻이다. (정말 혹시나 해서 적는 거다 무시 ㄱ ㄱ)

7) report\_timing -delay\_value max

setup 타임과 관련있는 max delay다 흔히들 critical path라 불리며 이것만 통과하면 setup은 걱정이 없는 것이다. (가장 심한게 만족이 되면, 그 밑에도 다 만족하는 것은 당연한 것이다)

A picture containing sitting

Description automatically generated

A screenshot of a cell phone

Description automatically generated

보이는 것 같이 slack이 required - arrival 이다. 즉, 먼저(slack이 양수)가서 기다리면 setup이 충족하지 않을까? 그니까 정확히 저 패스를 처리하는데 9.75ns를 소모하는데, 그 다음 데이터는 7.06ns만에 도착하는 것이다, 그니까 다음 데이터를 처리하는 2.69ns 를 기다리는 것이니 setup시간으로 충분한 것이다. float\_adder는 100MHz를 줘도 2.69ns 가 남는다. 이런 경우 원래 더 올려도 되는데 어차피 패드가 100MHz가 한계이므로 다들 100MHz가 만족한다면 거기서 멈춰라

A screenshot of a cell phone

Description automatically generated

설명이 부족한 것 같아 자료를 가져온다 Sequetial\_Circuit ppt이다. tsetup <= Tc - tpcq- tpd

이다. 무슨 말인지 이해갔으면 좋겠다. 아까 말했듯이 먼저 끝나면 다음 에지를 기다리므로(아까는 그 다음 데이터가 도착한다는 표현을 썼음, arrival 뜻을 이해시키기 위하여) setup시간이 충분해지는것이다. 즉 아까 위의 경우에서 보면 t\_setup = 2.69ns라는 것이다.

tpcq+tpd의 실제 값이 arrival time이고 t\_setup이 삼성 라이브러리에서 제공하는 최소의 값이 있는데 그 것이 최소한 필요로하는 setup타임을 계산해서 required time까지는 도착해야해 라고 알려주는 것이다. 만약 정 진짜 이해가 안가면 나한테 전화나 직접 물어봐라

8) report\_timing -delay\_value min

A picture containing sitting

Description automatically generated

min은 hold와 관련이 있다. max 와는 반대로 arrival - required이다. 이 계산이 음수가 나온다면 그 것은 hold violation이 일어난 것이다. 매우 매우 심각한 문제이다. 이는 원래 코드를 고치거나 더 좋은 PLL을 쓰거나, 더 좋은 기술이 나오는 것을 기다릴 수 밖에 없다.

max랑 반대로 생각해보자 t\_cpd + t\_ccq > t\_hold다

A screenshot of a cell phone

Description automatically generated

이것도 마찬가지로 그림 없인 설명이 안될 것 같아 그림을 가져왔다

내가 말한 t\_cpd 는 combinational propagation delay의 약자로 이 그림에선 t\_cd 이다.

t\_ccq는 clock to q contamination clock to q delay로 이 그림 에서도 t\_ccq이다.

t\_cd + tccq >= t\_hold이다. 이해가 바로 갔으면 좋겠는데 혹시 몰라 더 풀어서 설명하자면

클락의 rising edge이후 더 늦게 도착해야하는게 그림 보면 바로 알 것이다. 그 이유가 hold time 때문ㅇ에 그런건데 만약 아까 말했듯 이 시간이 음수면 클락의 rising edge보다 빨리 도착하는 것인데 빨리 도착해서 rising edge전에 그 값이 만약 바뀐다면 데이터 하나가 손실되는 것이랑 같은 것이라고 생각 될 것이다. 즉 데이터 손실이 있는 것이다. 이러면 ㅎㅌㅊ 중의 ㅎㅌㅊ이다. 사실 상 이것은 엄청나게 중요한 개념이다. hold violation을 막기 위해 two phase ff가 나온 것이고 이 문제는 위에서도 말했지만 회로를 바꿔야한다. 딜레이를 넣고 이런게 안된다. 차라리 two phase clock을 넣어 고쳐야하는 수밖에 없다. 근데 엄청 중요한 만큼 사실 상 고치기도 힘들지만 그런 상황이 나오기도 힘들다.

다시 한번 정리하면 arrival 이 require보다 커야한다. 늦게 도착해서 삼성 라이브러리 STA가 정해준 그 hold 를 넘긴 시간을 계산한게 require다. ㅅㅂ 이해안가면 전화해라

그래서 지금까지 우리가 뭘 고쳐야하고 뭘 봐야하는지 알았다. 이제는 고칠 단계이다. 고칠 단계는

다양한 constarints가 있고 기본적인건 ./tcl/README/README\_CONS에 적었으니 알아서 보고 내가 이번에 에더랑 멀티플라이어 하면서 얻은 꿀팁에 대해 이야기해보려 한다.

set\_input\_delay -max 0.31 [all\_inputs] -clock $clk\_name

set\_input\_delay -max 0.33 [all\_inputs] -clock $clk\_name -add\_delay

set\_output\_delay -max 0.14 [all\_outputs] -clock $clk\_name

처음 돌렸을 때 setup 이 문제가 있으면 이거 보고 대충 어림 잡아 고쳐라 2ns의 여유가 삼성이 추천하는거니 저렇게 둔거고 이거 몇번 숫자 바꾸고 딜레이 보다보면 걍 안다 어떻게 고쳐야할지

저 개념만 알면 이제 숫자 놀음이다. 제발 어려워하지 말길 바란다.

만약 저렇게 input\_delay를 설정했으면

remove\_input\_delay [get\_ports i\_CLK]

remove\_input\_delay [get\_ports i\_RST]

이거 무조건 넣어라 아니면 클락이랑 리셋도 같이 딜레이 생긴다 제발 이거 무조건 넣어라

set\_cost\_priority -delay

set\_fix\_hold [get\_clocks clk]

기존 컴파일은 DRC 우선으로 컴파일 하지만 이 녀석 둘을 쓰면 timing을 cost에서 최우선으로 둔다. 우리는 DRC 에 해칠 일이 잘 없으므로 (default\_constraints.tcl) 만약 자기가 timing적 우선인 컴파일을 하고 싶으면 이거 두개를 추가해라 둘 중 하나만 추가해도된다 사실 그냥 둘 다 쓰는게 간지나서 썼다.

set\_min\_delay (path)

set\_max\_delay (path)

#set\_min\_delay -from [get\_cells a\_reg[31]] -to [get\_cells a\_s\_reg] -0.08

요거 두개로 hold violation 고칠 생각하지마라 이거는 asynchronous 신호에만 적용가능한 제약조건이다. 그니까 이건 고쳤다고 보여지는 꼼수지 실상 회로는 그렇게 돌아가지 않는다. 제발 다른방법 이 있다. 만약 synch인 path에 이걸 썼다면 그냥 회로는 터진거나 다름 없다. asynch적으로 저 path로 신호가 들어갈 일이 없기 때문이다. 제발 함부로 쓰지말고 pt\_shell이나 자기가 path보고 판단하고 쓰자

다른 방법을 소개하겠다 그룹 패쓰이다.

group\_path -name a\_reg -from [get\_cells a\_reg[31]] -to [get\_cells a\_s\_reg] -weight

weight 는 0 ~ 100이다 자기가 중요하다고 생각한 만큼 가중치를 주면 그 가중치 순서에 맞게 최적화를 시켜준다. 즉 만약 어떤 path 가 100이면 그 path만큼은 절대 violation이 생길 일이 없게 하고 그 담 부터는 샥샥샥 정해지는것이다. (violation 있을 수 있고 없을 수 있음)

아 근데 이거 hold violation이 너무 많이 뜨고 이거 얼마 하지도 않는데(0.01 ~ 0.2) 이거 고치려면 어케 하나고 물을 수 있다. 솔직히 이거 path다 정해주기 귀찮다.

그래서 최강의 방법은 하나 있다. 대신 이러면 컴파일 시간이 10분이 넘을 수 있다. 이건 최후의 보루이다. 만약 이걸 써도 안되면 걍 포기하고 클락을 조금 내려라

#set startpoints \

[add\_to\_collection [all\_input] \

[all\_registers -clock\_pins]]

#foreach\_in\_collection startpt $startpoints {

# set start\_pin [get\_object\_name $startpt]

# group\_path -name $start\_pin -from $start\_pin -critical\_range 3.0

#}

setup time 보강용이다. 즉 max delay 에서 violation났을 때다. 이거는 설명할게 많은데 걍 갖다 써라 이거 쓰면 input 관련 즉 clock to q 에서 clock과 input port에 대한 모든 path를 worst case를 3ns까지 문제 없도록 최대한 컴파일 해주는 constraints이다.

#set endpoints \

[add\_to\_collection [all\_outputs] \

[all\_registers -data\_pins]]

#foreach\_in\_collection endpt $endpoints {

# set end\_pin [get\_object\_name $endpt]

# group\_path -name $end\_pin -to $end\_pin -critical\_range 3.0

# }

hold time 보강용이다. 즉 min delay 에서 violation났을 때다.이거 쓰면 output 관련 즉 clock to q 에서 q와 output port에 대한 모든 path를 worst case를 3ns까지 문제 없도록 최대한 컴파일 해주는 constraints이다.

마지막으로 조건 다만족하면 formality돌려서 확인해라. 그리고 넘어가는 것이고

자기 top 모듈 밑에 sub module 있으면 design\_compiler에서 set\_dont\_touch sub\_module무조건 해라

그리고 포말리티는 오늘 윤형이가 hierarchy관련해서 말해줬으니 그거 따라라.

더 궁금하다. 아니면 진짜 여기서 어케 할지 모르겠다 이러면 진짜 전화줘라 이제 우리 시간 없다.