

AI의 시대 : Optislang을 이용한 SERDES Metal Model 구축 및 최적화

김태진 이사

Brandon Kim
Senior Principal Application Engineer
ANSYS KOREA

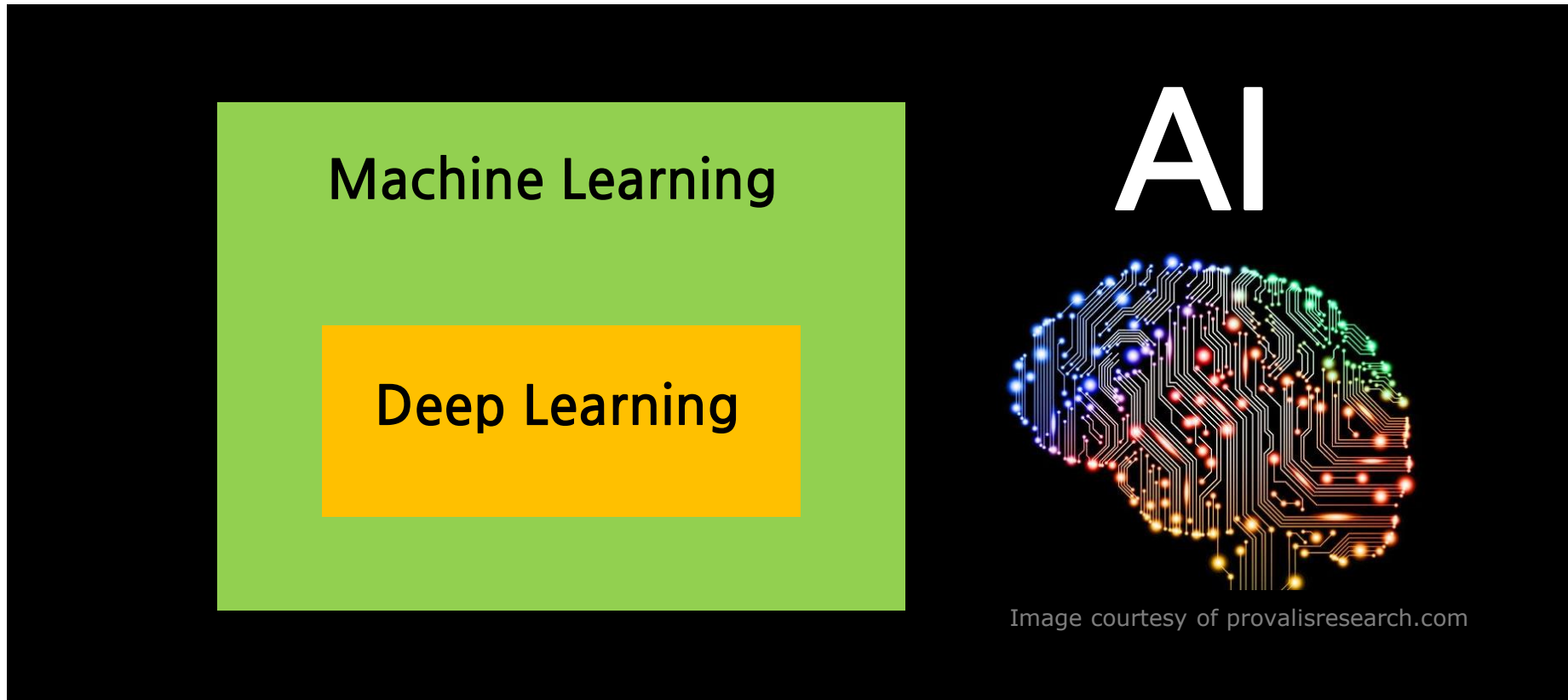


Agenda

1. AI & Machine Learning
2. ANSYS Optislang
3. Metal Model for SERDES PCB Optimization

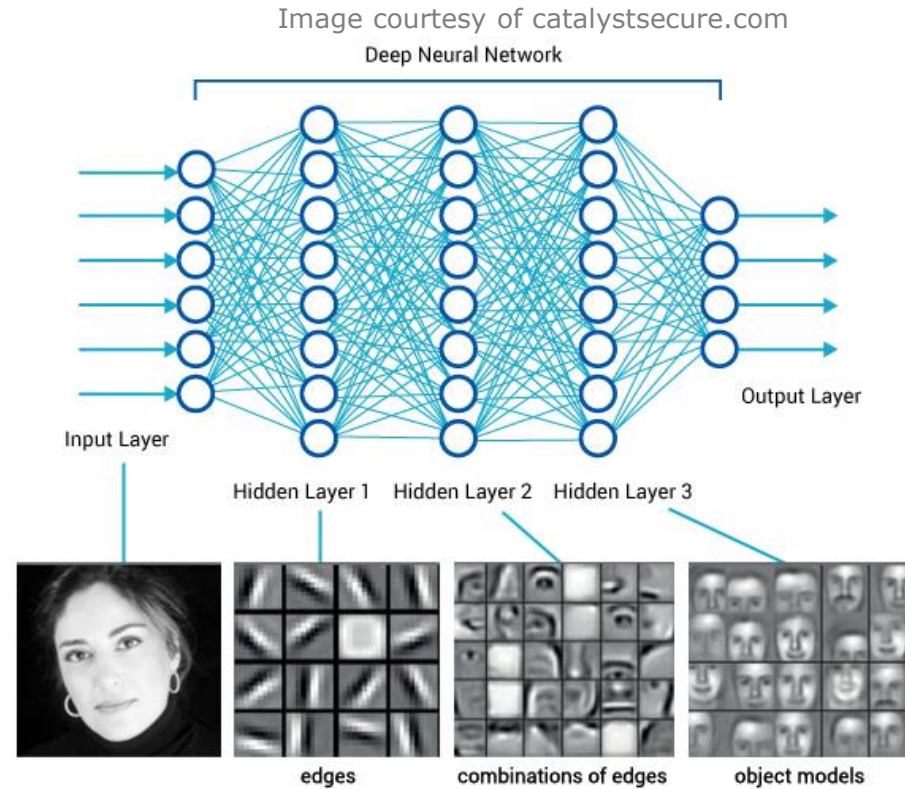
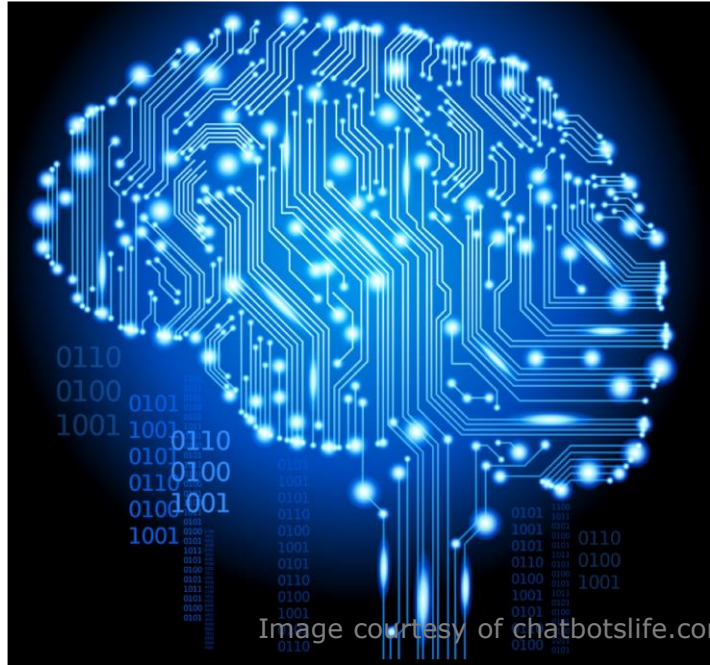
AI & Machine Learning

/ Machine Learning 이란?



- Machine Learning은 AI를 구현하기 위해 S/W를 훈련시키는 전략
- Deep Learning은 가장 대표적인 ML 알고리즘

Deep Learning



- 뇌의 신경망 (Neural network) 체계를 흉내 내어 구현된 학습기술
- 데이터에 내재된 공통된 패턴을 스스로 단계별로 분류하고 특징화
- **Machine Learning은 Big data에 기반한 deep learning을 통해 패턴과 규칙을 스스로 찾는 자기 학습 과정.**

Keywords of ML

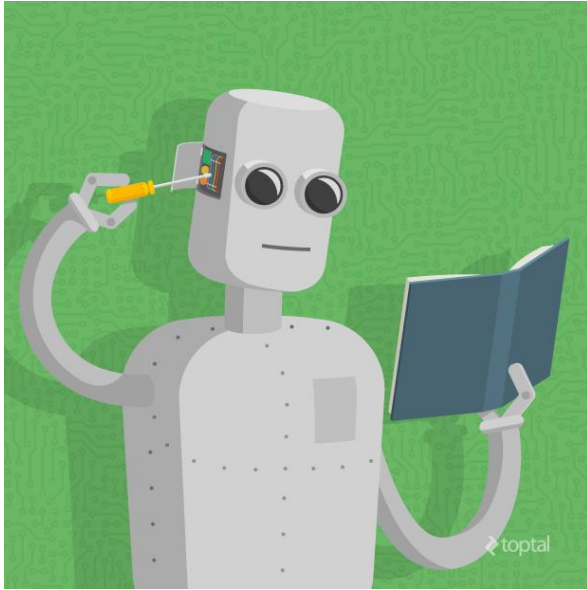


Image courtesy of toptal.com

1. 컴퓨터가 **자동**으로
2. 정규화된 **대규모 데이터**에서
3. 중요한 **패턴과 규칙**을 학습하고
4. 스스로 **판단** 및 예측을 수행

Automation

Big Data

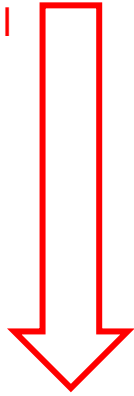
Pattern Study

Decision

/ ML의 핵심용어

Data Volume

small



Big

1. Supervised Learning (지도학습)

정해진 Input과 Output 데이터에 대한 학습

2. Unsupervised Learning (자율학습)

대량 데이터의 패턴, 특성, 구조를 스스로 분류 학습

3. Reinforcement Learning (증강학습)

학습 결과에 대한 feedback을 통해 보완/재학습

- 이 과정은 인간을 교육하는 과정과 완전히 동일하다는 점...

대표적인 ML tool 오픈소스 인기순위

Aggregate popularity (30•contrib + 10•issues + 5•forks)•1e-3			
#1:	97.53		tensorflow/tensorflow
#2:	71.11		BVLC/caffe
#3:	43.70		fchollet/keras
#4:	32.07		Theano/Theano
#5:	31.96		dmlc/mxnet
#6:	19.51		deeplearning4j/deeplearning4j
#7:	15.63		Microsoft/CNTK
#8:	13.90		torch/torch7
#9:	9.03		pfnnet/chainer
#10:	8.75		Lasagne/Lasagne
#11:	7.84		NVIDIA/DIGITS
#12:	7.83		mila-udem/blocks
#13:	5.95		karpathy/convnetjs
#14:	5.84		NervanaSystems/neon
#15:	4.91		tflearn/tflearn
#16:	3.28		amznlabs/amazon-dsstne
#17:	1.81		IDSIA/brainstorm
#18:	1.38		torchnet/torchnet

- 많은 종류의 ML tool kit이 무료로 공개되어 있음.
- ML을 구현함에 있어 다양한 알고리즘과 인터페이스가 존재
- 대부분 Python에 기반
- 현재는 Google의 TensorFlow가 대세

/ ML 관점에서 본 Simulation S/W의 장점

1. Big Data

- 시간과 machine 환경이 허락하는 한, 기계를 훈련시키는데 필요한 simulation data를 대량으로 만들어낼 수 있음

2. Quality of Data

- 특정한 목적을 가진 data를 만들어 낼 수 있기 때문에, 임의의 data를 분석해야 하는 일반적인 Big data에 비해 최적화가 빠르고 쉽다.

3. Normalization

- ML 의 performance는 데이터가 얼마나 정규화되어 있느냐도 관건인데, Simulation data는 대부분 명확하게 표준화되어 있어서 다루기 용이함

4. Automation

- Script 자동화를 통해 모델링 및 변수조절, 해석과 결과 export까지 자동화를 함으로써 일련의 수행과정 전체를 ML tool과 융합 가능

ML 구축 사례 : LMB (Load Model Builder)

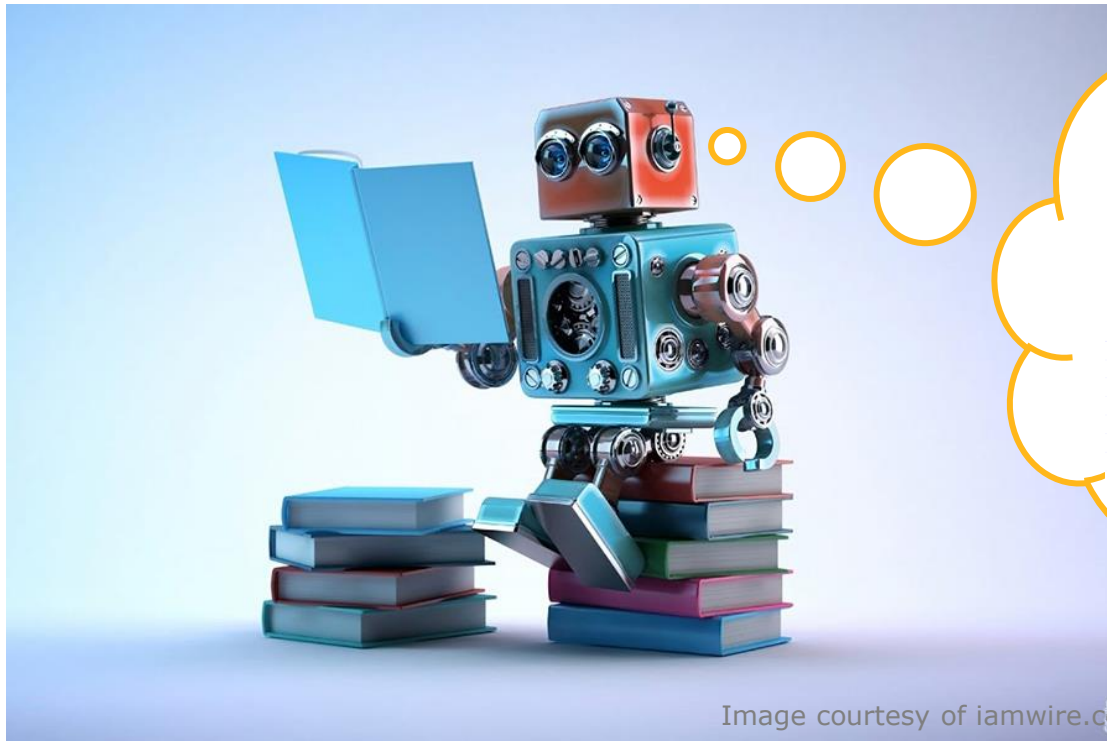
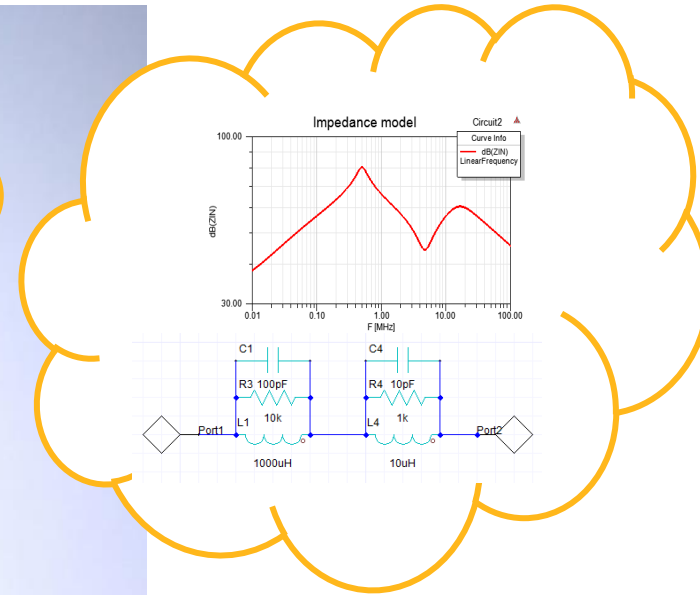


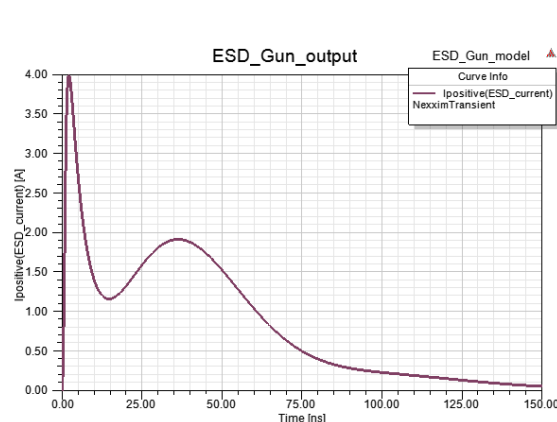
Image courtesy of iamwire.com



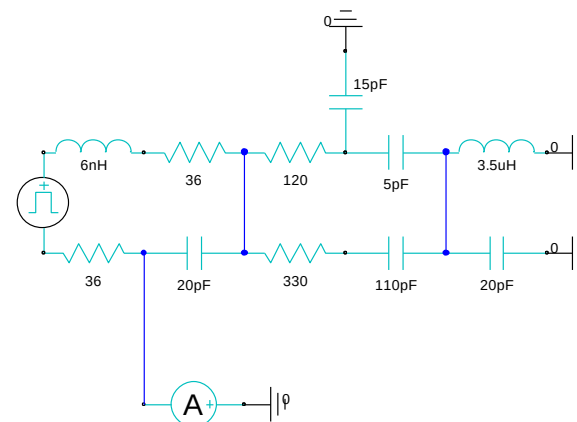
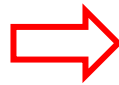
- Machine Learning 의 개념과 동작을 이해하기 위한 simulation 적용 사례
- 목표 : 특정한 임피던스 동작을 구현하는 등가회로를 자동으로 생성

등가회로의 필요성

- Circuit/System Simulation 에서 제조사로부터 특정한 모델을 받기 어려운 부품이 존재하는 경우, 파형 정보에 기반하여 동일한 회로적 동작을 수행하는 등가회로를 만들어야 하는 경우가 존재
 - System SI 분석 시 제조사에서 제공되지 않는 connector / cable / 특수소자 모델
 - CE 분석 시 motor/compressor 등의 전기소자 임피던스 모델
 - IV curve data에 기반한 반도체 device modeling
 - ESD / EOS / Surge 측정파형 모델 구축

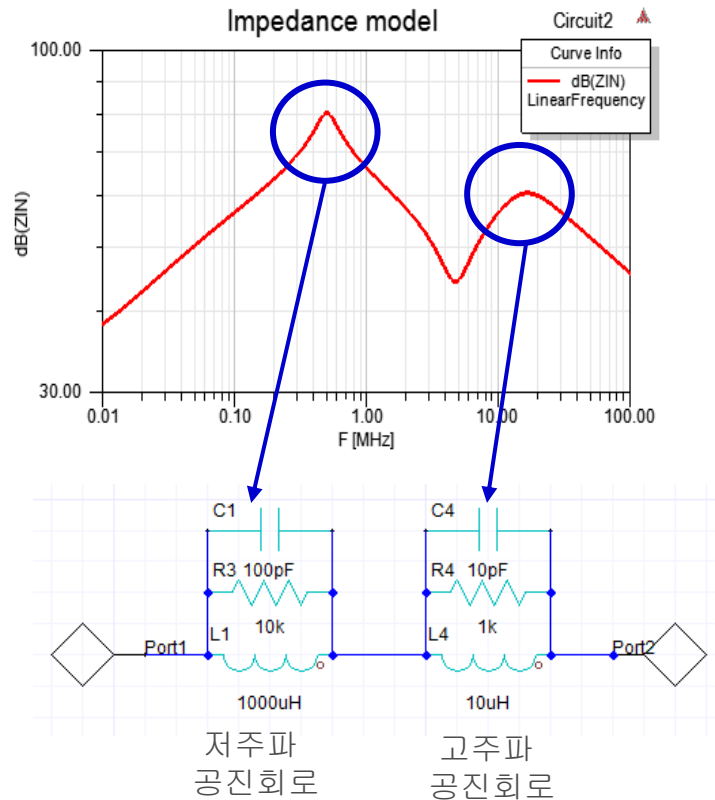


ESD Gun Waveform



Equivalent Circuit

기존의 등가 Load model 구축과정

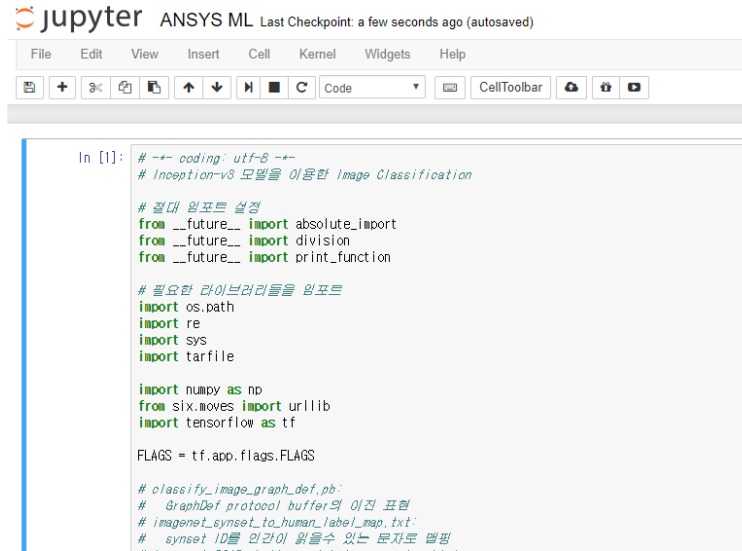
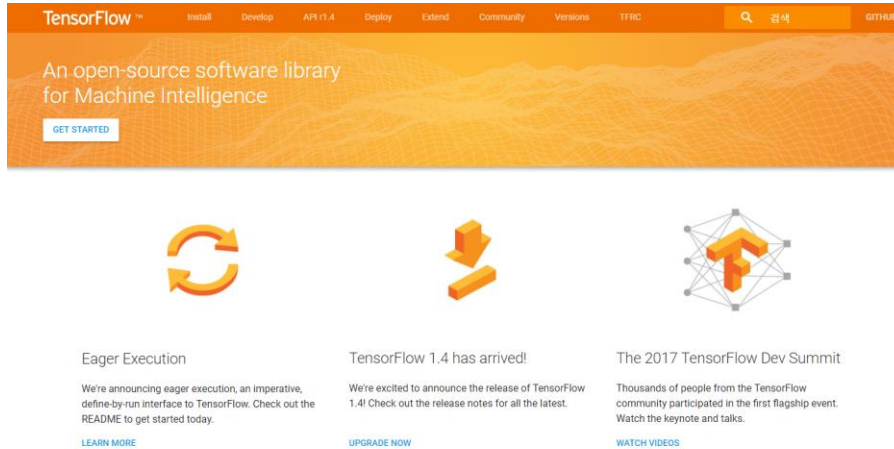


- **Step 1**
회로의 이해도가 높은 엔지니어가 파형정보를 기반으로 이론적으로 구현 가능한 회로 topology를 작성
- **Step 2**
유사한 파형이 나올 때까지 각 소자값을 무한 tuning
- **Step 3**
부족한 부분이나 정합이 잘 안되는 경우 회로를 추가/수정하면서 최적화

- 기본적으로 노가다 프로세스이며, 소요되는 시간이나 결과물의 수준은 엔지니어의 경험치에 크게 의존
- 어차피 **데이터에 기반한 판단 과정**인데, ML로 접근할 수 있지 않을까?

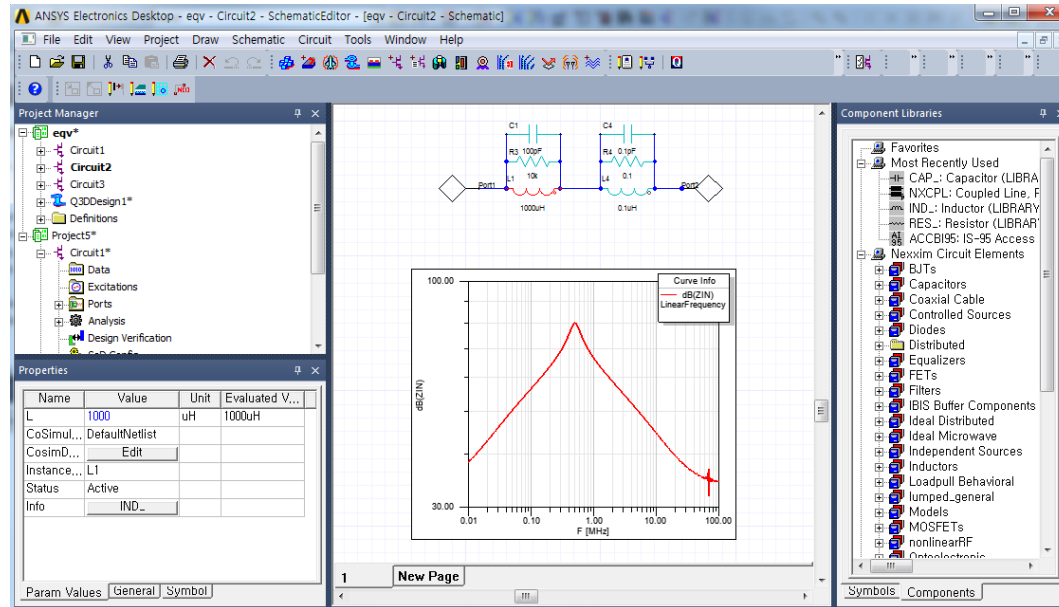


ML tool : Google TensorFlow



- 현재 가장 인기있는 ML tool 오픈소스
- Big Data의 pattern search에 최적화된 성능으로서, 입력 데이터를 다차원 배열로 사용.
- Simulation에서 도출되는 각종 그래프와 field pattern은 2D 또는 3D matrix 형태로 표현 가능
→ TensorFlow에 입력이 쉬움
- Python code로 되어 있어서 ANSYS python script와 찰떡궁합

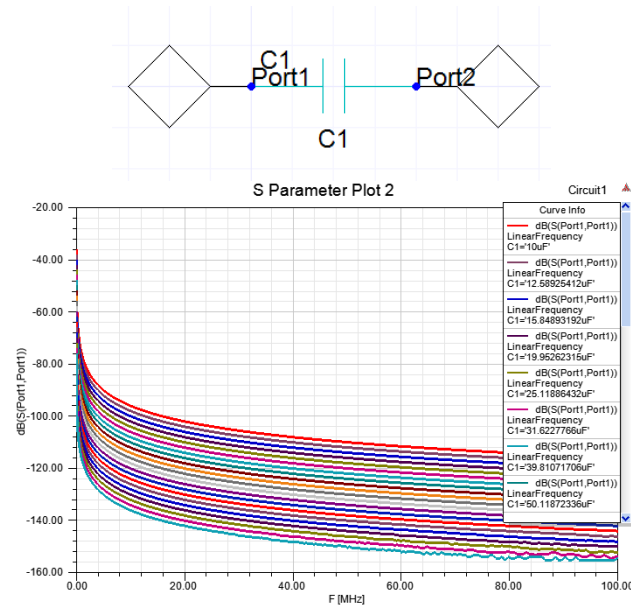
Automation for ML



```
# -----  
# Script Recorded by ANSYS Electronics Desktop Version 2017.2.0  
# 15:50:33 11 09, 2017  
# -----  
  
import ScriptEnv  
ScriptEnv.Initialize("Ansoft.ElectronicsDesktop")  
oDesktop.RestoreWindow()  
oProject = oDesktop.NewProject()  
oProject.InsertDesign("Circuit Design", "Circuit1", "", "")  
oDesign = oProject.SetActiveDesign("Circuit1")  
oEditor = oDesign.SetActiveEditor("SchematicEditor")  
oEditor.CreateComponent(  
  [  
    "NAME:ComponentProps",  
    "Name:=", "Nexxim Circuit Elements\\Capacitors:CAP_",  
    "Id:=", "1"  
  ],  
  [  
    "NAME:Attributes",  
    "Page:=", 1,  
    "X:=", 0.10668,  
    "Y:=", 0.0508,  
    "Angle:=", 0,  
    "Flip:=", False  
  ]  
)
```

- Designer를 이용하여 기본 회로도를 작성하고 SYZ 해석 수행
- 이 과정을 Python script로 자동화하여 소자값을 변수화하여 sweep하고, 결과값을 외부 파일로 export하는 자동화 코드 제작

Startup Test : Series Cap model



c1_10.csv
c1_20.csv
c1_30.csv
c1_40.csv
c1_50.csv
c1_60.csv
c1_70.csv
c1_80.csv
c1_90.csv
c1_100.csv
⋮

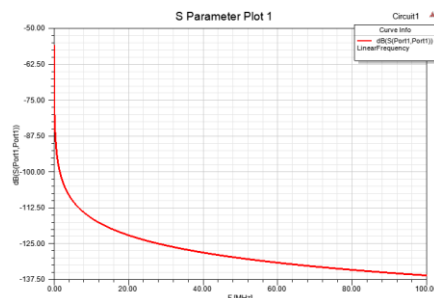
Training data set

일단 좀 가르쳐 주세요!



- 먼저 기본적인 회로를 이용하여 ML test를 시작.
- 직렬 cap 값을 sweep시키며 SYZ 해석 후 csv 파일로 export
→ ML tool이 이해하기 쉬운 **training data set** 형태로 구성
- 각 training data와 Cap값을 매칭시키며 ML tool의 지도학습 수행
(파일명에 cap값을 포함하면 간단한 구문분석으로 쉽게 처리 가능)

Series Cap model – 훈련결과 검증



100uF cap의 파형을 질문

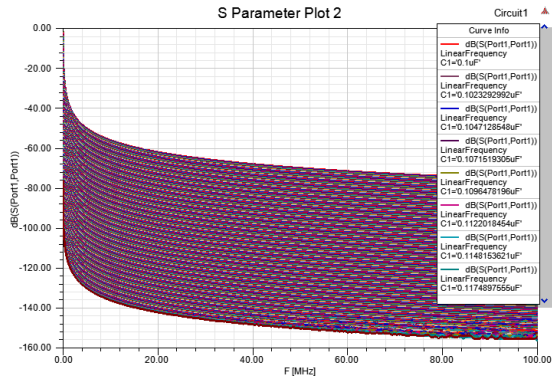


$C1 = 100\mu\text{F}$

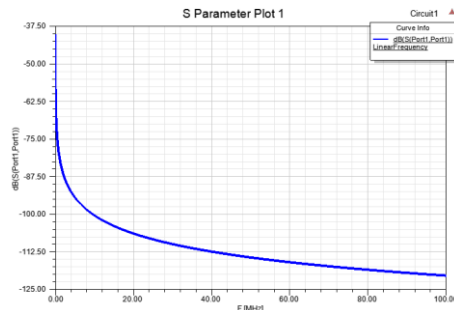
배운 대로 답해드리죠!

- 훈련된 ML tool에 입력했던 cap의 파형을 선택하여 입력하면 그에 해당하는 cap값을 출력값으로 도출
- 사람과 달리 ML tool은 모든 입력값을 암기하고 있기 때문에 즉각적으로 매칭되는 C값을 출력함.

Series Cap model – 진화의 시작



대량의 cap값 데이터로 맹훈련



Training data에 없는 파형을 질문



열심히 배우겠습니다!

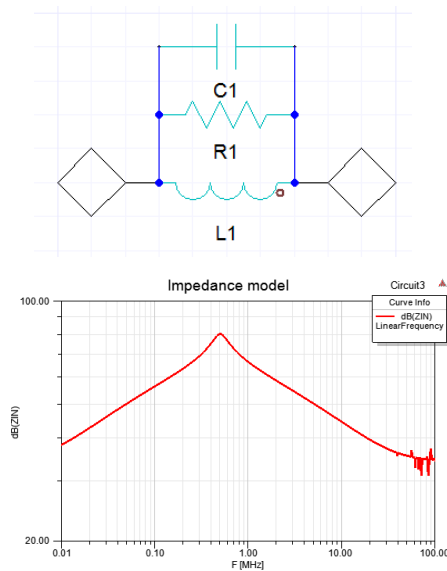


처음 보는 데이터긴 하지만 ...
대략 이정도쯤 될 것 같은데요?

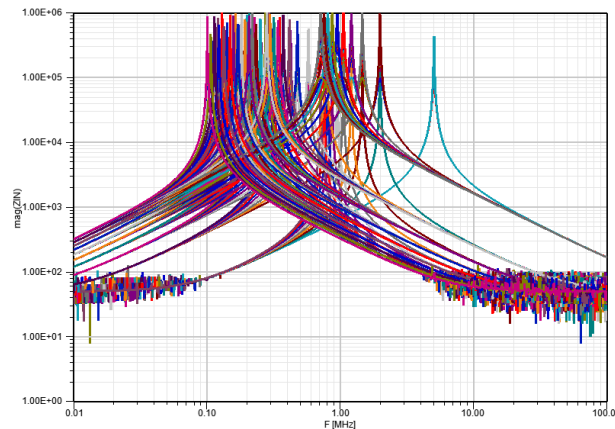
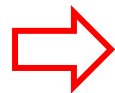
$C1 = 16.54\mu\text{F}$

- 대량의 추가 데이터로 훈련을 지속시켜 가다 보면, 훈련되지 않은 cap값의 파형을 입력해도 근사한 cap값을 스스로 계산해내기 시작
(training data가 증가면서 deep learning의 효과가 발현되기 시작)

Step 1 : 병렬공진회로 모델 훈련



병렬 RLC회로의 기본적인
임피던스 형태



R,L,C 값을 sweep시킨
대량의 data로 훈련

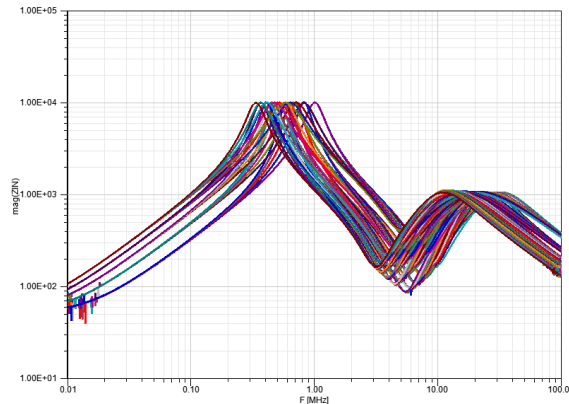
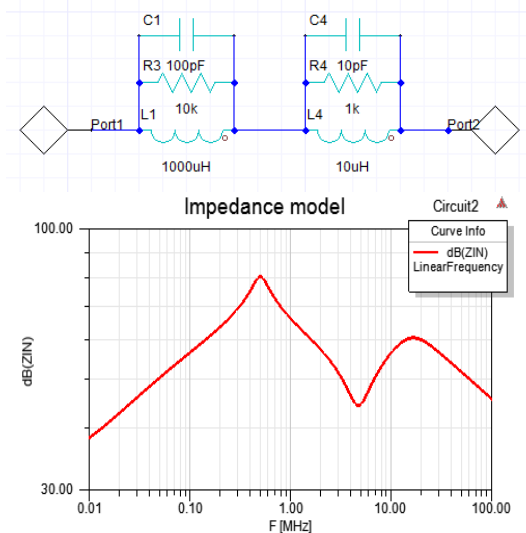


갑자기 복잡해졌네?



- 다양한 RLC 조합의 병렬 공진 회로 임피던스 데이터를 기반으로 훈련
- 데이터와 함께 동시에 그에 해당하는 3가지 label 값을 (R,L,C)를 인지하고 도출할 수 있도록 프로그래밍
- 밤새도록 sweep시키면 100만개 가까운 Big data set도 생성가능

Step 2 : 2중 병렬공진회로 모델 훈련

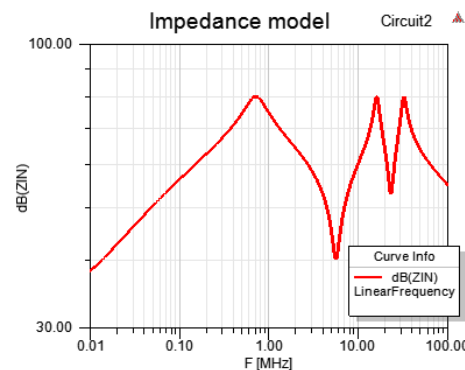
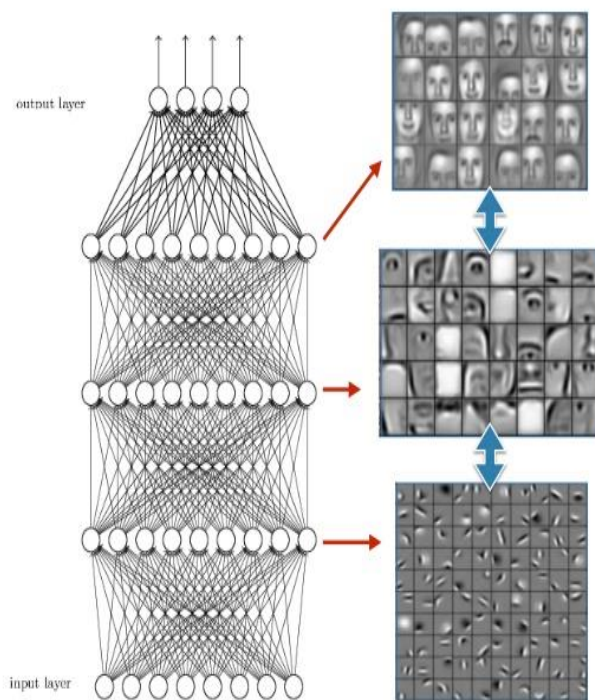


으음.. 공진회로가 2개가 되면
공진점도 2개가 되는구나..



- 2개의 공진점이 존재하는 회로모델에 대해 추가훈련
- 이 경우 변수가 많아지면서 data량이 기하급수적으로 늘어날 수 있지만, 이미 병렬 공진회로의 패턴 특성을 충분히 학습한 상태라면 적은 양의 training data set으로도 다중 공진의 특징을 스스로 분류 학습할 수 있음

Deep Learning Effect



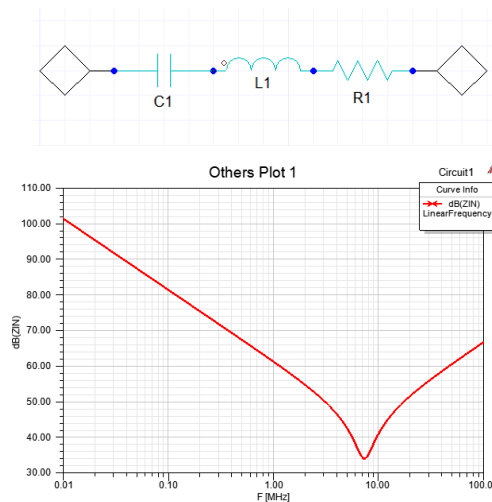
갑자기 공진점이
3개인 파형을 질문

처음 보는 형태의 그래프지만
볼록한 부분이 3 군데니까
공진회로가 3개 있어야겠네요.

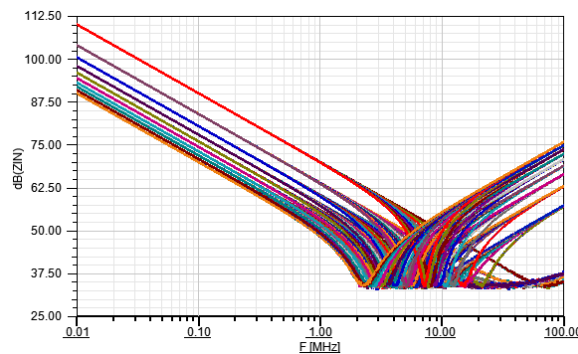


- ML의 핵심 알고리즘인 Deep Learning은 스스로 데이터의 패턴을 찾아내고 분류하면서 학습하는 과정
- 학습이 계속될수록 그래프의 공진 형태에 따라 회로 topology를 판단하는 능력도 점점 배양됨

Step 3 : 직렬공진회로 모델 훈련



직렬 RLC회로의 기본적인
임피던스 형태

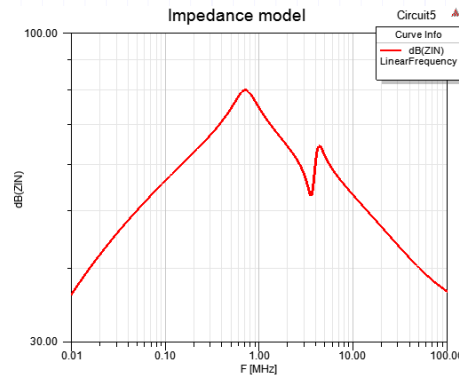
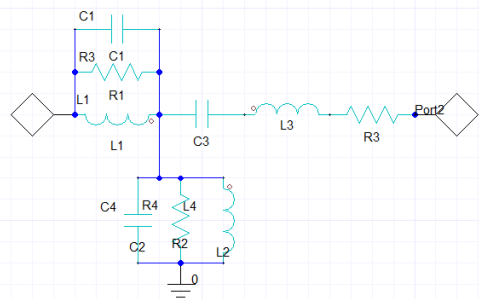
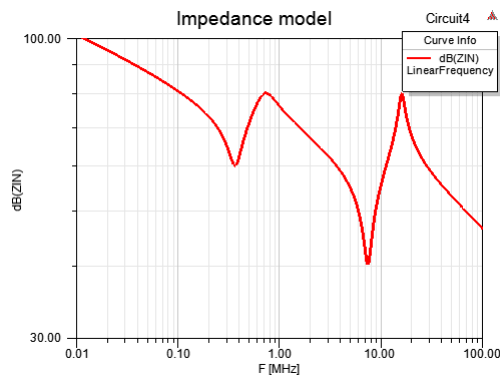
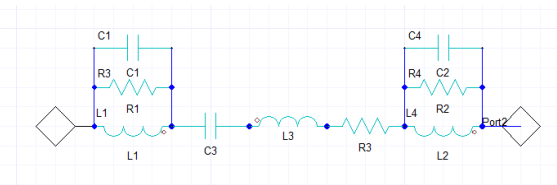


비슷한 듯 또 다르네..

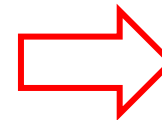


- 다양한 RLC 조합의 직렬 공진 회로 임피던스 데이터를 추가로 훈련
- 새로운 topology의 파형정보에 대한 패턴을 분석하고 학습

Step 4 : 복합회로 훈련

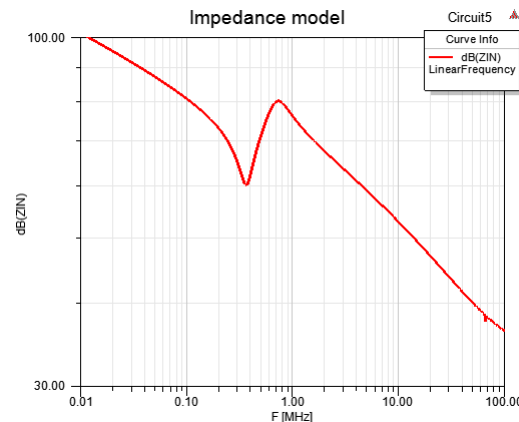


드루와~ 드루와~



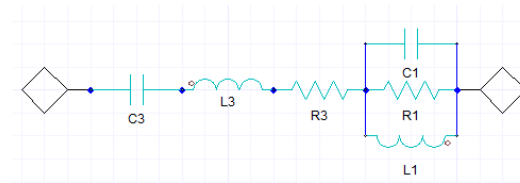
- 병렬공진과 직렬공진 회로의 특성을 충분히 학습한 후, 복합적인 회로에 대한 데이터를 훈련시켜서 다양한 상황에 대해 topology를 판단할 수 있도록 훈련

Step 5 : 실전 응용



측정된 임피던스 파형을
질문

훗 이까이꺼...



```
.param C1=5e-11
.param R1=10000
.param L1=0.001
.param C2=1e-11
.param R2=10000
.param L2=1e-05
.param C3=1.35e-10
.param L3=1.5e-09
.param R3=0.03
L9 Port2 net_13 L='L1'
R15 Port2 net_13 'R1'
C18 net_13 Port2 'C1'
R86 net_14 net_13 'R3'
L87 net_15 net_14 L='L3'
C88 Port1 net_15 'C3'
```

- 충분히 훈련이 진행된 후 실제 궁금한 임피던스 파형을 질문하고 ML tool이 스스로 회로를 구성하게 함.
- 판단 결과물을 SPICE netlist 형태로 도출하는 code를 추가하여 곧바로 해석에 사용가능한 회로모델로 사용 (schematic 까지 구현할 필요 없음)

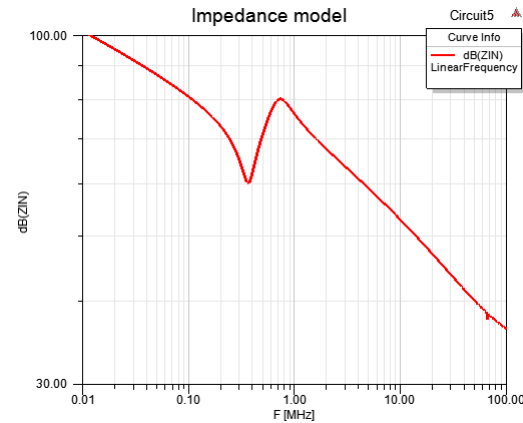
Step 6 : Self Verification / tuning

추측한 회로가 맞는지
제가 한번 검증해보겠습니다.



```
.param C1=5e-11  
.param R1=10000  
.param L1=0.001  
.param C2=1e-11  
.param R2=10000  
.param L2=1e-05  
.param C3=1.35e-10  
.param L3=1.5e-09  
.param R3=0.03  
L9 Port2 net_13 L='L1'  
R15 Port2 net_13 'R1'  
C18 net_13 Port2 'C1'  
R86 net_14 net_13 'R3'  
L87 net_15 net_14 L='L3'  
C88 Port1 net_15 'C3'
```

Simulation



스스로 simulation한 결과

- Simulation tool의 자동화 script를 integration하여, 스스로 예측한 회로도에 대해 simulation을 수행하여 자체 진단
- 입력 조건과 정합성이 부족한 경우, 일정 범위에서 주요 변수들에 대해 parameter sweep 을 수행하여 추가적인 tuning을 수행

Lesson

1. ML은 특정한 목적을 먼저 규정하고, 목표의 범위를 좁혀서 훈련시킬수록 높은 performance가 나온다. (만능이 아님)
2. 목표를 이루기 위해 어떤 데이터와 어떤 방법으로 훈련시키느냐가 ML performance의 관건.
3. ML 을 구현하는 알고리즘은 매우 다양하기 때문에, 어떤 알고리즘을 써야 원하는 목적에 가장 적합한지 검토하고 Test하는 과정이 중요. (실제 coding시 가장 어렵게 느껴지는 부분)
4. Deep Learning에 기반한 ML은 신경망 학습 알고리즘에 근거하여 스스로 대량의 데이터 패턴 속에서 공통된 부분을 분류하고 classify하기 때문에, **Data간의 새로운 연관관계를 찾아낼 수 있음.**
5. ML은 무에서 유를 창조하는 창의력이 아니라, **대량의 Data 의 특성에 숨어있는 속성을 찾아내어 인간에게 도움이 되는 통찰력을 제공하는 tool.**

Surrogate Model / Meta Model

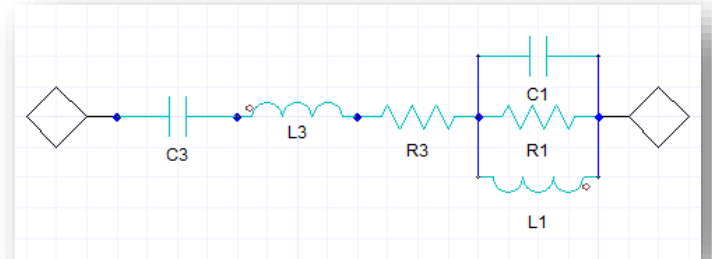
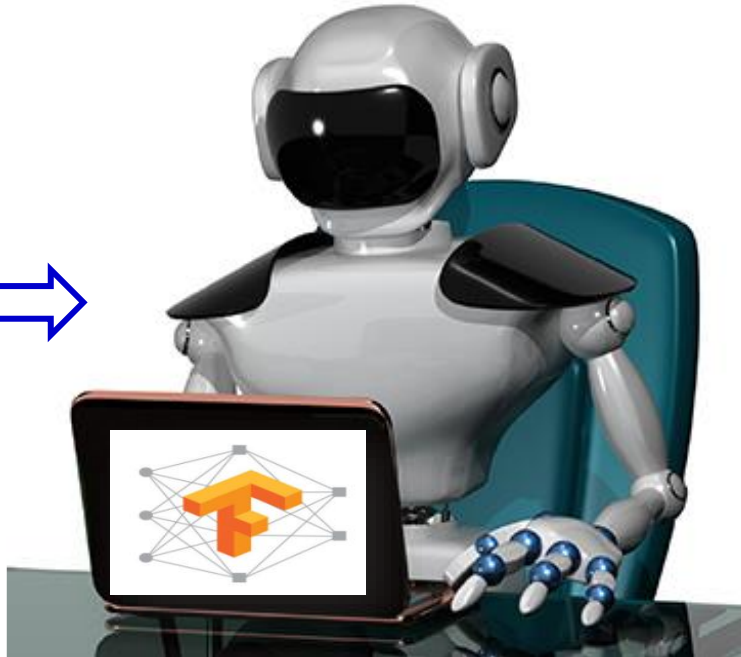
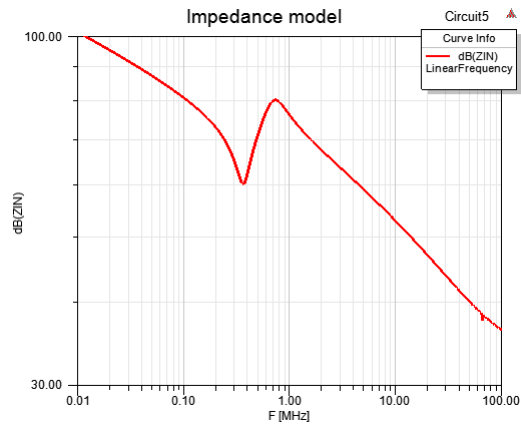


Image courtesy of devops.com

- 대량의 Input/Output Data 로 훈련된 Machine Learning Tool → AI
- AI는 결국 훈련된 대로 원하는 질문/목적을 던졌을 때 이를 위한 설계적 답을 도출
→ 다른 이름으로 **Surrogate Model** 또는 **Meta Model** 이라 부름

ANSYS Optislang



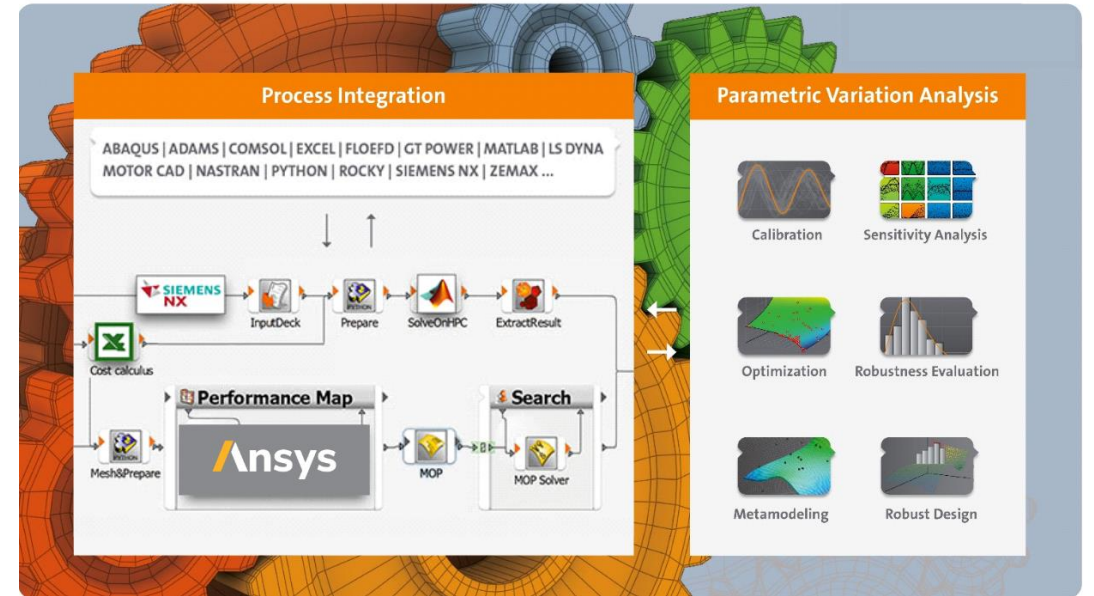
ANSYS Optislang

- 고성능 설계 최적화 Tool

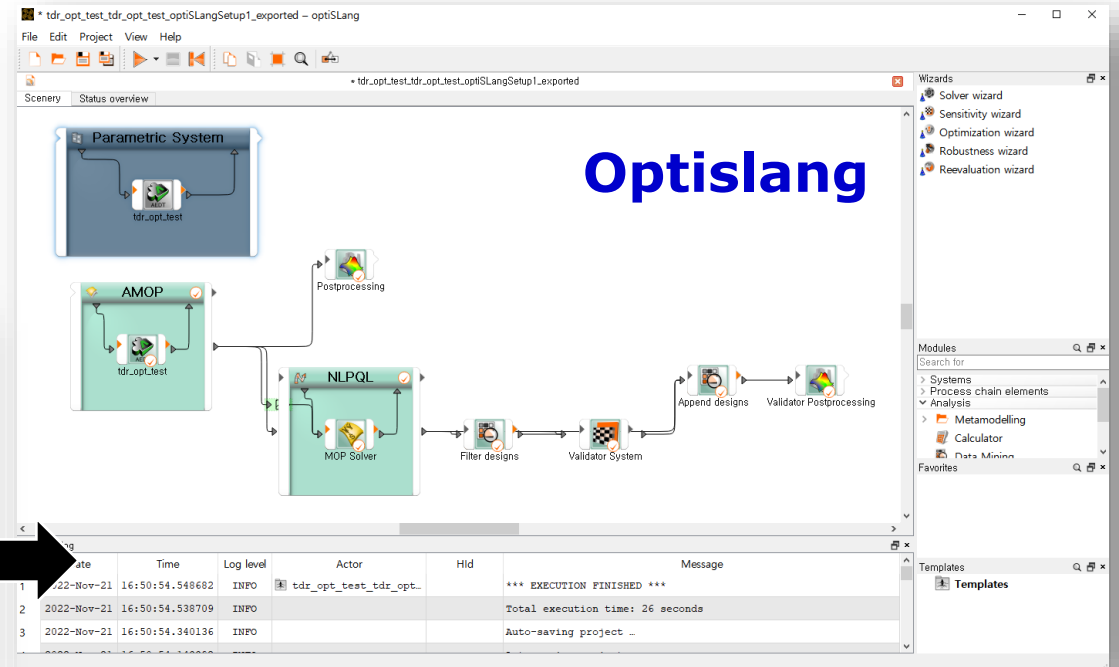
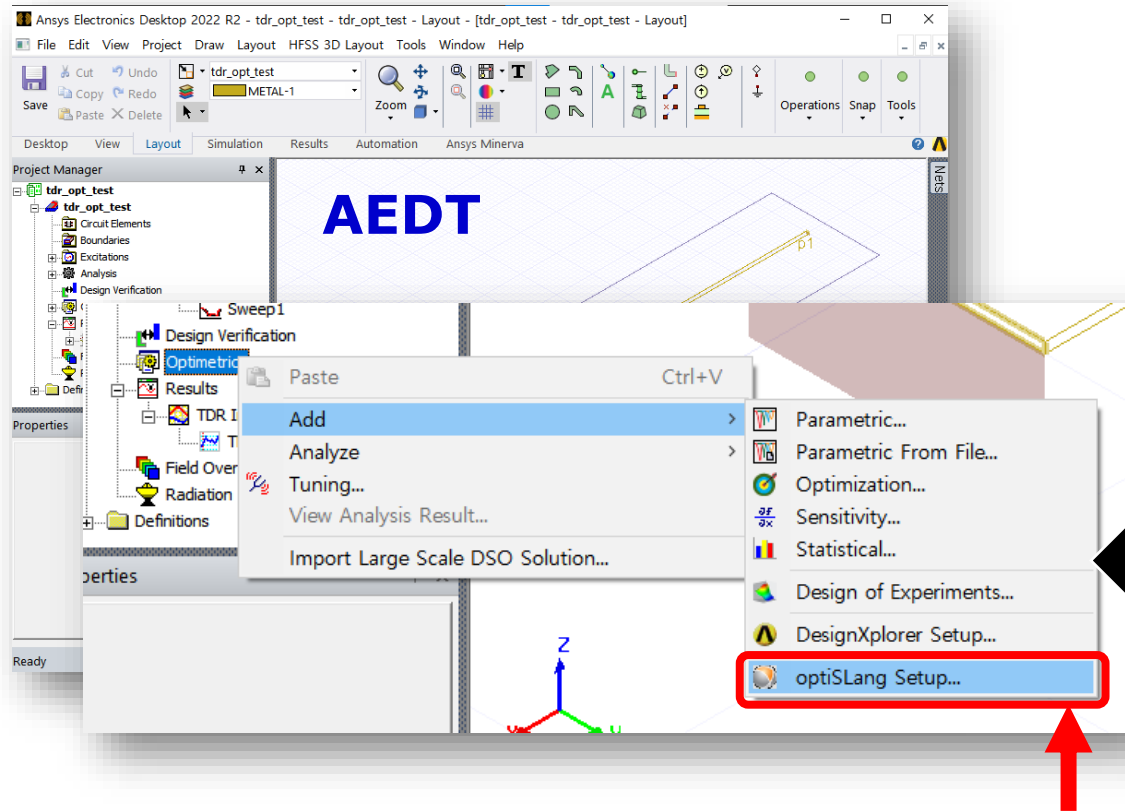
- Metamodel 기반의 고속 최적화 가능
- Deep Learning 등 최신의 AI 알고리즘 포함

- 장점

1. ANSYS Simulation tool과의 완벽한 호환성 (입/출력 추가 coding 불필요)
2. 설계물에 특성에 맞는 최적의 Metamodel 을 자동으로 생성
3. COP (R Square) 자동 계산 및 다양한 분석지표 제시



Optislang in AEDT

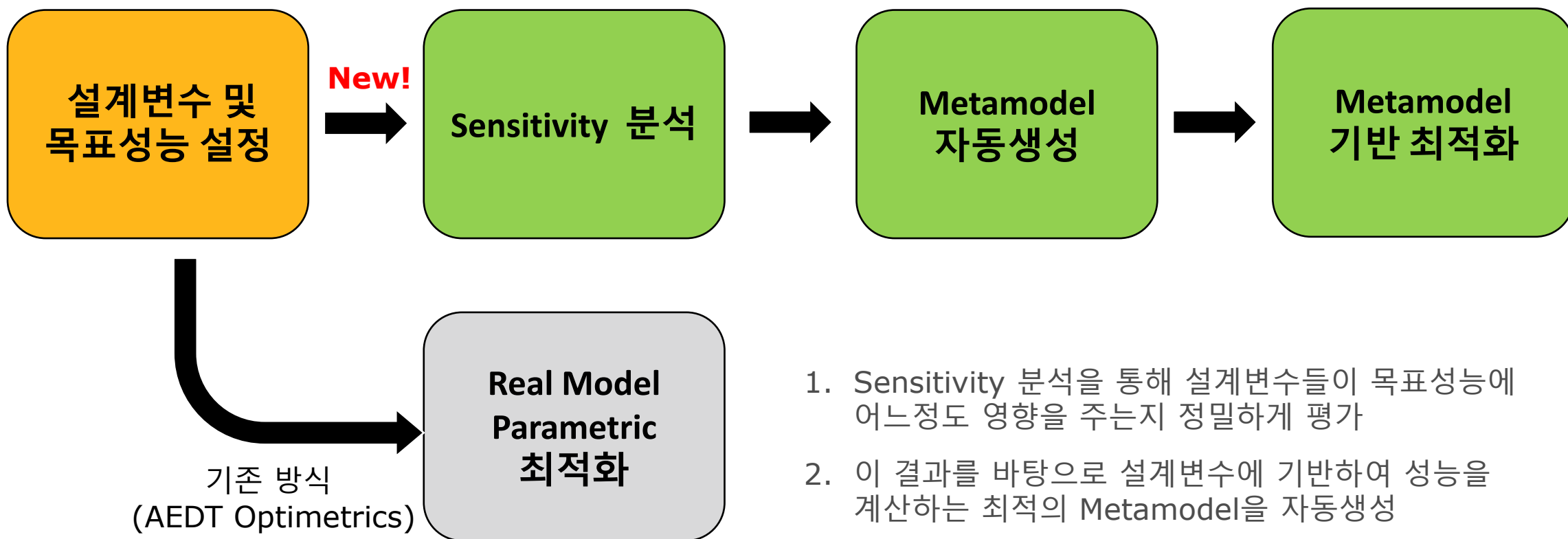


- AEDT (Ansys Electronic Desktop) 의 메뉴로 **Optislang Setup**이 Integration 되어 있음.
- 별도의 Input / Output 코딩 작업 없이 자동화된 연동 가능
- 적용가능 Tool : SIwave, HFSS, Q3D, Maxwell, Designer Circuit, Simplorer (Twin Builder)

/ 핵심 개념 : Metamodel

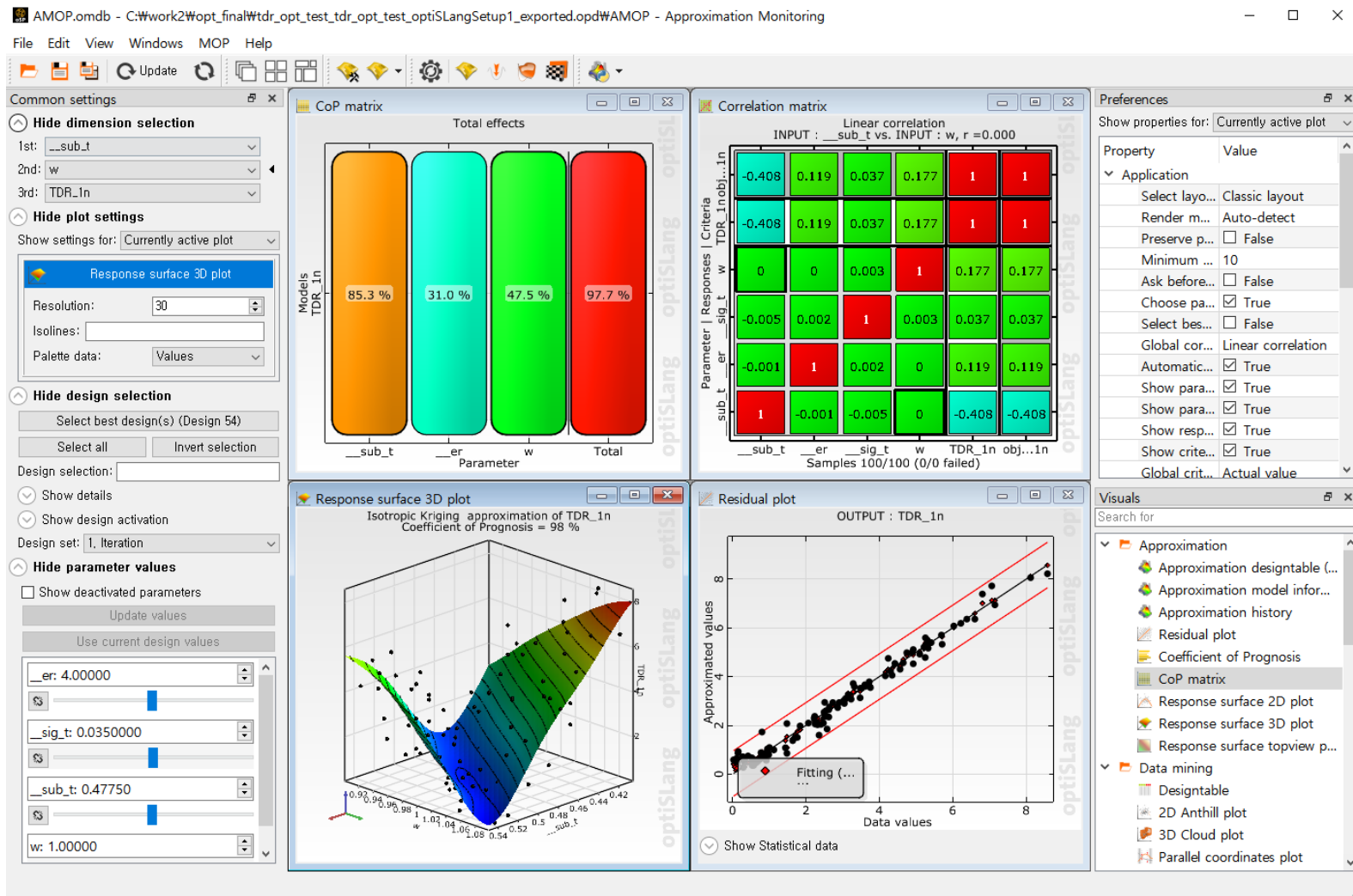
- 설계최적화를 위해서는, 목표하는 성능을 얻기 어떤 설계변수가 유효하고 또 어떤 변수범위에서 최적화를 수행할지 결정하는 것이 가장 중요함.
- 하지만 새로운 설계라든지, 설계변수가 너무 많은 경우 숙련된 엔지니어도 쉽지 않은 일.
- **Metamodel이란?**
 1. 각 변수 조합별 해석 결과를 수집한 Sensitivity 분석을 기반으로
 2. 입력된 변수들 중에서 유효한 설계변수와 변수범위를 추려내어
 3. 추가적인 Simulation 없이 입력 변수에 따른 성능을 예측할 수 있는 일종의 ROM(Reduced Order Model) 또는 Behavior Model / Surrogate Model 을 의미.
 4. Metamodel은 Analytic model이기 때문에 **수초 내에** 최적화 변수를 찾아낼 수 있음
- Optislang은 이러한 Metamodel을 자동으로 생성해주며, 이렇게 생성된 Metamodel을 **AMOP**(Adaptive **Metamodel** of Optimal Prognosis)라고 부릅니다.

/ Optislang : Metamodel 기반 최적화



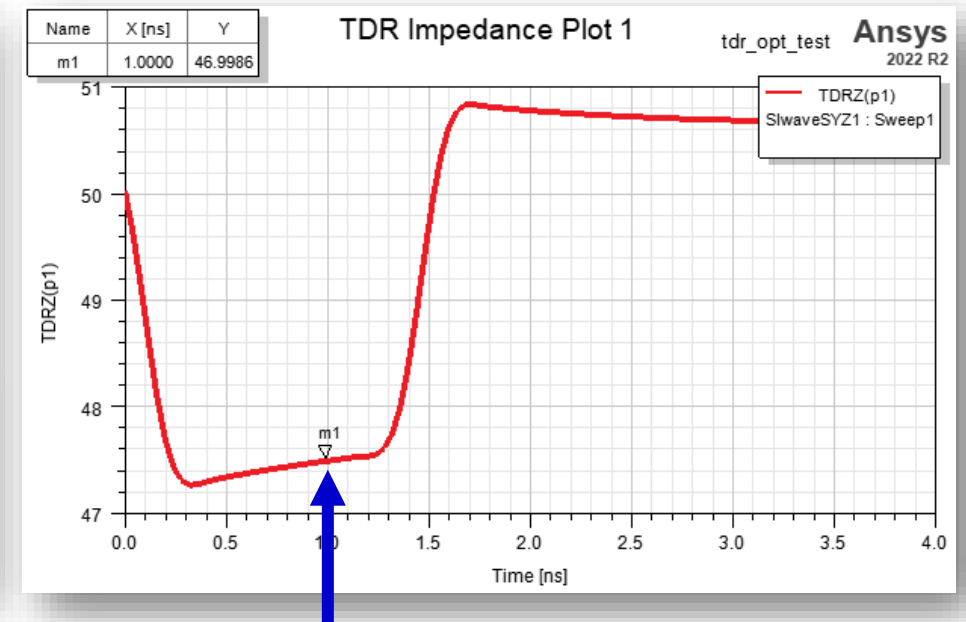
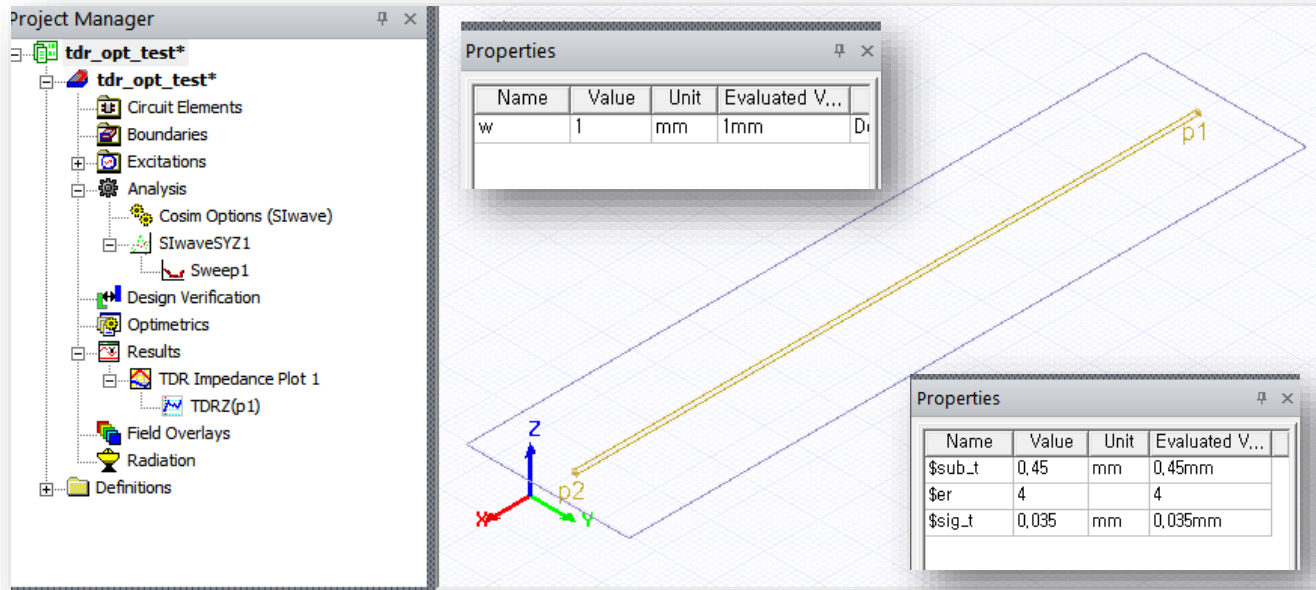
1. Sensitivity 분석을 통해 설계변수들이 목표성능에 어느정도 영향을 주는지 정밀하게 평가
2. 이 결과를 바탕으로 설계변수에 기반하여 성능을 계산하는 최적의 Metamodel을 자동생성
3. Metamodel 에 기반한 초고속 설계 최적화 수행
: 최적화 자체 수행시간은 통상 수 초 이내

Optislang : 다양한 최적화 평가지표



- Metamodel의 완성도를 가늠할 수 있는 COP (R Square) 자동계산
 - 최적의 목표성능을 구현하는 각 변수 별 조합을 2D/3D 그래프로 확인
 - 각 설계변수가 최종목표를 달성하는데 어느정도 기여하는지를 정량적으로 계산
- 설계자로 하여금 어떤 변수를 집중적으로 tuning해야 하는지 직관적으로 제시

간단예제 – PCB 선로의 임피던스 최적화

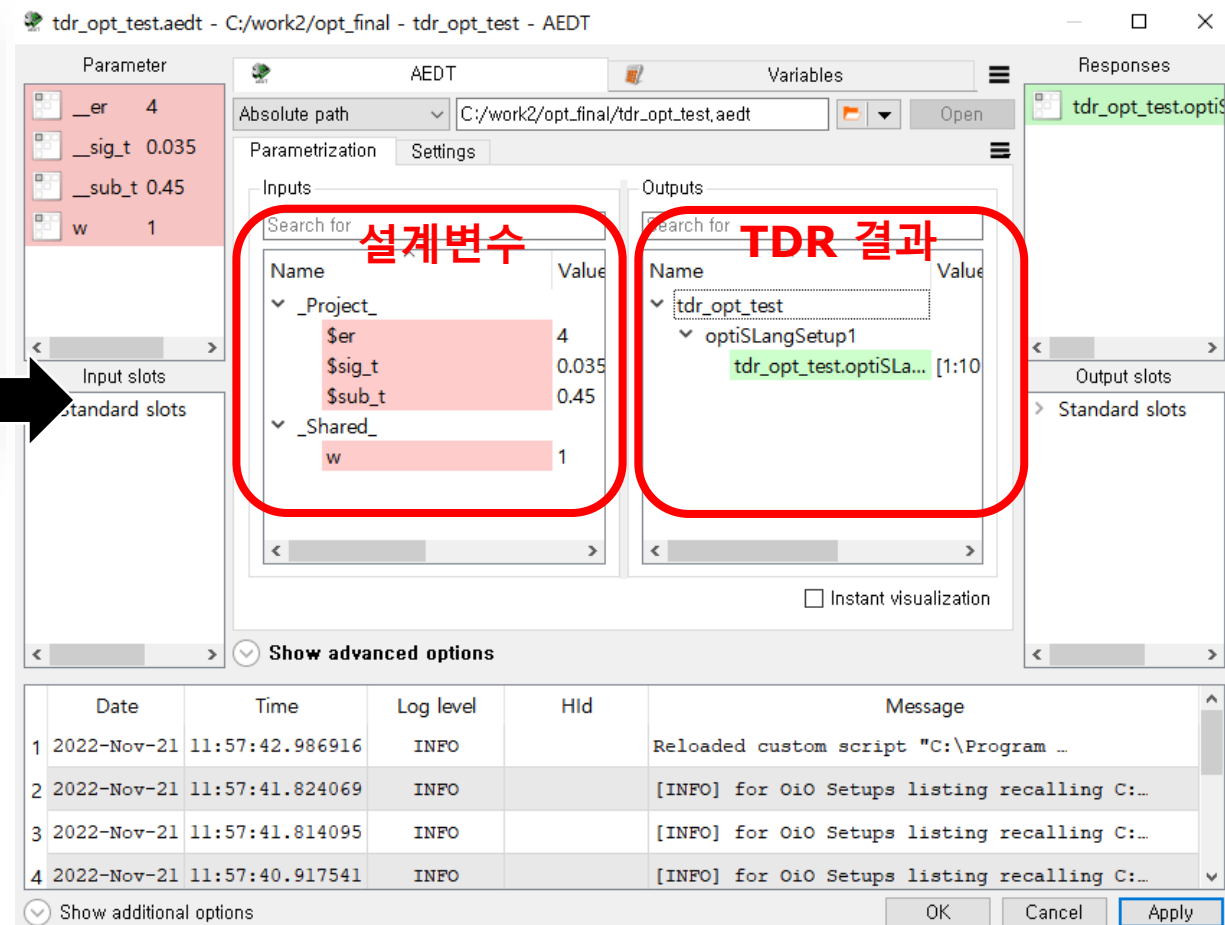
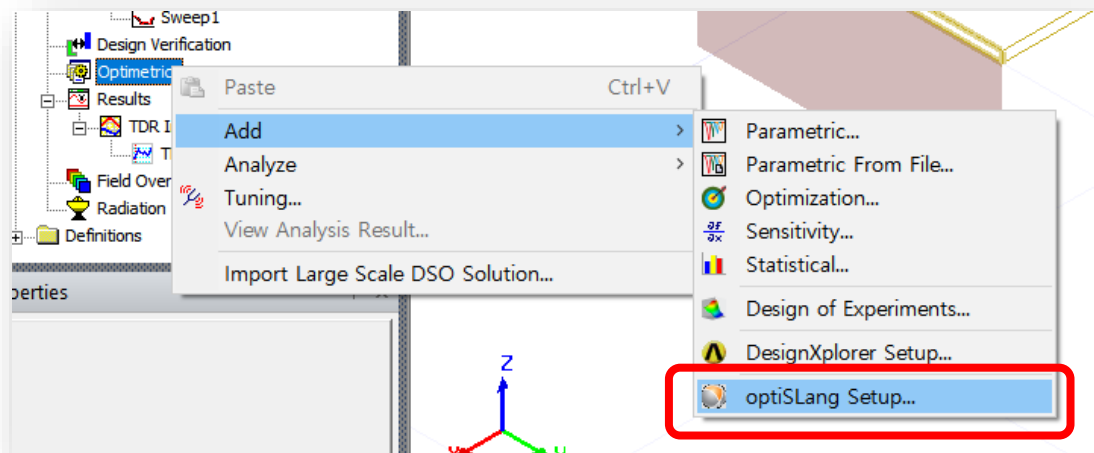


AEDT 설계변수 : trace 선폭, 유전체 두께, 유전율, 도체 두께

목표성능 : 1ns 지점의 임피던스를 50옴으로 최적화

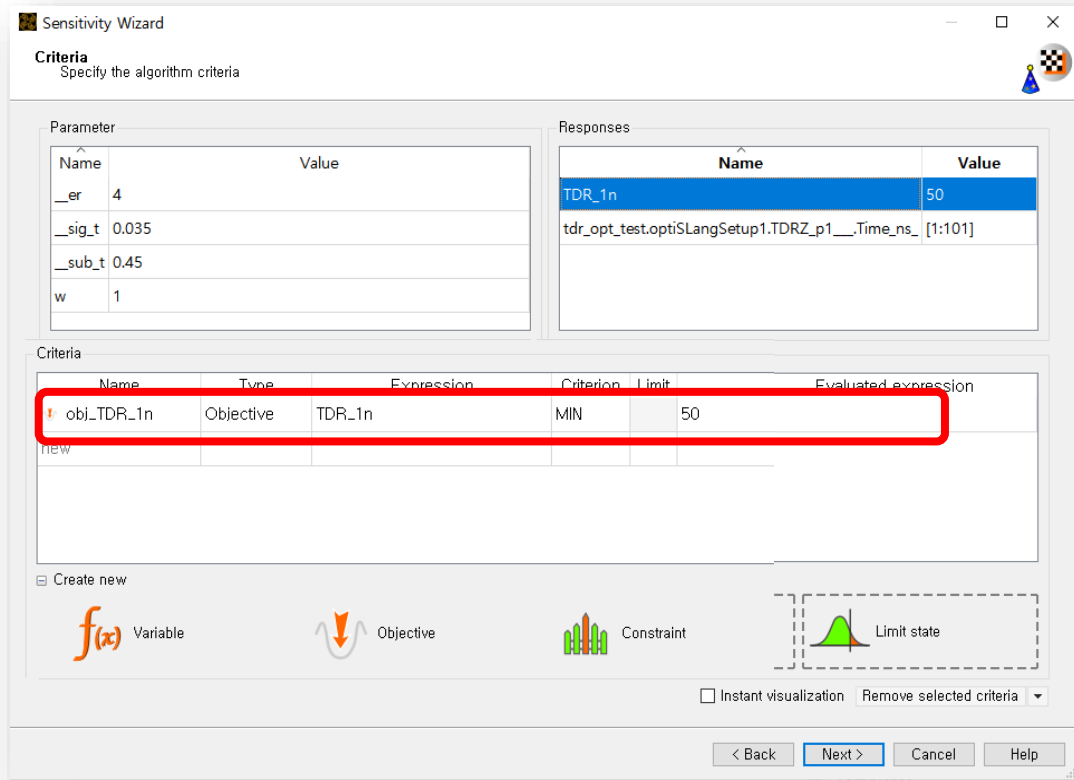
- AEDT의 HFSS 3D Layout GUI에서 SIwave solver를 이용하여 TDR 해석
- Trace의 임피던스를 50옴으로 최적화하는 예제를 통해 AEDT + Optislang 의 동작을 이해해봅니다.

AEDT 메뉴에서 Optislang Setup 실행

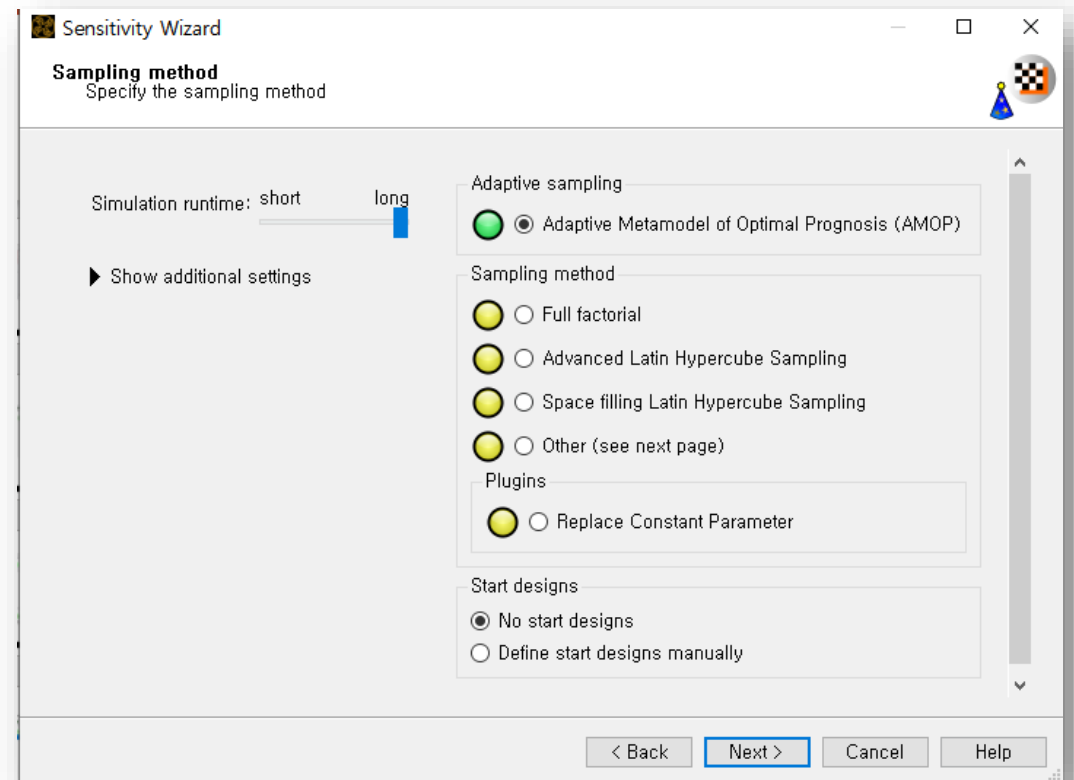
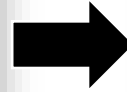


- AEDT 메뉴에서 Optislang Setup을 구동시키면 자동으로 설계변수와 목표성능에 해당하는 TDR 결과가 Optislang에 자동으로 입력됨 (별도의 코딩작업 불필요)

Optislang : Sensitivity 분석

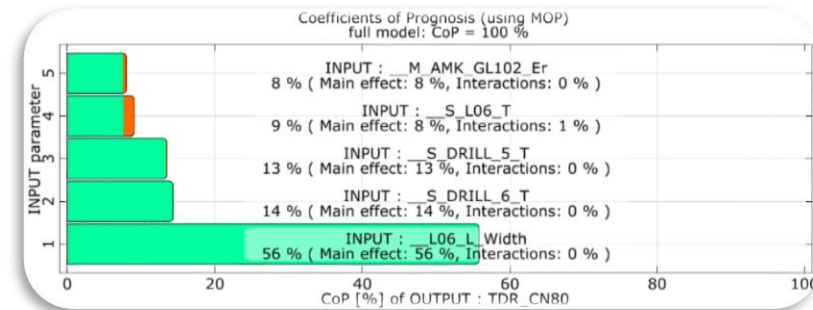
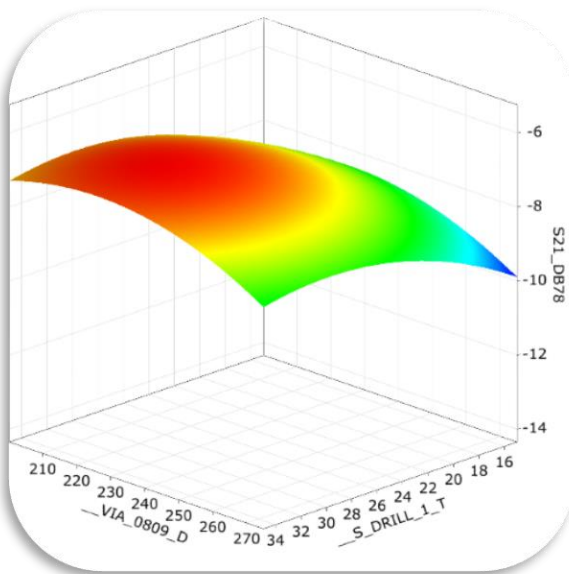
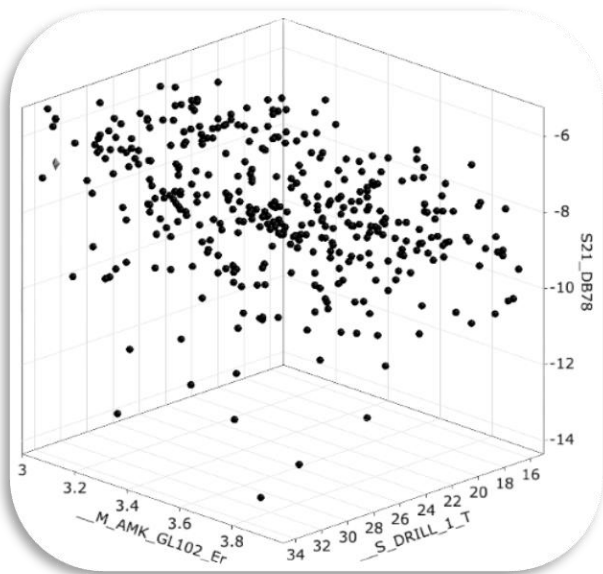


- 1ns의 TDR 임피던스를 50옴으로 목표성능 설정



- 설계대상에 최적화된 Metamodel 인 AMOP 를 자동으로 생성하는 Sensitivity 분석 수행

What is a Sensitivity Analysis in OptiSlang?

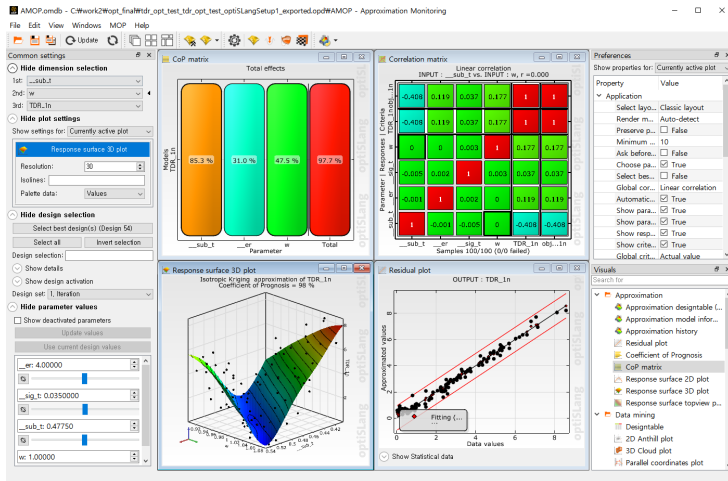


- 다양한 설계 변수 조합에 따른 특정한 해석결과 수집
- 각 설계변수 별로 해석결과에 미치는 영향 탐색
- 각 설계변수 간의 Trade off 관계 및 연관성 탐색

- 해석된 Big Data를 통해 Metal Model(MOP) 구축
- Meta Model은 복잡한 변수 입력 조합에 따른 모든 성능 결과를 포함하고 있으며, 해석되지 않은 변수 조합의 결과까지 예측

- Meta Model을 통해 각 변수가 성능결과에 얼마나 기여하는지 다양한 관점에서 제시
 - 작은 변화에도 큰 성능결과를 이끌어내는 설계변수는?
 - 성능결과에 크게 영향을 주지 않는 설계변수는?
 - 각 설계변수간의 조합이 성능결과에 미치는 정도는?

Sensitivity 분석결과의 예 - Correlation Matrix



Example

Sig_t 와 TDR_1n 결과는 0.04의
연관지표를 보여줌

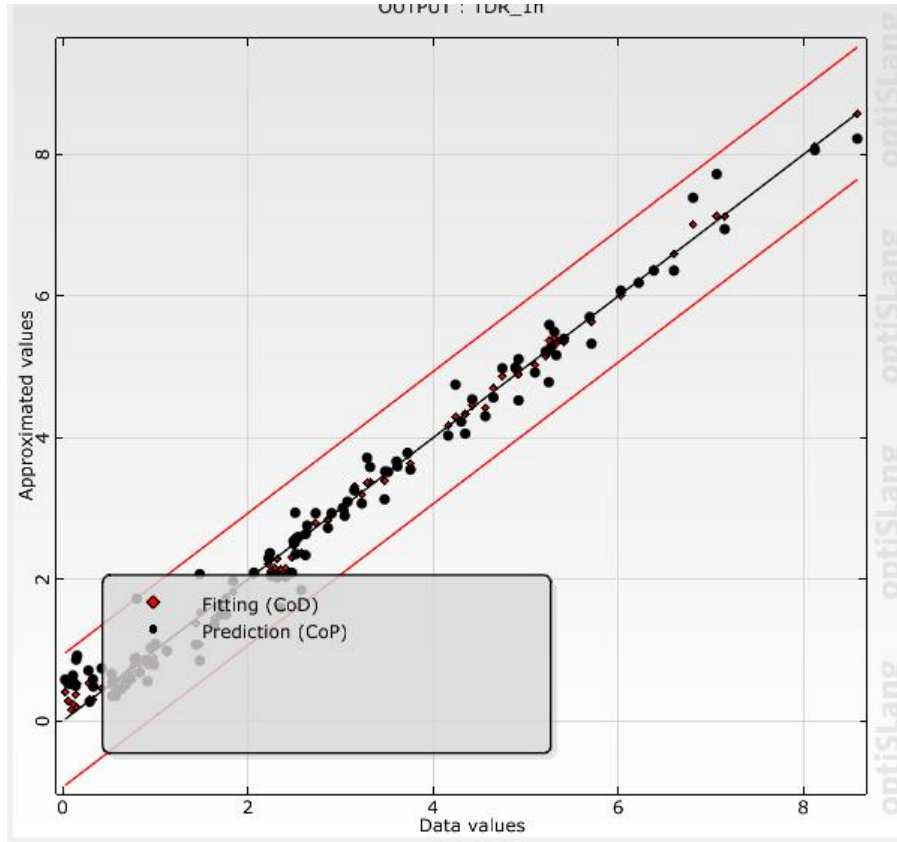
→ trace 도체의 두께는
임피던스에 별 영향이 없다는 의미

Sub_t 와 TDR_1n 결과는
-0.41의 연관지표를 보여줌

→ 유전체 두께가 임피던스에 큰
영향을 줄 수 있음

- 각 설계변수가 임피던스에 어느정도 영향을 주는지 확인함으로써,
어떤 변수가 성능에 핵심적 영향을 주는지 평가 가능
- 절대값이 1에 가까울수록 연관성이 높음
(+ 부호는 변수 값이 커질수록 성능이 좋아짐
- 부호는 변수 값이 작아질수록 성능이 좋아짐)

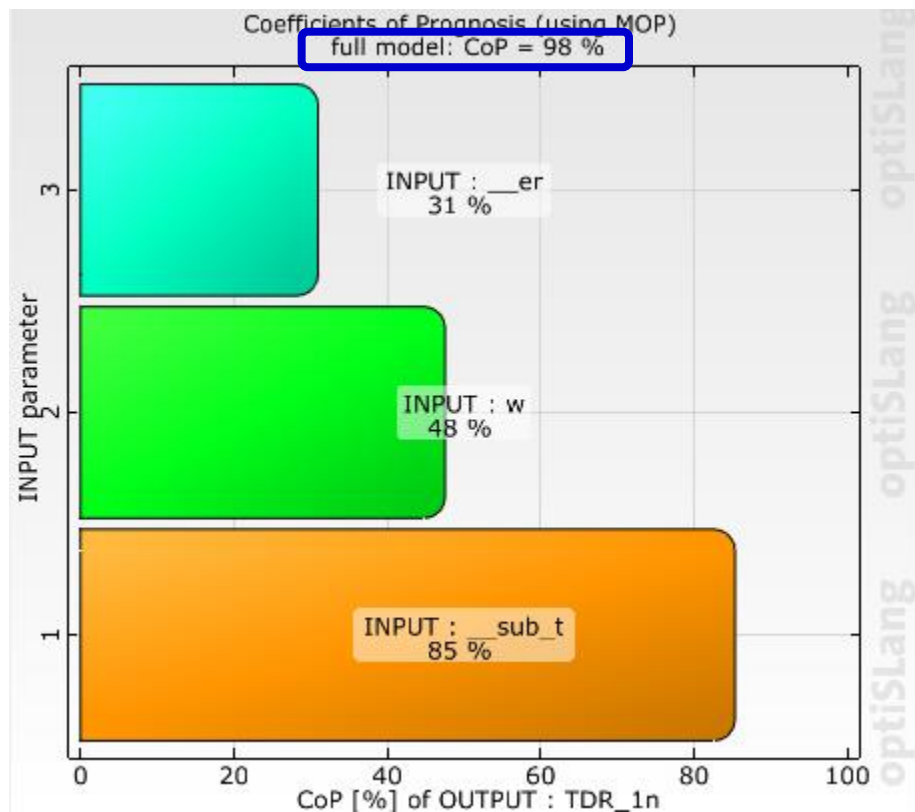
Sensitivity 분석결과의 예 — Residual Plot



- Linear한 형태로 설계변수 조합 sample 들이 point 형태로 모여 있음
- 즉 목표와 크게 동떨어진 설계변수 조합이 잘 걸러졌다는 의미

- Metamodel이 입력변수를 조절함에 따라 목표 성능이 얼마나 가깝게 가는지에 대한 전체적인 분석값
- 실험점들이 선형(Linear)에 가까울수록 (즉 대각선에 가까울수록) 최적의 metamodel이 형성됨

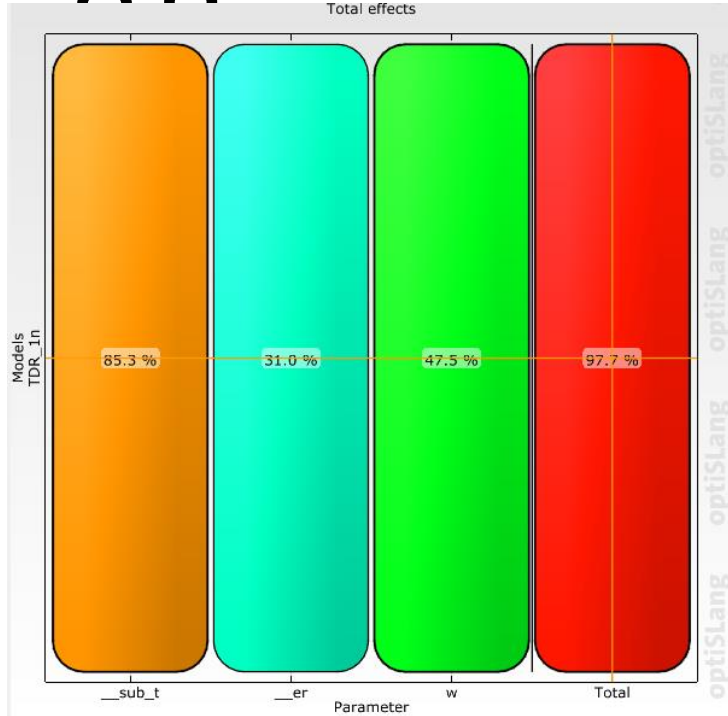
Meta Model의 품질 평가 – CoP



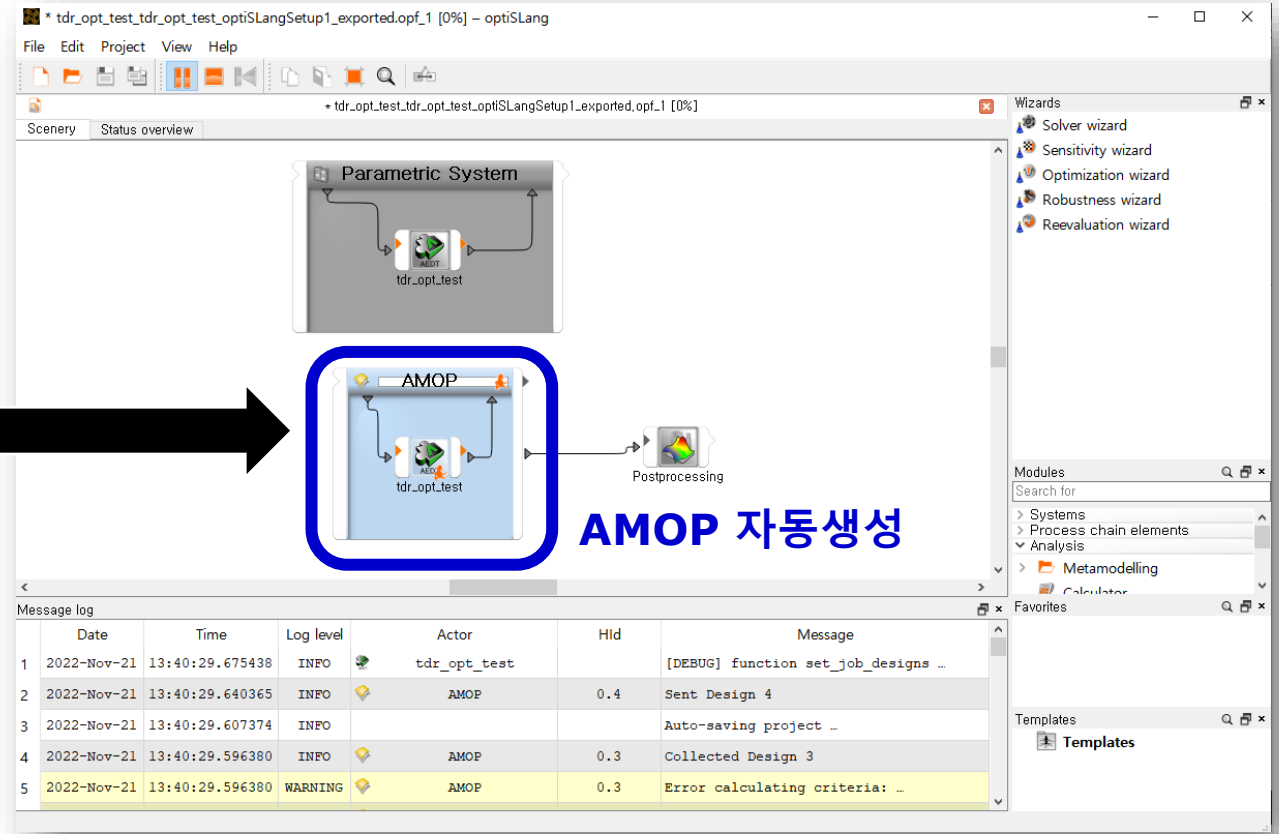
- Sensitivity 분석을 통해 만들어진 Metamodel 의 정확도는 CoP(**Coefficient of Prognosis**)라는 지표로 평가
- CoP가 1 (또는 100%)에 가까울수록, 최적화를 수행한 Metamodel이 실제 모델에 가깝게 잘 구현되었다는 뜻
- TDR 최적화 사례의 경우, 모든 변수 조합(full model)의 CoP가 98%로서, 매우 정확한 metamodel이 구축됨
- 주요 변수별 CoP를 보면, 유전율(er)이 31%, 선풍(w)가 48%, 기판두께(sub_t)가 85%로서, 기판두께 – 선풍 – 유전율의 순서로 임피던스 결과에 큰 영향을 주고 있다는 의미.
- 이를 통해 어떤 설계변수와 목표성능에 가장 영향을 주는지 알 수 있으며, 도체두께(sig_t)와 같이 영향도가 적은 설계변수는 알아서 제외됨.
- 이런 식으로, 정확한 최적화를 위해서는 CoP가 높은 Metamodel이 생성될 수 있도록 User가 입력변수와 범위를 잘 설정하여 Sensitivity 분석을 수행하는 것이 중요.

AMOP (Adaptive Metamodel of Optimal Prognosis)

생성

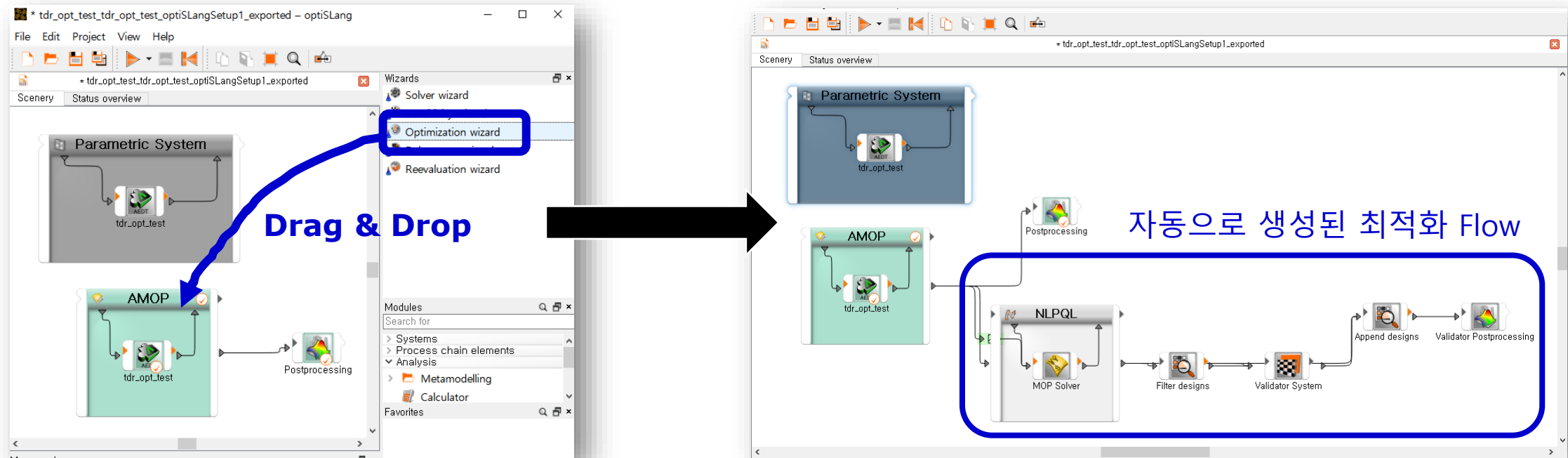


- 각 설계변수별로 임피던스에 어느정도 영향을 주는지를 종합 평가
- 이 예제의 경우 유전체 두께(sub_t), 선평(w), 유전율(er)의 순서대로 임피던스 결과에 영향을 줄을 알 수 있음



- Sensitivity 분석을 수행하면 최적화된 Metamodel, 즉 **AMOP**가 자동으로 Optislang 내부에 구축됨

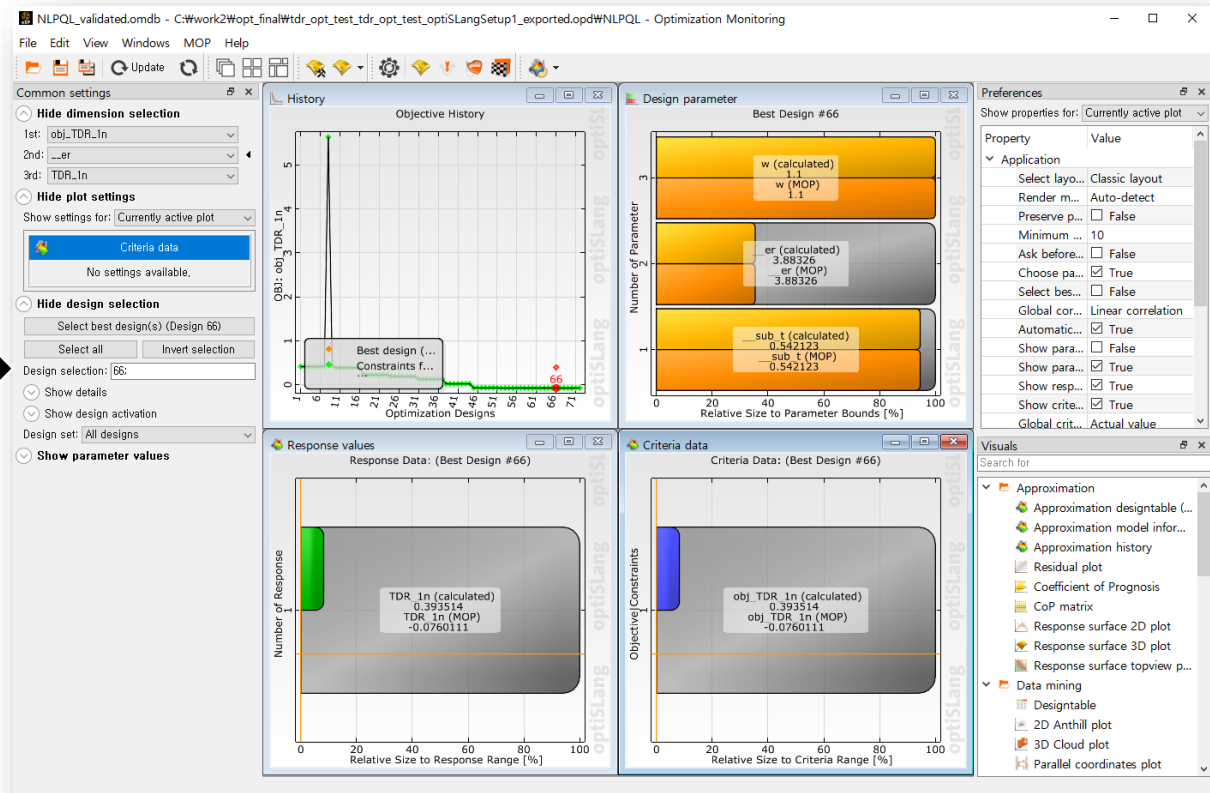
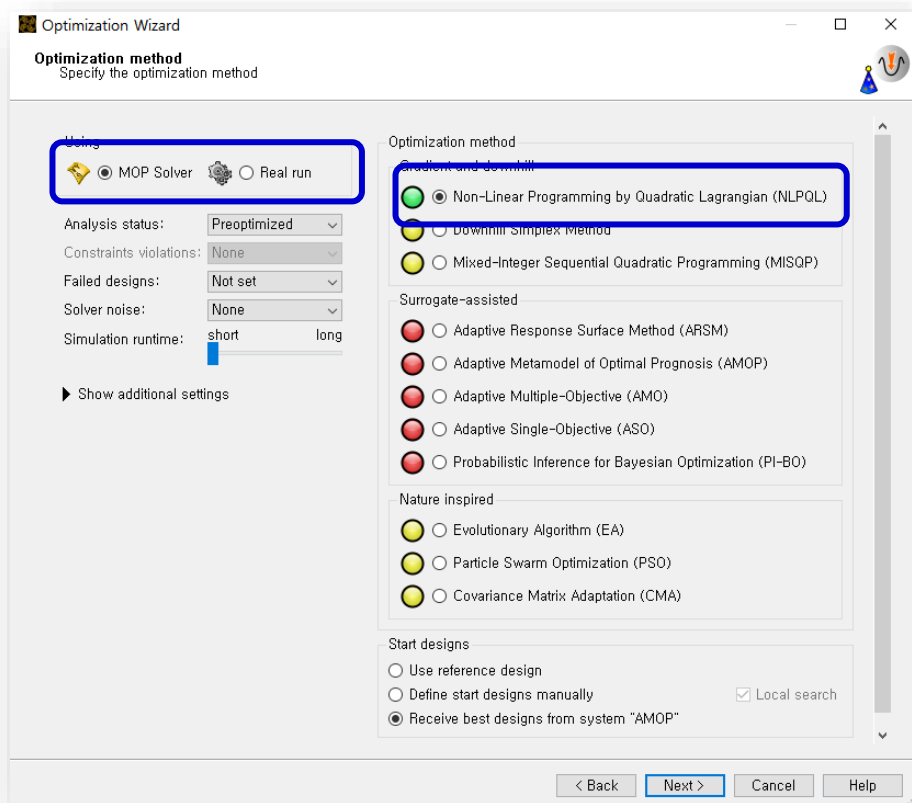
AMOP 기반의 최적화 설정



- Optislang 창에서 Optimization Wizard를 Drag & Drop하여 AMOP 블록에 내려놓으면 자동으로 최적화 Flow가 구축됨 (사용자가 manual로 작업할 필요 전혀 없음)

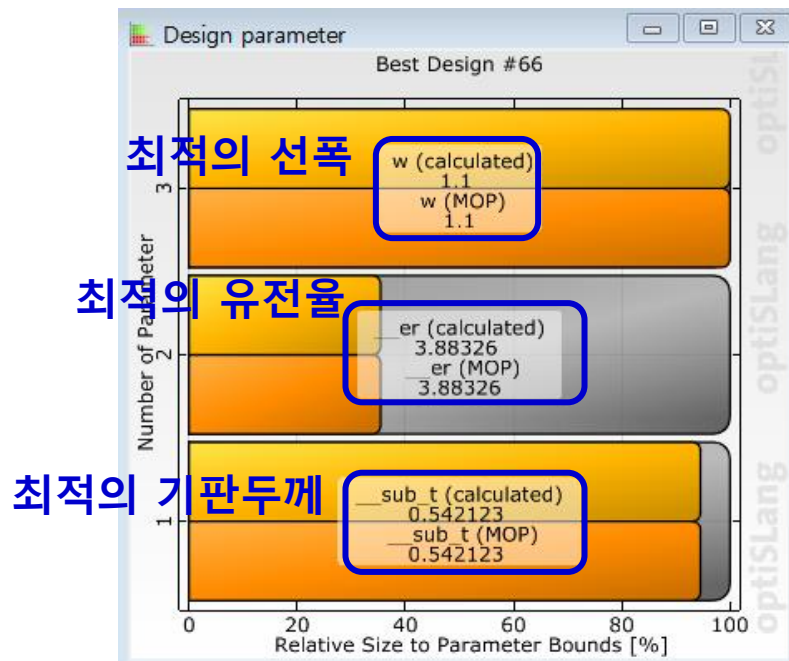
최적화 실행

최적화 결과



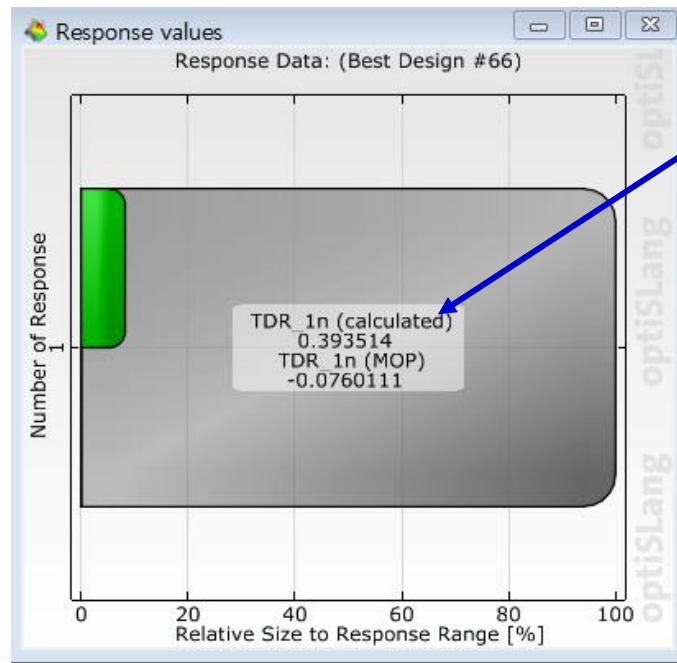
- 최적화 설정에서 AMOP 기반의 MOP Solver를 선택하거나, 실제 AEDT 해석을 수행하는 Real Run을 선택 가능
- Metamodel (AMOP)기반의 MOP Solver로 수행하면 통상 **수초 이내**에 최적화 수행

최적화 결과



- 최적화의 결과로, 선폭은 1.1mm, 유전율은 3.88, 기판두께는 0.54mm일 때 가장 50옴에 근접함을 알 수 있음
- Sensitivity 결과에 기반하여, 영향도가 적은 도체 두께는 자동으로 배제하고 유효한 변수범위만 자동 선정되어 최적화가 진행됨.

최적화된 TDR 임피던스 비교결과



간편한 목표설정을 위해, 계산된 임피던스에서 50옴을 뺀 값을 최소화하도록 최적화 수행한 결과

즉 AEDT에서 실제로 해석(Calculated)된 임피던스는 $50 + 0.393514 = 50.4$ 옴

MOP 기반으로 예측된 최종 임피던스는 $50 - 0.076 = 49.94$ 옴

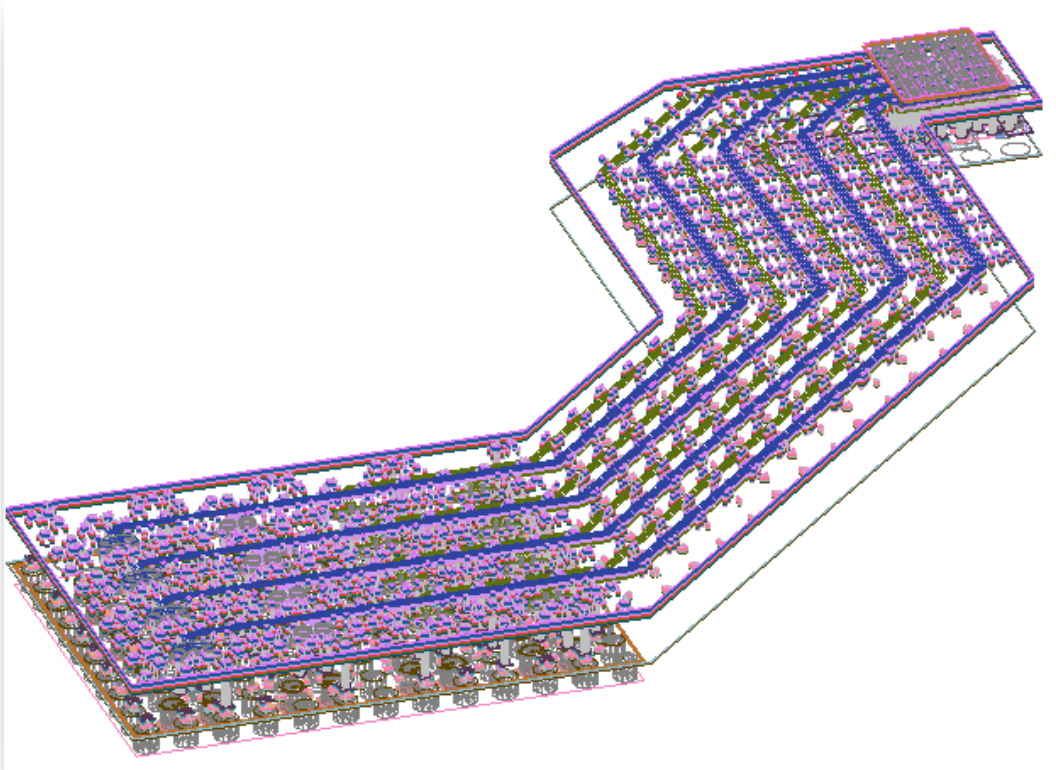
- 마지막으로 최적화된 변수값에 따른 AEDT TDR 해석이 자동으로 수행되고, 그 결과값이 AMOP 기반으로 계산된 TDR 결과값과 자동 비교됨
- 위 결과의 경우, MOP Solver가 예측한 최종 임피던스는 49.94옴, 실제 AEDT에서 최적화된 변수로 계산된 임피던스는 50.39옴 정도로 관측됨 (거의 비슷..)

/ Optislang in AEDT

- Optislang을 이용하면, 기존 AEDT에서 할 수 없었던 Metamodel / Deep learning 과 같은 고성능의 최적화 기능을 수행할 수 있습니다.
- SIwave, HFSS 해석을 수행하는 AEDT 에는 Optislang 설정이 메뉴로 내장되어 있습니다.
- AEDT와 Optislang은 최적화를 위한 설계변수의 입력(input)과 최적변수의 결과검증(output)이 모두 자동화되어 있어서, In/Out coding이 전혀 필요하지 않습니다.
- Optislang에서 제공하는 Metamodel 기반의 최적화를 활용하면, 엔지니어가 유효한 설계변수와 범위를 찾는 노력을 대폭 줄일 수 있습니다.
- 유효한 Metamodel을 구축하기 위해 설계변수조합별로 해석하는 시간이 많이 소요될 수 있지만, 모든 과정이 자동으로 진행되기 때문에 실질적으로 엔지니어는 적은 노력으로 시간과 Hardware만 투자하여 효율적인 최적화 모델을 구축할 수 있습니다.
- Optislang은 Ansys 뿐 아니라 거의 모든 Simulation tool과도 collaboration이 가능한 Common Platform 입니다.

Meta Model for SERDES PCB Optimization

Example : SerDes 3D Model Optimization using Meta Model



Parameterized SerDes PCB Design

Step 1

다양한 설계 변수 조합에 따른
Meta Model 생성



Step 2

S 파라미터 결과에 기반한
PCB Design 최적화



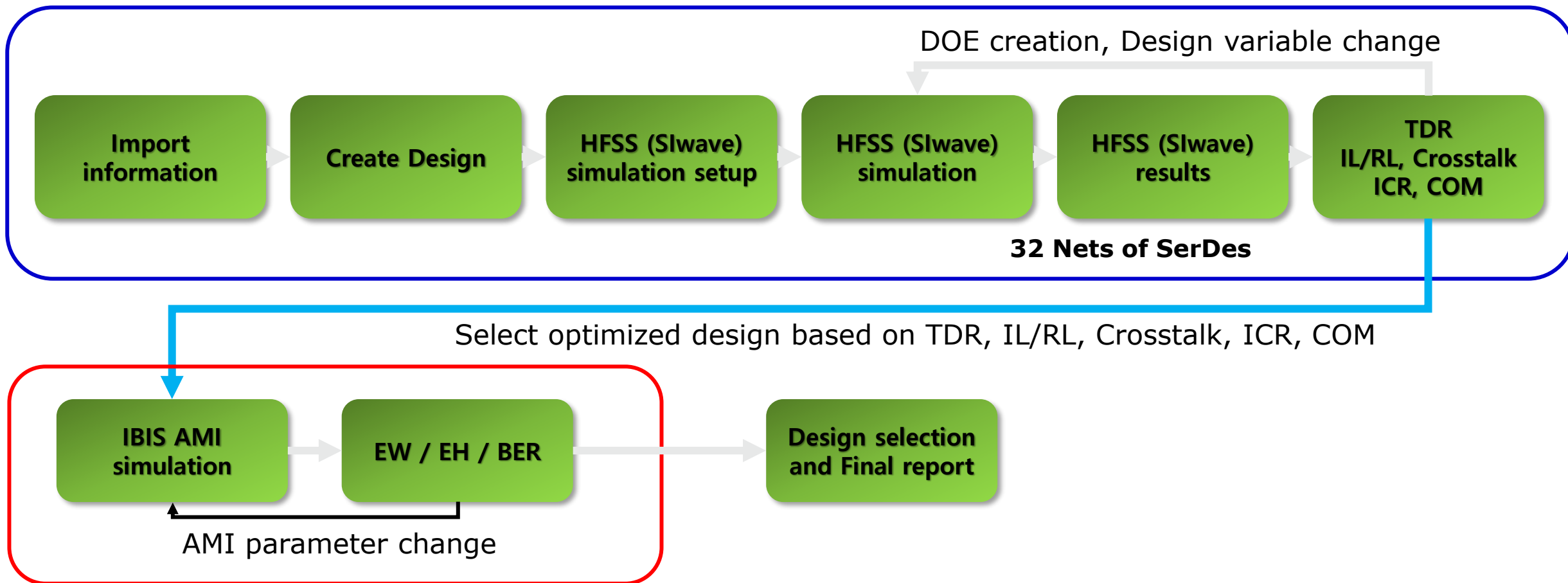
Step 3

EYE 분석을 통한
AMI Parameter 최적화

SerDes Optimization Workflow

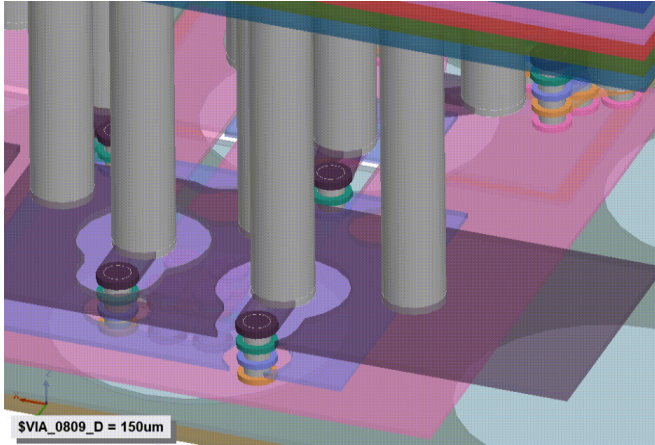
ANSYS 2023

Part1 : 3D model optimization

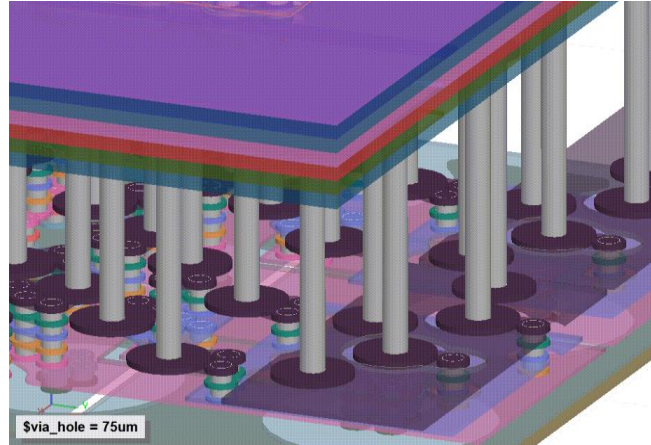


Part2 : Circuit optimization

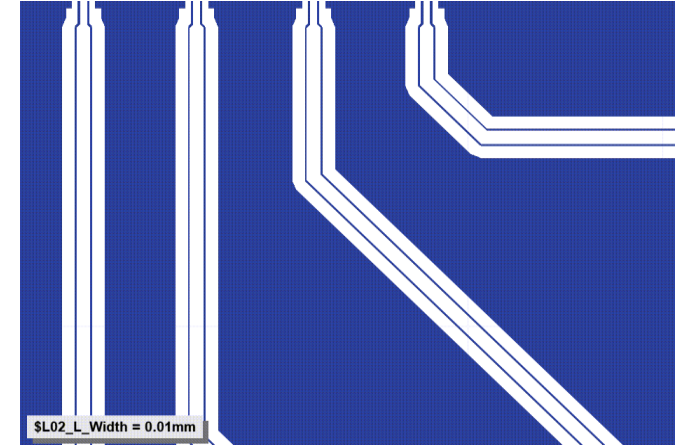
Parametrized SerDes 3D Model – PCB 설계변수의 예



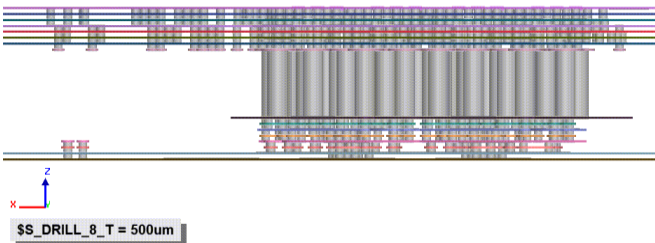
Via pad Size



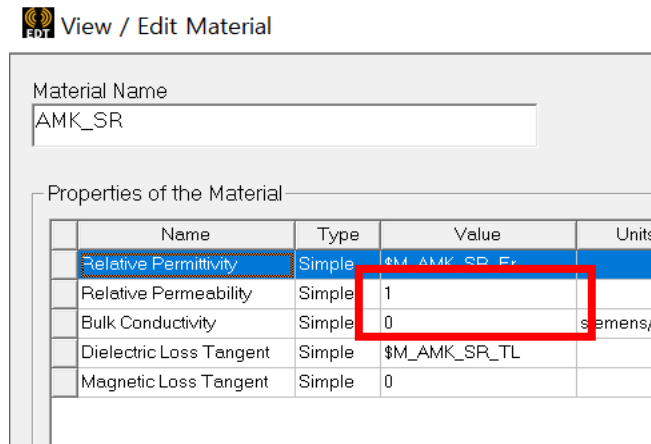
Drill size



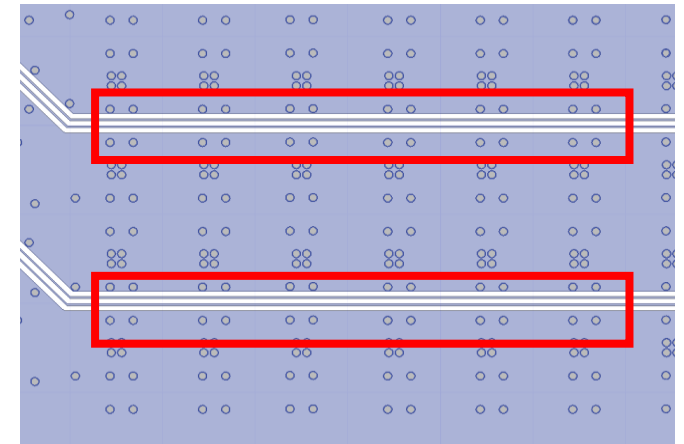
Trace width



Stack-up thickness



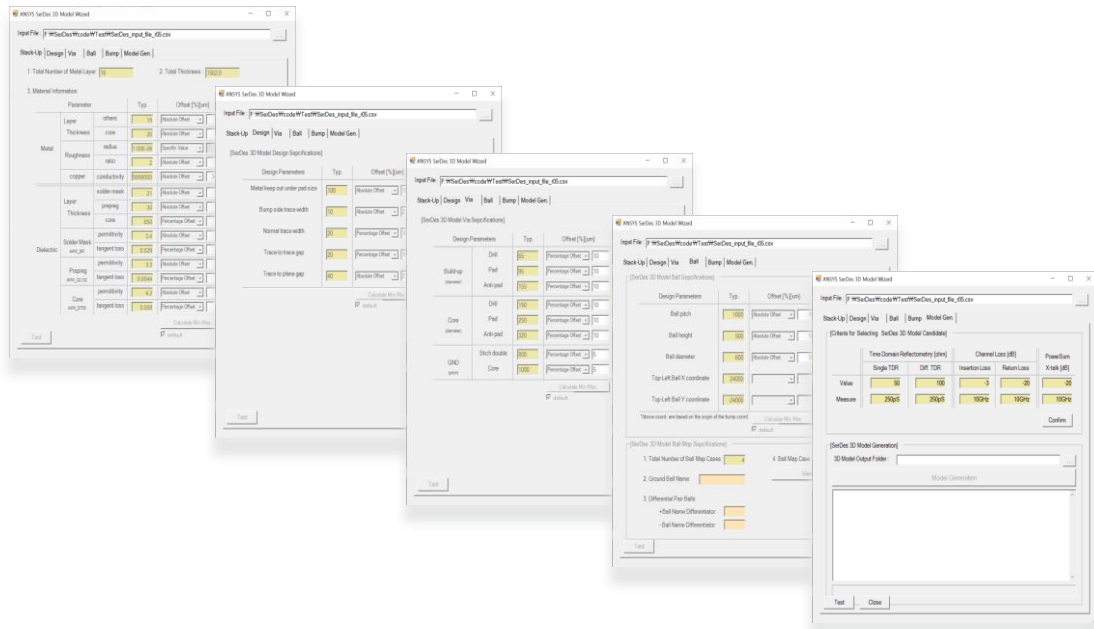
Material properties



Ground stitch via distance

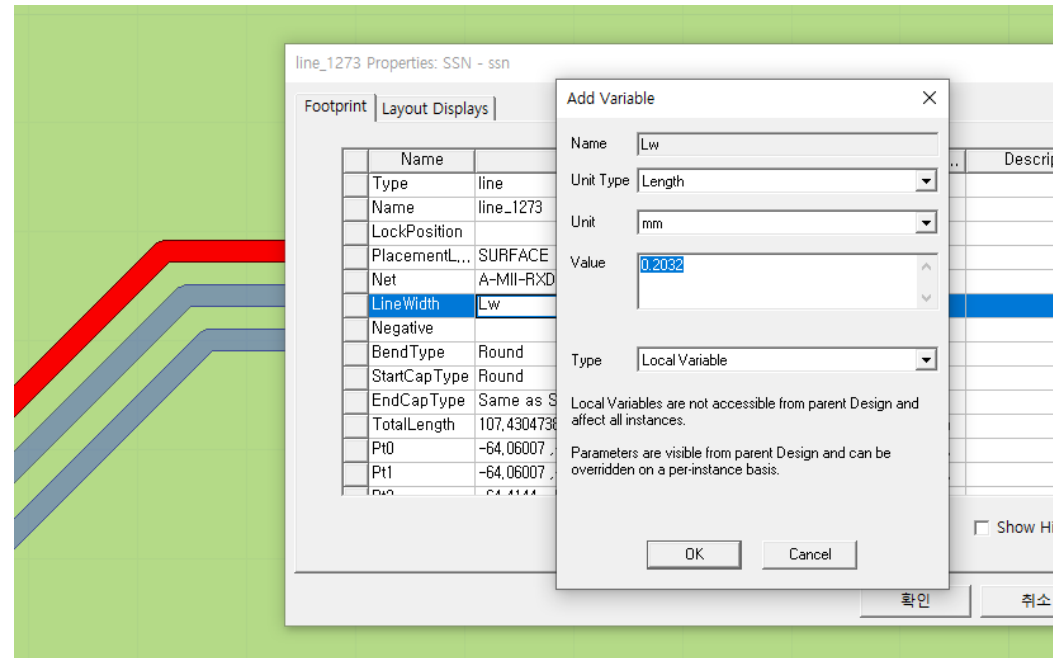
PCB 설계변수 설정 방법

1. Automated PCB Generation



- Script를 이용한 SerDes PCB 생성 자동화
- 완벽한 parameterizing이 가능하지만 공수가 많이 필요

2. Variable Setup



- 선폭, via pad 크기, 적층 두께, 유전율 등을 수작업으로 변수화
- 빠르고 간단하지만 trace routing 변경, 적층 조합 변경 등 고수준의 변수화는 불가능

Overview: 3D-model optimization

- **Design variables**

1. Trace width
2. Trace to Trace gap
3. Ground gap
4. Anti-pad clearance
5. Metal keep-out for pad
6. Ball map change

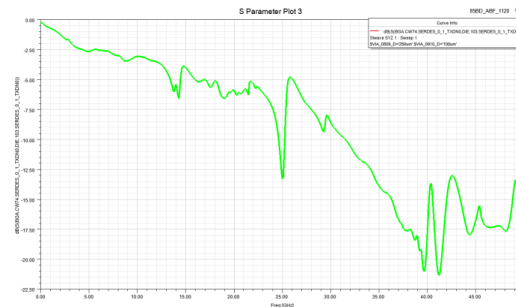
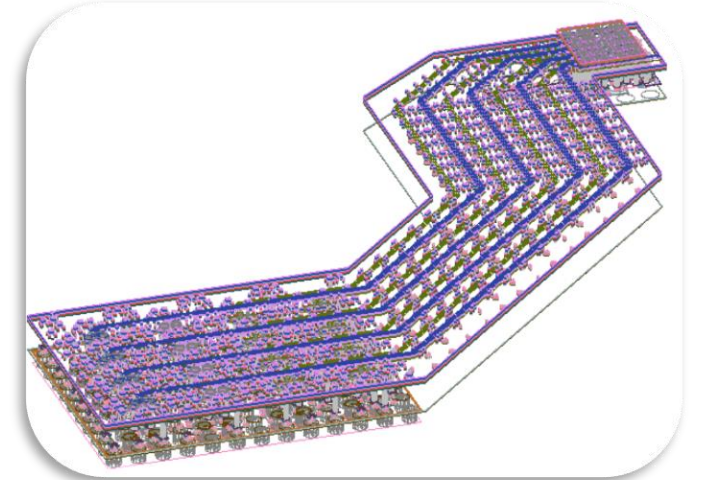
- **Optimization goal:**

1. TDR Impedance(5) = 50 Ohm @252ps $\rightarrow$ Euclid norm(50-TDR)
2. S11 (5) < -10dB @10GHz
3. S21 (5) > -10dB @25GHz

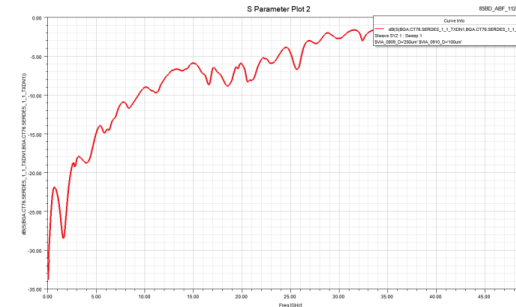
- **Initial design:**

- $\sum TDR(e)$ 16.5436
- Max.S11 -9.02388
- Max.S21 -13.2874

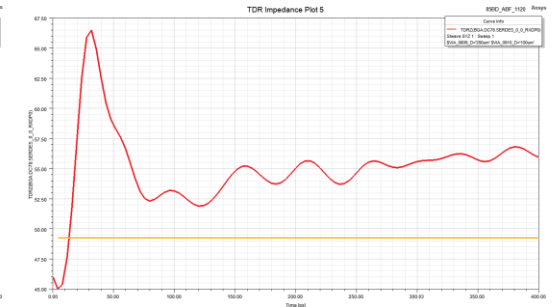
Fully parameterized 3D model



Insertion loss (S21)



Return loss (S11)



TDR

Initial design performance

constraint										objective				
S11					S21					TDR				
S11_CN74	S11_CN80	S11_CT76	S11_CW74	S11_DB78	S21_CN74	S21_CN80	S21_CT76	S21_CW74	S21_DB78	TDR_CN74	TDR_CN80	TDR_CT76	TDR_CW74	TDR_DB78
-13.7652	-13.4327	-9.02388	-15.863	-13.6569	-7.42397	-4.72534	-7.94566	-13.2874	-7.32358	46.45167	46.63721	47.26802	46.57181	46.52766

#S21

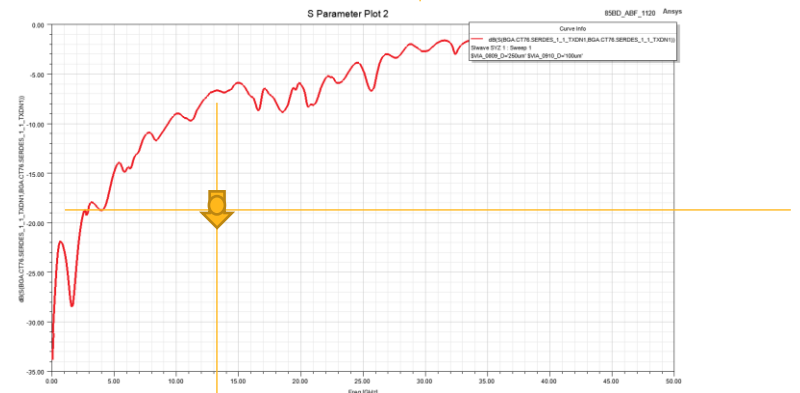
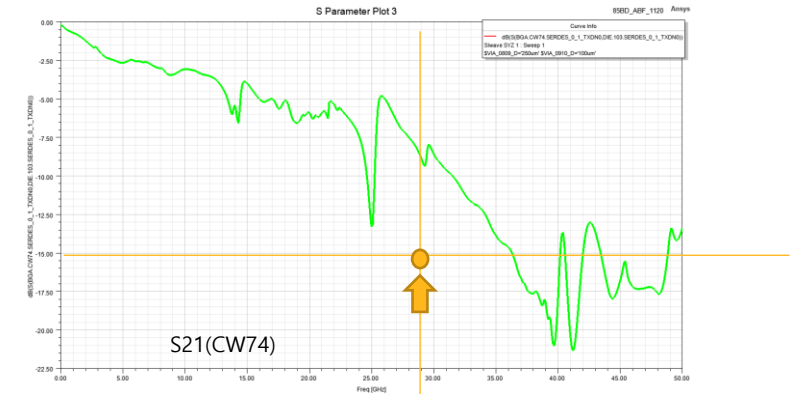
dB(S(DIE.1352.SERDES_1_1_RXDN0,BGA.CN74.SERDES_1_1_RXDN0))	>	-10dB	25GHz
dB(S(DIE.1287.SERDES_1_0_RXDN1,BGA.CN80.SERDES_1_0_RXDN1))	>	-10dB	25GHz
dB(S(DIE.1329.SERDES_1_1_TXDN1,BGA.CT76.SERDES_1_1_TXDN1))	>	-10dB	25GHz
dB(S(DIE.103.SERDES_0_1_TXDN0,BGA.CW74.SERDES_0_1_TXDN0))	>	-10dB	25GHz
dB(S(DIE.60.SERDES_0_0_RXDN0,BGA.DB78.SERDES_0_0_RXDN0))	>	-10dB	25GHz

#S11

dB(S(BGA.CN74.SERDES_1_1_RXDN0,BGA.CN74.SERDES_1_1_RXDN0))	<	-10dB	10GHz
dB(S(BGA.CN80.SERDES_1_0_RXDN1,BGA.CN80.SERDES_1_0_RXDN1))	<	-10dB	10GHz
dB(S(BGA.CT76.SERDES_1_1_TXDN1,BGA.CT76.SERDES_1_1_TXDN1))	<	-10dB	10GHz
dB(S(BGA.CW74.SERDES_0_1_TXDN0,BGA.CW74.SERDES_0_1_TXDN0))	<	-10dB	10GHz
dB(S(BGA.DB78.SERDES_0_0_RXDN0,BGA.DB78.SERDES_0_0_RXDN0))	<	-10dB	10GHz

#TDR

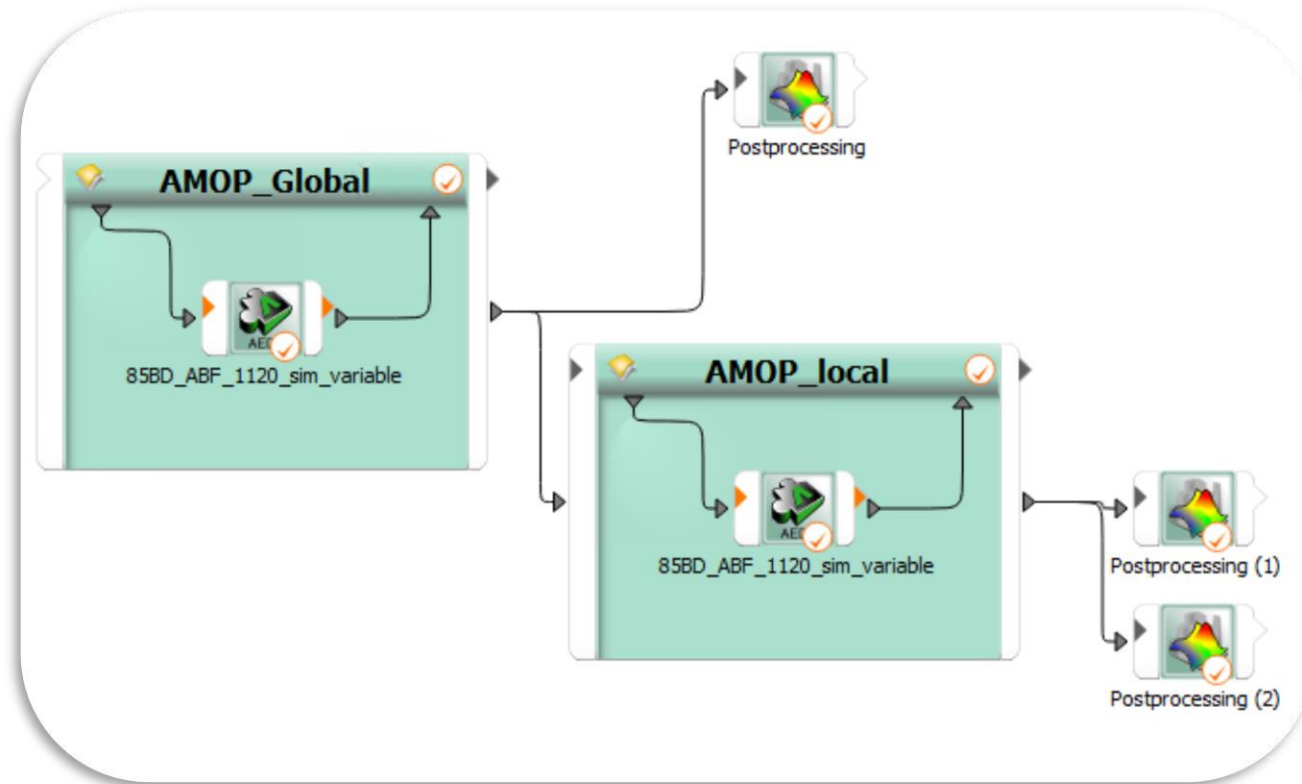
TDRZ(BGA.CN74.SERDES_1_1_RXDN0)	=	50Ohm	250ps
TDRZ(BGA.CN80.SERDES_1_0_RXDN1)	=	50Ohm	250ps
TDRZ(BGA.CT76.SERDES_1_1_TXDN1)	=	50Ohm	250ps
TDRZ(BGA.CW74.SERDES_0_1_TXDN0)	=	50Ohm	250ps
TDRZ(BGA.DB78.SERDES_0_0_RXDN0)	=	50Ohm	250ps



S11(CT76)

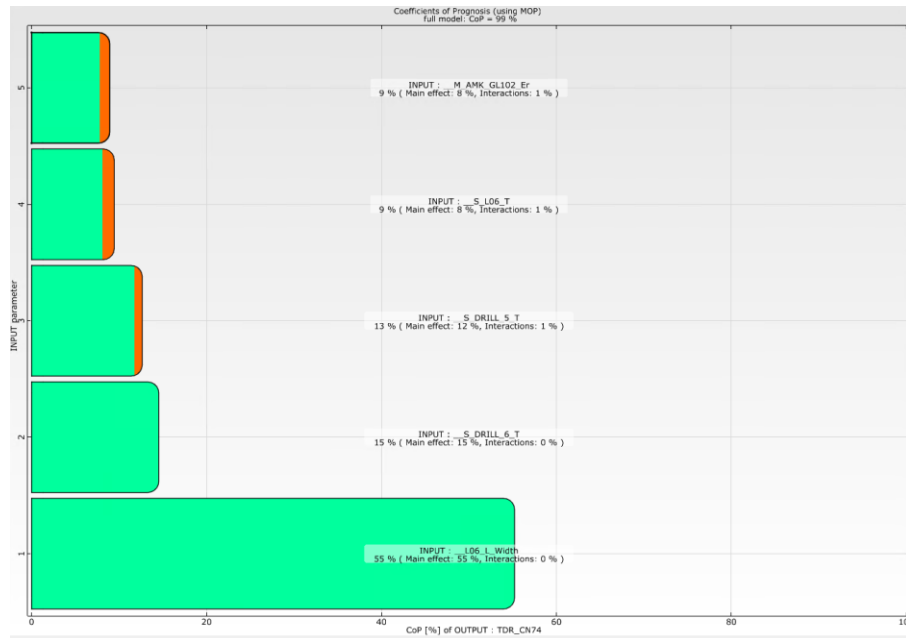
Sensitivity Analysis for Meta Model

- Metal Model 구축에 여러가지 다양한 알고리즘을 사용할 수 있음.
- 본 사례에는 **advanced** and **space-filling Latin Hypercube Sampling** 적용



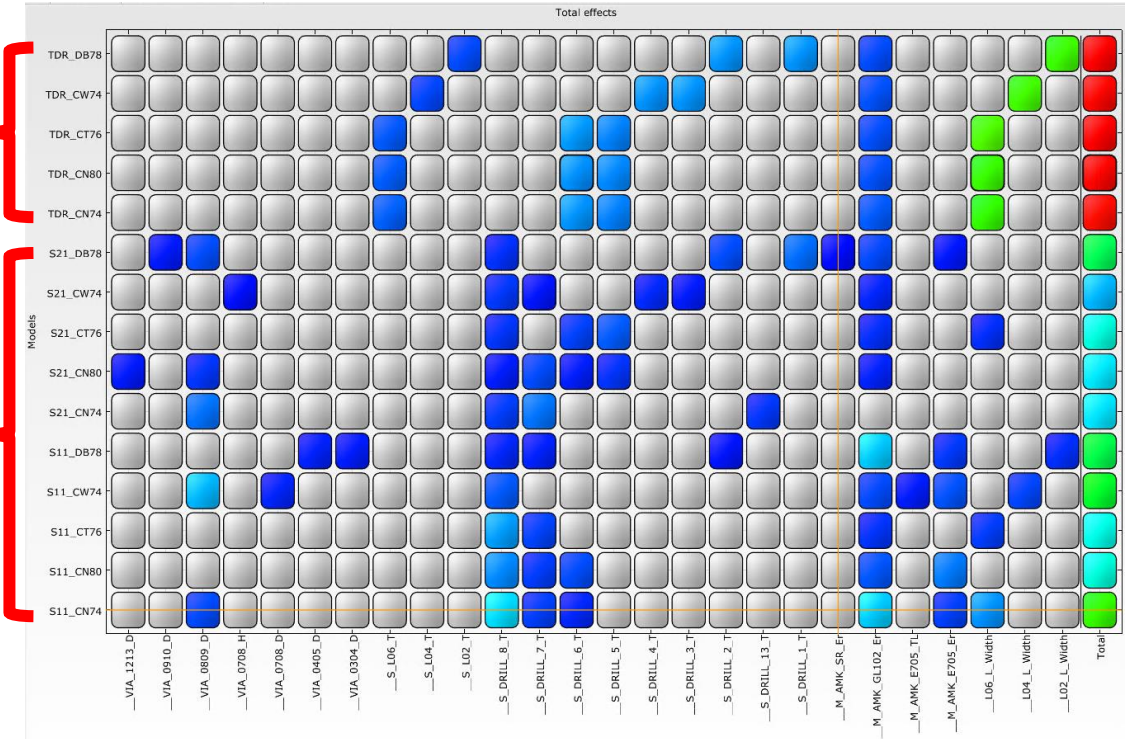
Sensitivity analysis results: Coefficient of Prognosis (COP)

- COP를 통해 완성된 SerDes Meta Model을 평가
- S_DRILL_6_T (유전체 두께), S_DRILL_5_T(유전체 두께) , L06_L_Width(선폭) 의c의 3개 설계변수가 TDR 결과에 가장 크게 영향을 주고 있음

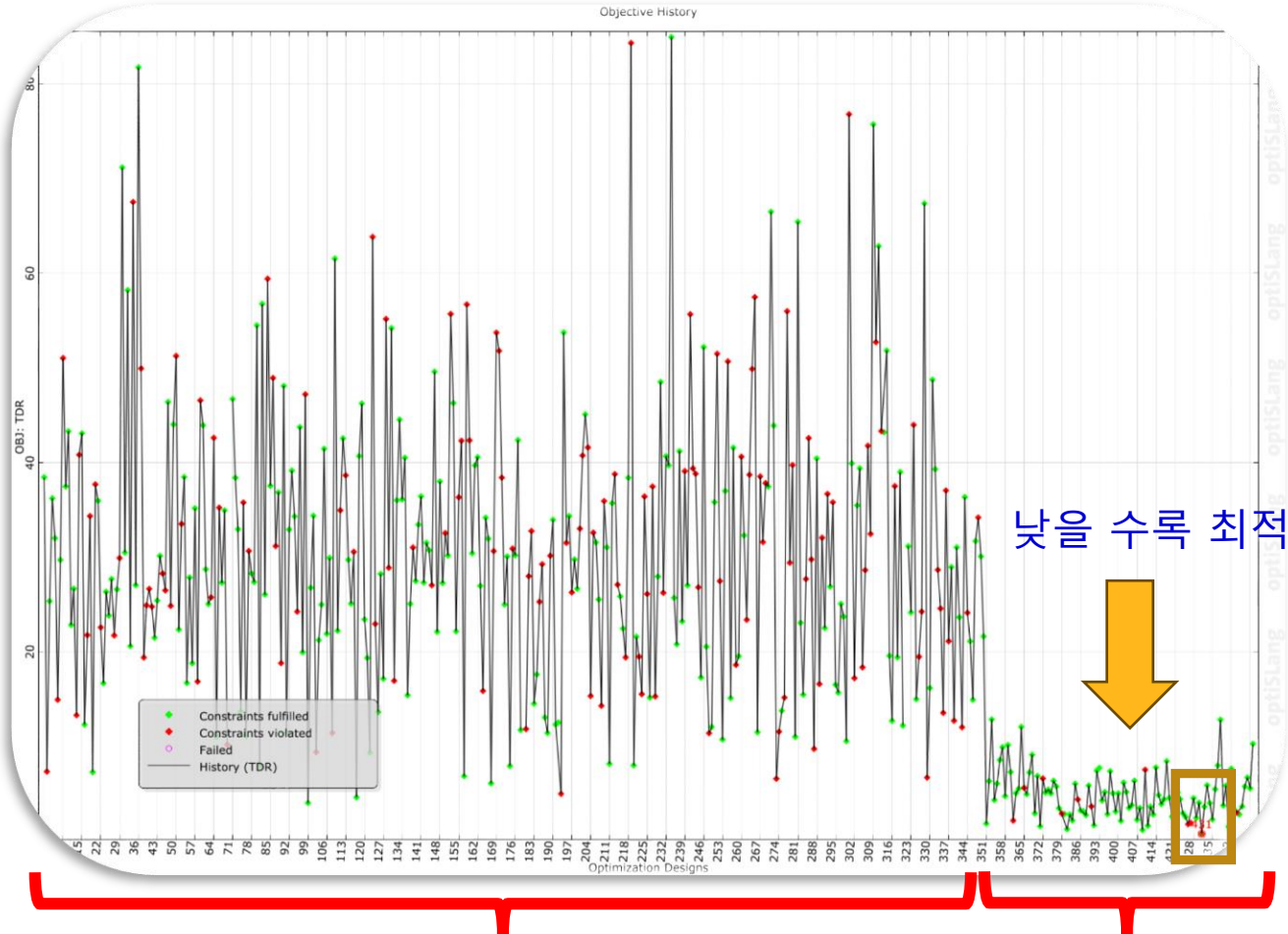


objective

constraint



Optimization history



Global refinement (Sensitivity Analysis)

Local refinement (optimization)

- 본 사례에서는 최적의 디자인을 찾기 위해 자동적으로 총 431 의 변수조합 iteration이 수행됨
- 이 과정은 Sensitivity 분석을 통해 다양한 변수 조합을 흔들어보는 Global Refinement 과정과, 이를 토대로 특정 성능 결과를 위한 최적 설계변수값을 찾는 최적화 (Local refinement)과정으로 분류됨

Best design candidate

Responses (Performance)

Responses	S11_CN74	S11_CN80	S11_CT76	S11_CW74	S11_DB78	S21_CN74	S21_CN80	S21_CT76	S21_CW74	S21_DB78	TDR_CN74	TDR_CN80	TDR_CT76	TDR_CW74	TDR_DB78
Initial design	-13.7652	-13.4327	-9.02388	-15.863	-13.6569	-7.42397	-4.72534	-7.94566	-13.2874	-7.32358	55.2131	55.2627	55.4818	54.6697	54.721
Best design	-14.5801	-14.8748	-10.9605	-18.957	-14.5918	-5.99246	-4.72646	-7.77611	-8.58138	-7.02873	49.64426	49.88954	50.06748	50.04095	49.84067

Best design variables

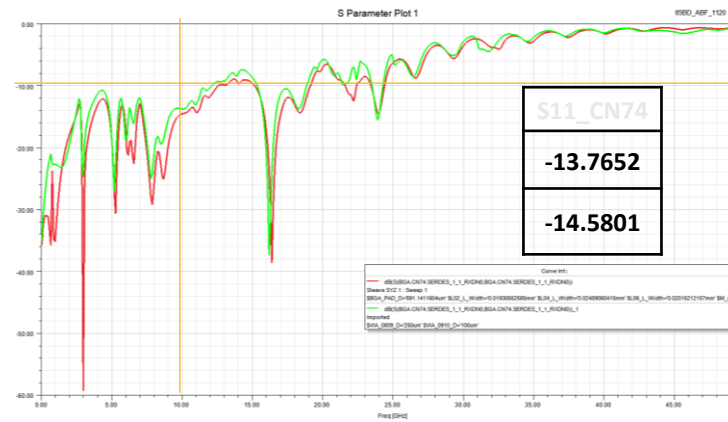
- $TDR = 50ohm$
- $S_{11}(5) < -10dB$ @10GHz
- $S_{21}(5) > -10dB$ @25GHz

__VIA_1516_H	__VIA_1516_D	__VIA_1415_H	__VIA_1415_D	__VIA_1314_H	__VIA_1314_D	__VIA_1213_H	__VIA_1213_D	__VIA_1112_H	__VIA_1112_D	__VIA_1011_H	__VIA_1011_D	__VIA_0910_H	__VIA_0910_D	__VIA_0809_H	__VIA_0809_D	__VIA_0708_H	__VIA_0708_D
68.71475	94.85738	61.14734	90.4744	62.17826	91.79132	69.5574	83.05177	66.10662	82.90407	67.61937	82.5812	63.86336	90.3874	155.6377	250.7131	71.70406	95.15054
__VIA_0607_H	__VIA_0607_D	__VIA_0506_H	__VIA_0506_D	__VIA_0405_H	__VIA_0405_D	__VIA_0304_H	__VIA_0304_D	__VIA_0203_H	__VIA_0203_D	__VIA_0102_H	__VIA_0102_D	__S_SRT_T	__S_SRB_T	__S_L16_T	__S_L15_T	__S_L14_T	__S_L13_T
76.02189	87.28645	65.08614	95.80163	67.61814	92.80269	64.87662	87.67984	71.55098	91.53261	75.44042	97.58107	20.1862	21.82394	16.84941	11.5978	16.70777	13.23562
__S_L12_T	__S_L11_T	__S_L10_T	__S_L09_T	__S_L08_T	__S_L07_T	__S_L06_T	__S_L05_T	__S_L04_T	__S_L03_T	__S_L02_T	__S_L01_T	__S_DRILL_9_T	__S_DRILL_8_T	__S_DRILL_7_T	__S_DRILL_6_T	__S_DRILL_5_T	__S_DRILL_4_T
12.15214	13.37361	18.54991	20.49506	20.89622	12.03484	16.16126	16.51926	13.7426	16.98024	12.96289	14.23069	23.50858	814.01	26.52775	23.80845	26.85462	31.55211
__S_DRILL_3_T	__S_DRILL_2_T	__S_DRILL_15_T	__S_DRILL_14_T	__S_DRILL_13_T	__S_DRILL_12_T	__S_DRILL_11_T	__S_DRILL_10_T	__S_DRILL_1_T	__M_AMK_SR_TL	__M_AMK_SR_Er	__M_AMK_GL102_TL	__M_AMK_GL102_Er	__M_AMK_E705_TL	__M_AMK_E705_Er	__L06_L_Width	__L04_L_Width	__L02_L_Width
25.50927	20.98262	23.57688	30.60801	36.63555	31.77226	22.67096	29.86836	23.79859	0.032997	3.562214	0.004673	3.289138	0.008677	4.096811	0.020152	0.024891	0.019306
__BGA_PAD_D																	
591.1412																	

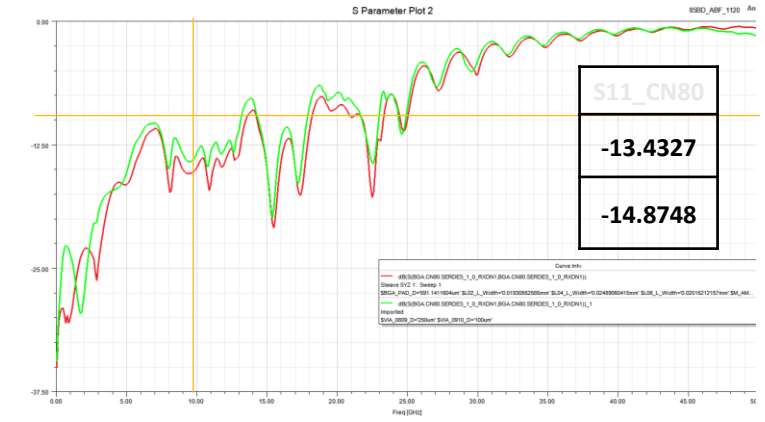
Best design performance, Return Loss

$S_{11} < -10\text{dB}$ @10GHz

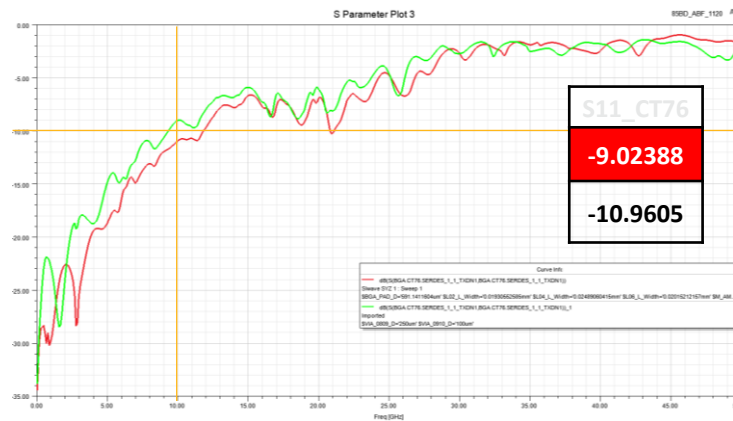
 Initial Design
 Best Design



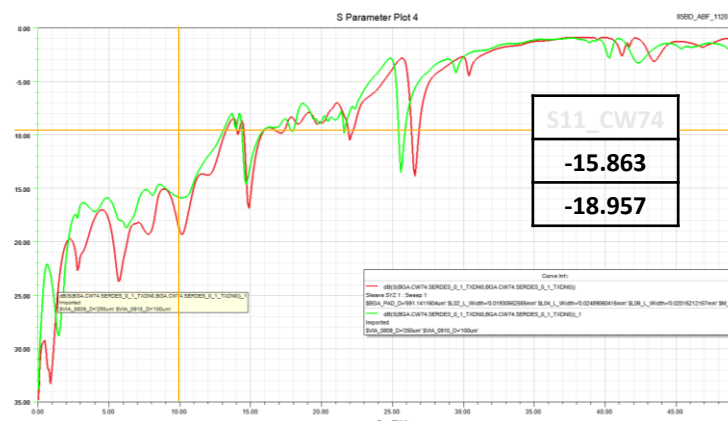
dB(S(BGA.CN74.SERDES_1_1_RXDN0,BGA.CN74.SERDES_1_1_RXDN0))



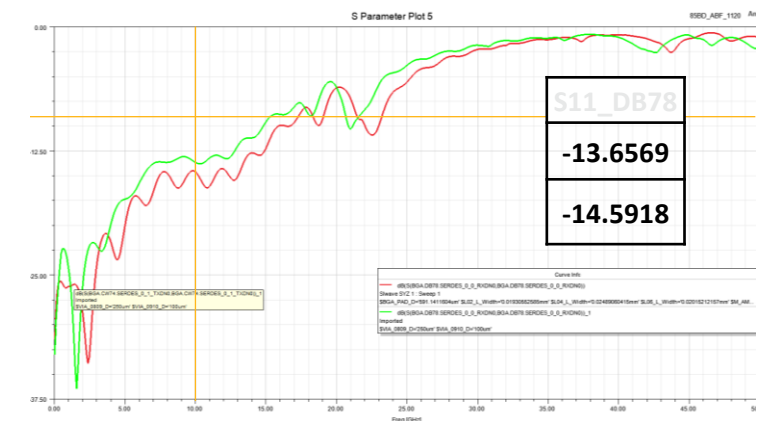
dB(S(BGA.CN80.SERDES_1_0_RXDN1,BGA.CN80.SERDES_1_0_RXDN1))



dB(S(BGA.CT76.SERDES_1_1_TXDN1,BGA.CT76.SERDES_1_1_TXDN1))



dB(S(BGA.CW74.SERDES_0_1_TXDN0,BGA.CW74.SERDES_0_1_TXDN0))

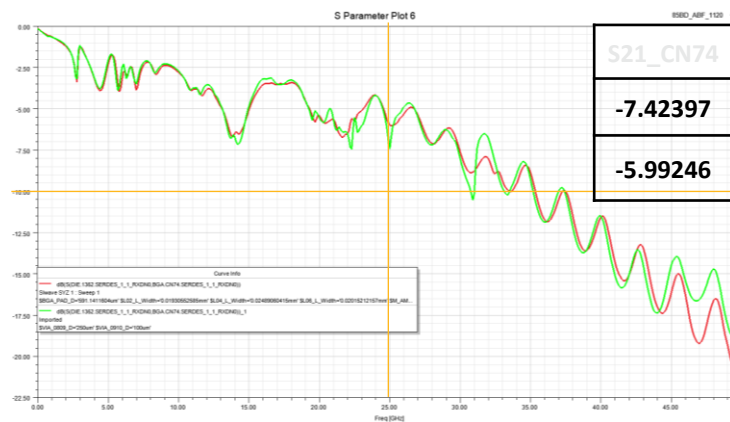


dB(S(BGA.DB78.SERDES_0_0_RXDN0,BGA.DB78.SERDES_0_0_RXDN0))

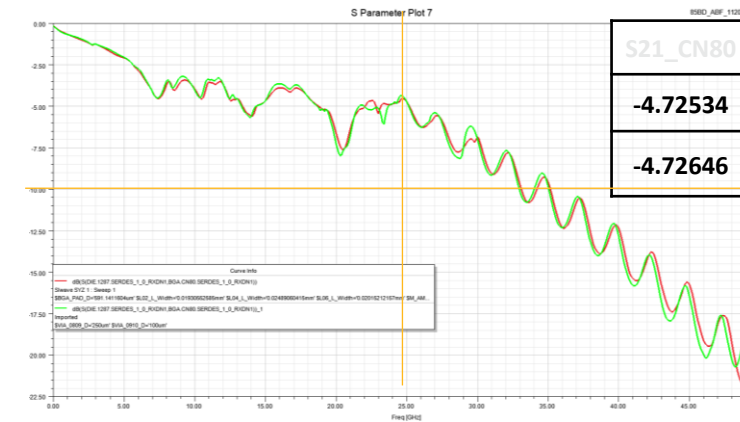
Best design performance, Insertion Loss

$S_{21} > -10\text{dB}$ @25GHz

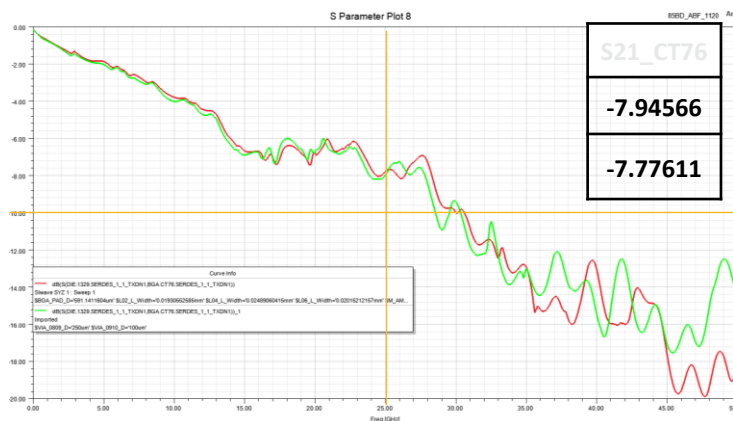
 Initial Design
 Best Design



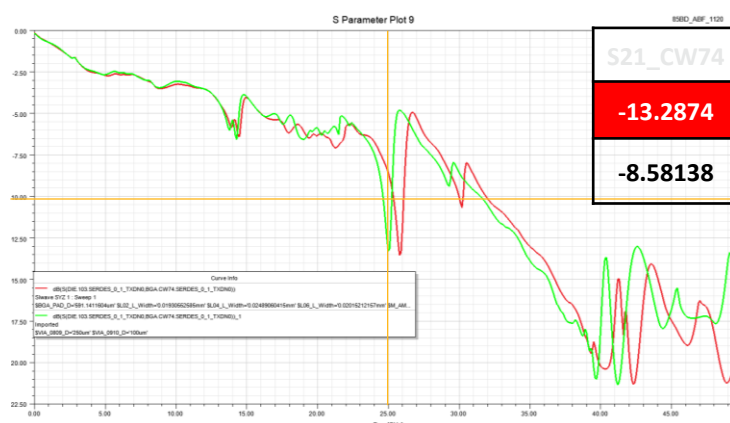
dB(S(DIE.1352.SERDES_1_1_RXDN0,BGA.CN74.SERDES_1_1_RXDN0))



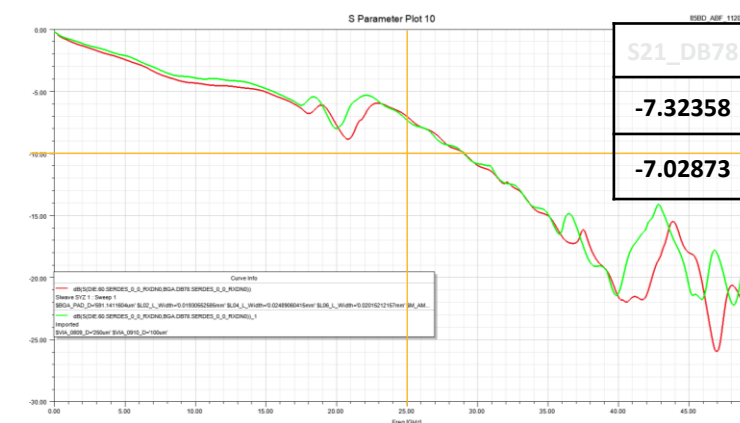
dB(S(DIE.1287.SERDES_1_0_RXDN1,BGA.CN80.SERDES_1_0_RXDN1))



dB(S(DIE.1329.SERDES_1_1_TXDN1,BGA.CT76.SERDES_1_1_TXDN1))



dB(S(DIE.103.SERDES_0_1_TXDN0,BGA.CW74.SERDES_0_1_TXDN0))

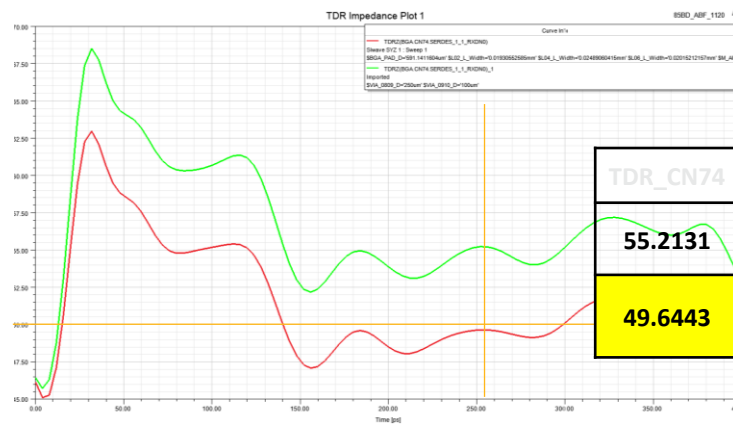


dB(S(DIE.60.SERDES_0_0_RXDN0,BGA.DB78.SERDES_0_0_RXDN0))

Best design performance, TDR

**TDR Impedance
= 50 Ohm @252ps**

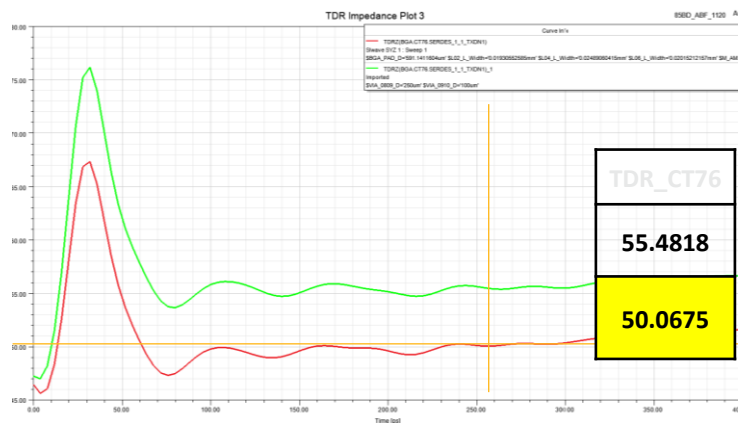
Initial Design
Best Design



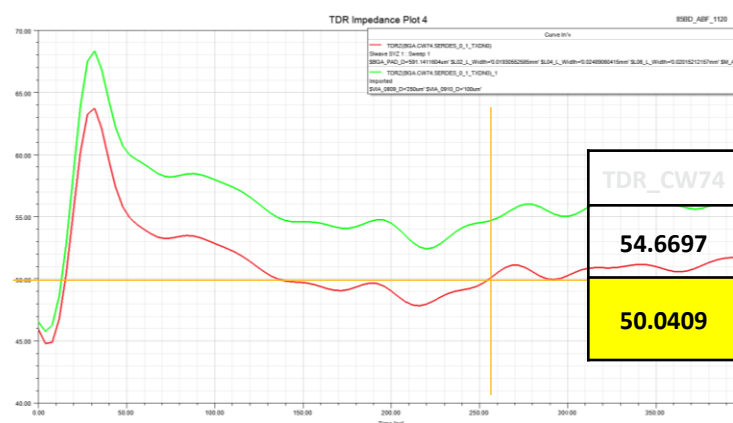
TDRZ(BGA.CN74.SERDES_1_1_RXDN0)



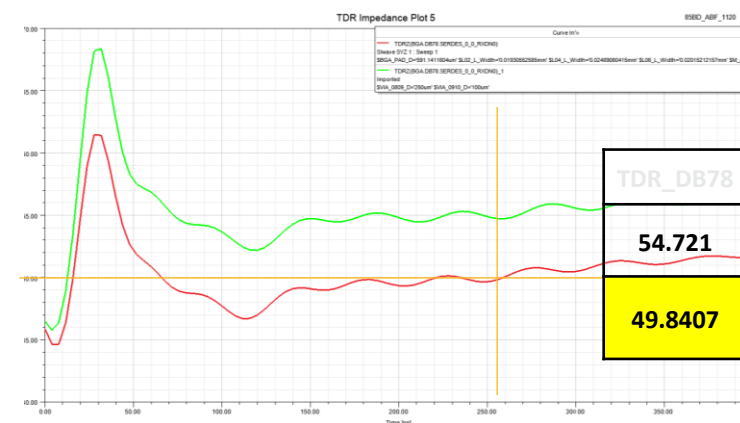
TDRZ(BGA.CN80.SERDES_1_0_RXDN1)



TDRZ(BGA.CT76.SERDES_1_1_TXDN1)

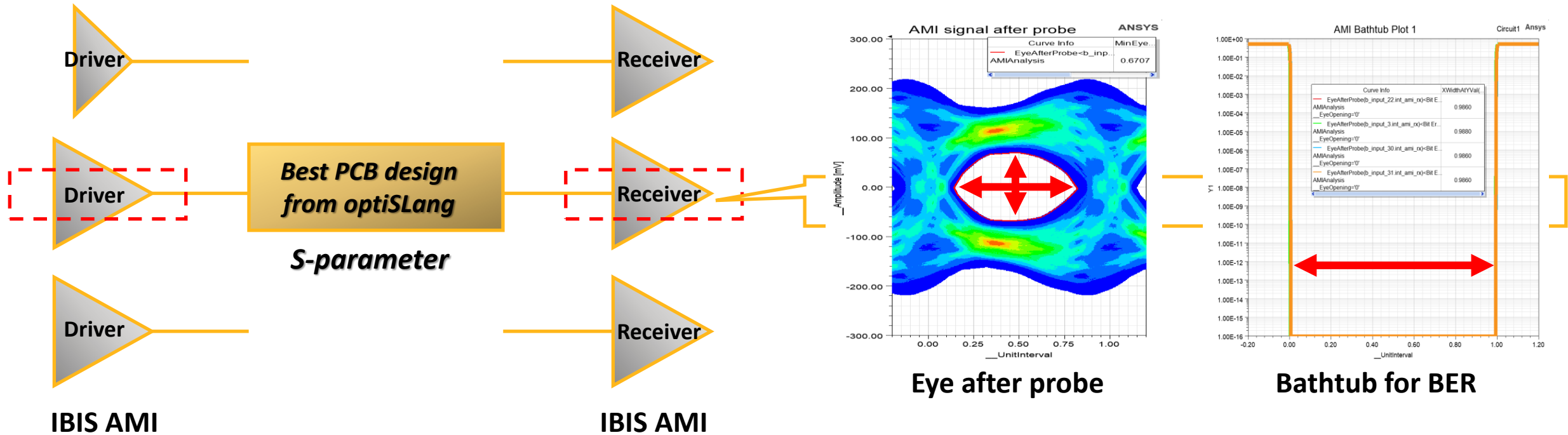


TDRZ(BGA.CW74.SERDES_0_1_TXDN0)



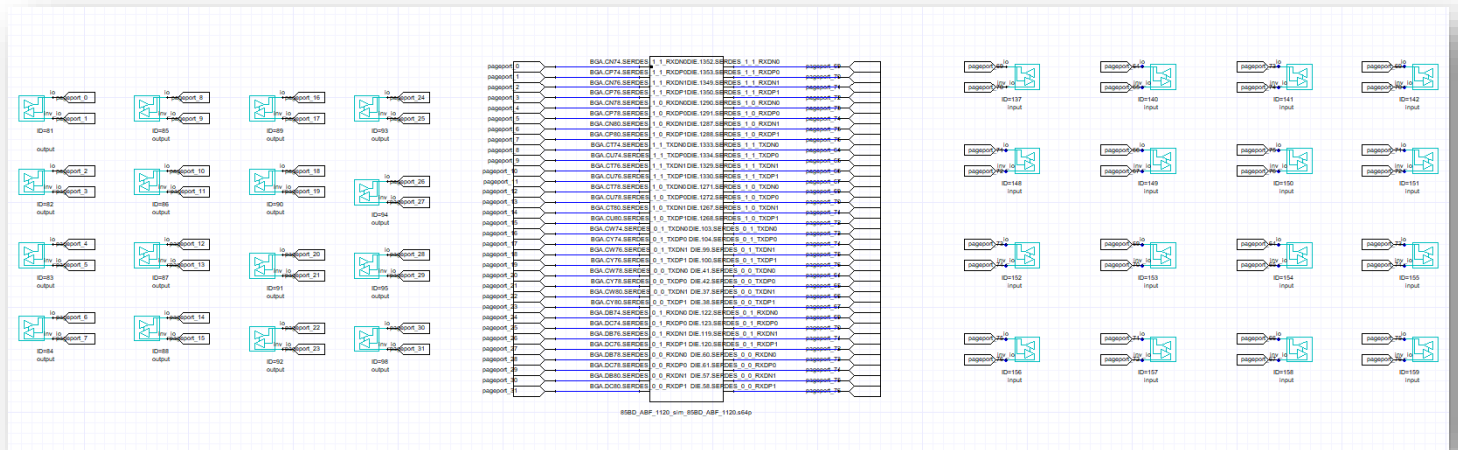
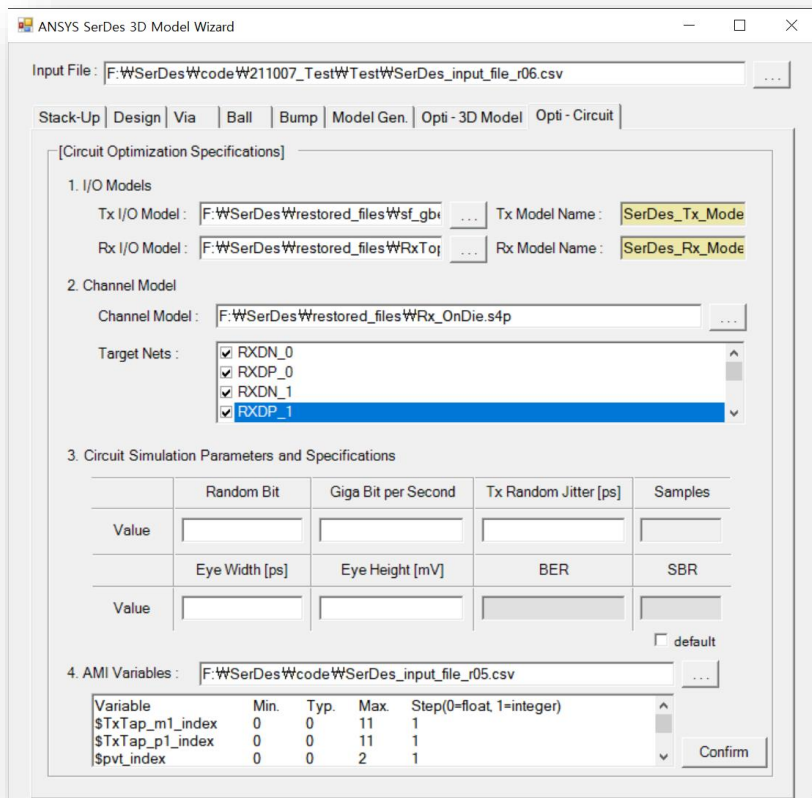
TDRZ(BGA.DB78.SERDES_0_0_RXDN0)

2단계 SerDes 최적화 : AMI Parameter



- TDR과 S 파라미터가 최적화된 PCB를 이용하여, EYE를 최대화하기 위해 AMI parameter 를 추가로 최적화

Automation for AEDT Circuit Schematic for Optimization



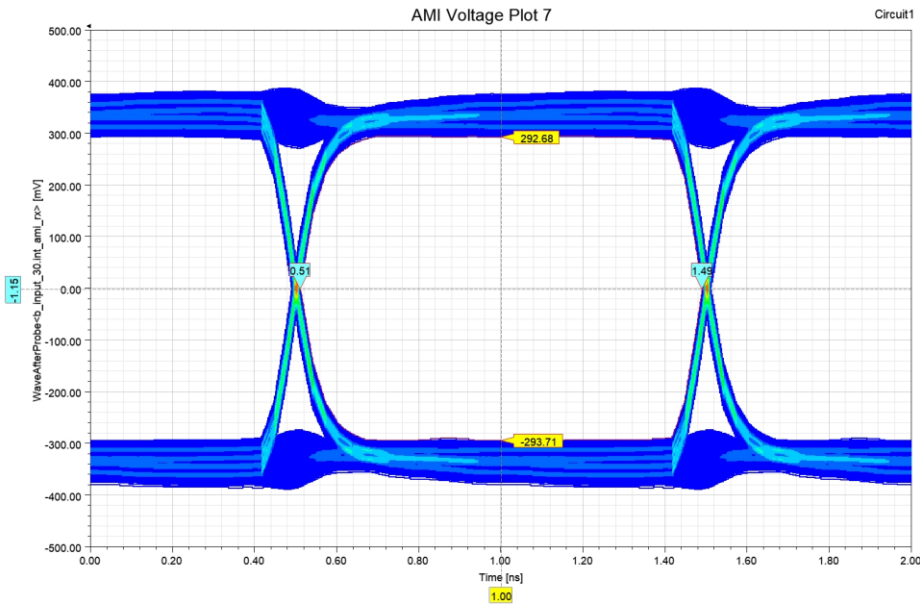
Design of Experiments Setup

Design of Experiments				
Design of Experiments	Table	Response Surface	Calculations	Goals
Variables				
Options				
Name	Solution	Calculation	Calculation Range	
MinEyeWidth_WaveAfterProbe(b_input_22.int_amiLrx)	AMIAAnalysis	MinEyeWidth(WaveAfterProbe<b_input_22.int_amiLrx>, 1ns, 0ns, 0ns, 1, 0mV, 1, 1ns)	Time(All)	
MinEyeWidth_WaveAfterProbe(b_input_3.int_amiLrx)	AMIAAnalysis	MinEyeWidth(WaveAfterProbe<b_input_3.int_amiLrx>, 1ns, 0ns, 0ns, 1, 0mV, 1, 1ns)	Time(All)	
MinEyeWidth_WaveAfterProbe(b_input_30.int_amiLrx)	AMIAAnalysis	MinEyeWidth(WaveAfterProbe<b_input_30.int_amiLrx>, 1ns, 0ns, 0ns, 1, 0mV, 1, 1ns)	Time(All)	
MinEyeWidth_WaveAfterProbe(b_input_31.int_amiLrx)	AMIAAnalysis	MinEyeWidth(WaveAfterProbe<b_input_31.int_amiLrx>, 1ns, 0ns, 0ns, 1, 0mV, 1, 1ns)	Time(All)	
MinEyeHeight_WaveAfterProbe(b_input_22.int_amiLrx)	AMIAAnalysis	MinEyeHeight(WaveAfterProbe<b_input_22.int_amiLrx>, 1ns, 0ns, 0ns, 1, 0mV, 1, 1ns)	Time(All)	
MinEyeHeight_WaveAfterProbe(b_input_3.int_amiLrx)	AMIAAnalysis	MinEyeHeight(WaveAfterProbe<b_input_3.int_amiLrx>, 1ns, 0ns, 0ns, 1, 0mV, 1, 1ns)	Time(All)	
MinEyeHeight_WaveAfterProbe(b_input_30.int_amiLrx)	AMIAAnalysis	MinEyeHeight(WaveAfterProbe<b_input_30.int_amiLrx>, 1ns, 0ns, 0ns, 1, 0mV, 1, 1ns)	Time(All)	
MinEyeHeight_WaveAfterProbe(b_input_31.int_amiLrx)	AMIAAnalysis	MinEyeHeight(WaveAfterProbe<b_input_31.int_amiLrx>, 1ns, 0ns, 0ns, 1, 0mV, 1, 1ns)	Time(All)	
XWidthAtYVal_EyeAfterProbe(b_input_22.int_amiLrx){Bit Error Rate}	AMIAAnalysis	XWidthAtYVal(EyeAfterProbe(b_input_22.int_amiLrx)<Bit Error Rate>, 1e-12)	__UnitInterval(All)___EyeOpening(0)	
XWidthAtYVal_EyeAfterProbe(b_input_30.int_amiLrx){Bit Error Rate}	AMIAAnalysis	XWidthAtYVal(EyeAfterProbe(b_input_30.int_amiLrx)<Bit Error Rate>, 1e-12)	__UnitInterval(All)___EyeOpening(0)	
XWidthAtYVal_EyeAfterProbe(b_input_31.int_amiLrx){Bit Error Rate}	AMIAAnalysis	XWidthAtYVal(EyeAfterProbe(b_input_31.int_amiLrx)<Bit Error Rate>, 1e-12)	__UnitInterval(All)___EyeOpening(0)	
XWidthAtYVal_EyeAfterProbe(b_input_3.int_amiLrx){Bit Error Rate}	AMIAAnalysis	XWidthAtYVal(EyeAfterProbe(b_input_3.int_amiLrx)<Bit Error Rate>, 1e-12)	__UnitInterval(All)___EyeOpening(0)	

Target Performance : MinEye Width, MinEye Height, BER

Optimized EYE

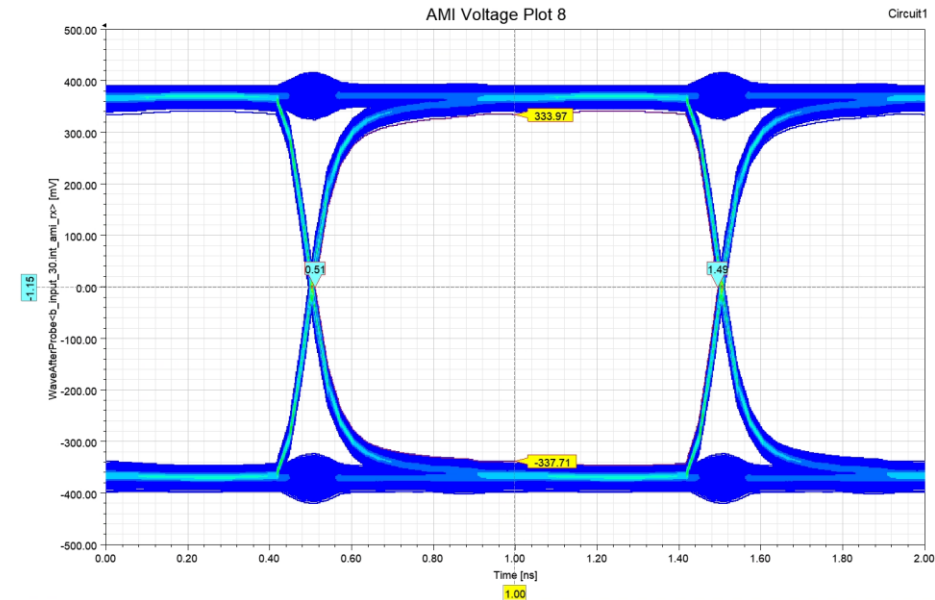
Min Eye Hight 586.2mV
Min Eye Width 0.98



Default AMI parameter combinations



Min Eye Hight 671.6mV
Min Eye Width 0.984



Best AMI parameter combinations

/ Optislang - Meta Model 기반의 Optimizatin의 의미

- 실제 SerDes 설계에는 수많은 입력 변수 조합이 있을 수 있음
 - 모든 조합을 simulation 하는 것은 매우 비효율 적임.
 - 기존의 generic과 같은 일반적인 최적화 알고리즘으로는 변수가 많아졌을 때 효율이 극도로 떨어지며, 최적화 결과는 Engineer의 변수 범위 및 목표 설정에 크게 의존.
- **Deep Learning / Meta Model 기반의 최적화는...**
 - 수많은 변수조합 결과에서 실질적으로 의미있는 변수조합을 빠르고 효율적으로 찾아냄
- Example) 설계변수가 6개, 각 10가지 입력값이 존재하는 경우 총 조합 수는 1000000 가지.
 - Optislang의 AI적인 Meta Model 알고리즘은 이중 의미있는 변수조합 450개만 찾아내 해석
 - Meta model의 완성도를 COP와 같은 지표로 쉽게 확인하면서 최적화 성능 개선
 - 사용자가 미리 예측하지 못한 변수조합간의 문제도 확인 가능
 - **Meta model은 사실상 목표 성능을 구현하기 위해 Design 을 수행하는 AI 와 동일한 개념**