

SPICE NETLIST TO VERILOG GATES CONVERTER

CatalystAD

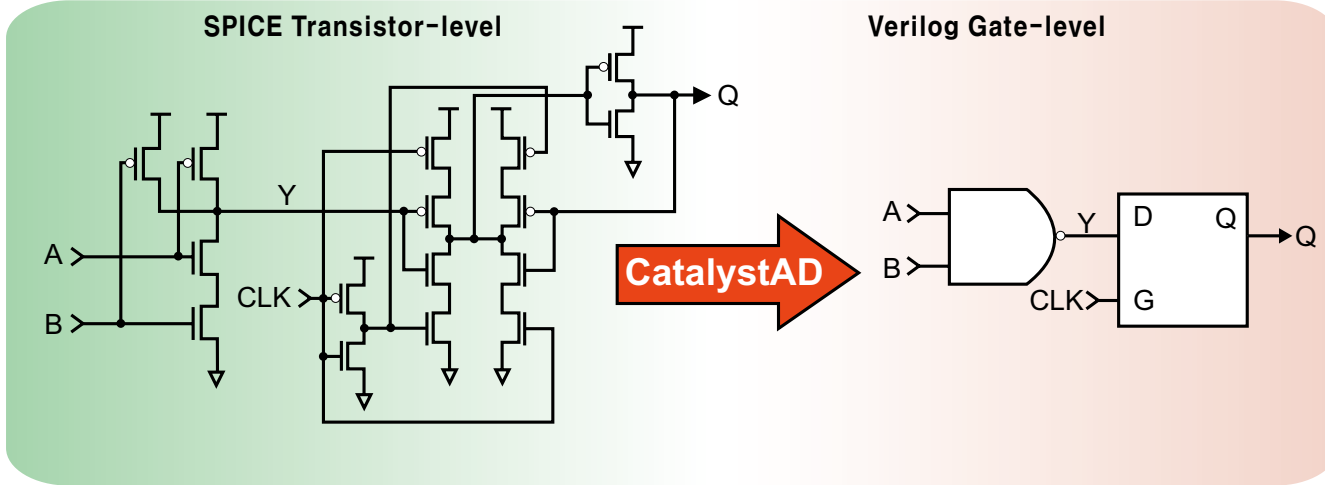
트랜지스터 레벨의 설계를 verilog 게이트 레벨의 표현으로 변환하는 프리미어 툴로서, 마이크로프로세서, DSP, 그래픽, 고속 통신 시장에 적용됩니다.



- 트랜지스터 넷리스트로부터 게이트 레벨 verilog 넷리스트 및 모델을 생성하기 위한 자동화 솔루션을 제공합니다.
- 디자인의 재사용 및 마이그레이션을 위해, 기존 하드 IP 및 커스텀 로직을 리버스-엔지니어링하는데 이상적입니다.
- HSPICE™/SPECTRE™, DSPF의 계층형 또는 플랫폼 넷리스트를 지원합니다.
- 모든 종류의 CMOS/SOI 설계 방식을 처리합니다(standard cell, custom, static, dynamic, combinational, sequential, domino, footed, footless, self-timed, post-charged, cascode, DCVS, pass transistor, barrel-shifters, cross-bar switching structures, m-of-n logic trees, etc).
- 수십만 개의 병렬 경로를 포함하는 24개 이상 입력의 와이드 fan-in pass-gate, 스니크 경로(sneak path) 및 출력 경로의 깊이에 대해 적절한 모델링을 제어합니다.
- CatalystAD는 AccuCore와 함께 완벽한 검증 및 타이밍 모델링 솔루션을 제시합니다.

특징

- 수백만 트랜지스터급의 넷리스트를 손쉽게 처리합니다.
- 풀 커스텀 및 하드 IP 블록을 자동으로 추출합니다.
- 게이트 레벨 시뮬레이션과 합성(synthesis)의 verilog 넷리스트 및 모델을 생성합니다.
- HSPICE™/SPECTRE™, DSPF의 계층형 또는 플랫폼 넷리스트를 지원합니다.
- 사용하기 쉬운 배치 모드 Tcl 스크립트와 설정 파일 인터페이스를 제공합니다.



트랜지스터 레벨 넷리스트로부터 게이트 레벨 함수를 자동으로 추출.

순서

1. SPICE/DSPF 넷리스트를 읽습니다. 이것은 필요한 경우 플랫폼화하며, 전체 계층 경로 정보와 함께 모듈, 포트/넷 이름, 버스를 보존합니다.
2. clock divider, clock doubler, phase shifter, PLL 등의 비-정적 엘리먼트에서 클록을 얻어서, 부울 분석으로 클록을 전파하며, 일련의 노드를 확인합니다. 이는 특정 노드를 통해서 정지 클록을 전파하는 클록 노드입니다.
3. 사용자-정의 패턴을 지정하는 black-box를 통해서 로직으로부터 아날로그 회로를 분리합니다.
4. 자동으로 래치 및 플립-플롭 구조와 변화를 인식합니다.
5. 디자인을 셀로 분할합니다.
6. 알고리즘과 패턴에 기초하여, 자동으로 함수를 추출하고 셀을 분류합니다.
7. 구조적인 verilog 셀의 넷리스트를 생성하며, 선택적으로 “black-boxed” 컴포넌트를 포함합니다.
8. 게이트 레벨의 verilog 모델을 생성합니다.

입력

- 계층형 또는 플랫폼 SPICE/DSPF 넷리스트, RC 정보는 옵션
- 명령어/변수 설정, 핀 정보 및 파일 이름에 대한 .cfg 환경 설정 파일
- 기본적인 .tcl 런타임 스크립트 파일

출력

- 구조적인 verilog 셀의 넷리스트
- 게이트 레벨 verilog 모델 라이브러리
- verilog 템플릿 라이브러리

입력 파일:

Example and.cfg input file:

```
inputs a b c
outputs out
powers vdd
grounds gnd
TOP_VLOG_MODULE and
TOP_SPICE_SUBCKT and
IN_FILE_NAME and.cir
MOSFET_TYPE p pmos
MOSFET_TYPE n nmos
```

Example and.tcl input file:

```
gen_model and.cfg
```

Run-time command:

```
catalystad and.tcl |& tee and.log
```

출력 파일:

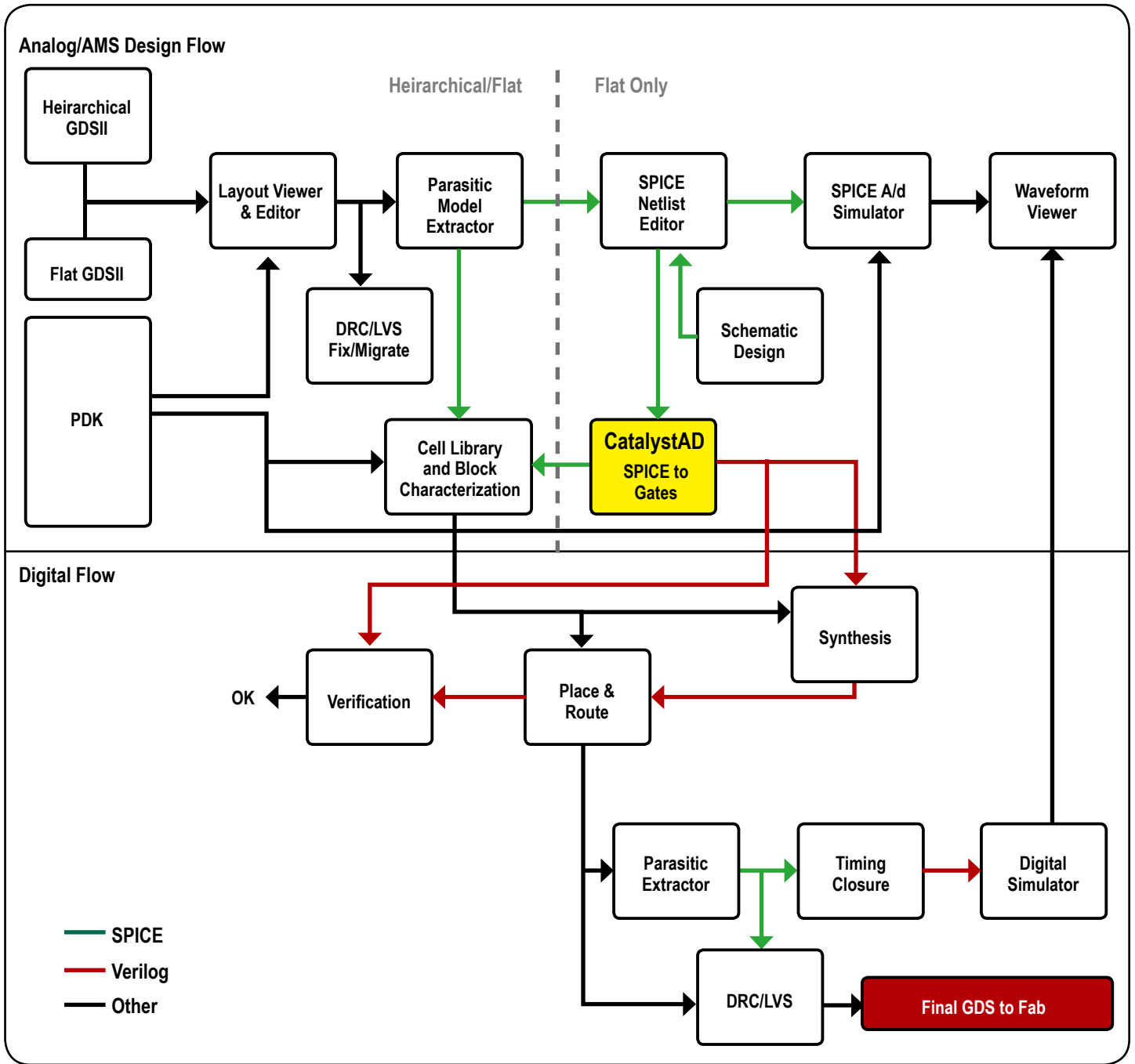
Example and.v netlist output file:

```
module and( out , a , b , c );
output out ;
input a , b , c ;
supply1 vdd ;
supply0 gnd ;
wire z;
and_dc_2 i_and_dc_2( .z(z) , .a(a) , .b(b) , .c(c) );
and_dc_3 i_and_dc_3( .out(out) , .z(z) );
endmodule
```

Example and_vlg.lib model output file (partial):

```
`include "template.v"
`celldefine
module and_dc_2( z , \a<1> , \a<0> );
output z ;
input \a<1> , \a<0> ;
// gate type static for z
wire net_z_1_0 ;
wire z_out_0 ;
wire net_z_1_1 ;
wire z_out_1 ;
buf ( net_z_1_0 , \a<0> );
mux mux_inst_z_0_0 ( z_out_0 , \a<1> , 1'b0 , net_z_1_0 );
not ( net_z_1_1 , \a<0> );
mux mux_inst_z_0_1 ( z_out_1 , \a<1> , 1'b1 , net_z_1_1 );
// inverter for 0 term
wire z_out_0_bar ;
not ( z_out_0_bar , z_out_0 );
// output driver
and ( z , z_out_0_bar , z_out_1 );
endmodule
`endcelldefine
```

Design Center Tool Flow



SILVACO

(주)실바코 코리아

134-020

서울특별시 강동구 천호동 469-1

스타시티빌딩 5층

Phone: 02-447-5421

Fax: 02-447-5420

E-mail: krsales@silvaco.com

WWW.SILVACO.CO.KR

Rev. 010609_03