SmartSpice HPP: SPICE의 정확성을 갖는 고성능 병렬 엔진

1. 소개

기술이 발전함에 따라 회로 설계는 점점 복잡해지는 반면에, 설계 사이클은 점점 짧아지고 있습니다. 결과적으로, 회로 시뮬레이션은 설계 검증에서 병목 현상을 일으키기 쉽습니다. 따라서 아날로그 시뮬레이터는 주어진 시뮬레이션 시간 동안 정확도를 동일하게 유지하면서, 최신 소자를 더 많이 처리해야 합니다.

이와 같은 시뮬레이터의 성능 향상에 대한 요청에 부응하고자, SmartSpice는 새로운 시뮬레이션 엔진인 HPP (High Performance Parallel)를 도입하였습니다. SmartSpice HPP는 멀티 코어 하드웨어 플랫폼을 활용하여, 아날로그 회로의 과도 시뮬레이션을 보다 빨리 처리합니다.

HPP 모드 SmartSpice의 주요 차별점은 분할 기반 시뮬레이션으로서, 분할, 병렬 처리를 통해 매트릭스 로딩 단계, 선형 솔버 연산 및 일반적인 과도 시뮬레이션 과정의 시간을 단축합니다. 블록 동형 및 블록 잠복 기능을 사용할 때에도 추가적인 향상을 이룰 수 있습니다. 마지막으로 기생 감소에 Jivaro를 활용하여, 포스트 레이아웃 시뮬레이션을 위해 SmartSpice HPP에서 최대한 추출할 수 있습니다.

본 애플리케이션 노트에서는 중간 규모에서 대규모 회로까지, 레이아웃 전후의 다양한 설계에 대해 빠르고 정확한 과도 시뮬레이션을 제공하는 SmartSpice HPP의 동작 방식 및 사용 방법에 대해 소개합니다. 또한 다양한 사용 모드를 통해, SmartSpice HPP가 정확도를 유지하면서 일반 SmartSpice보다 용량은 최대 8배, 속도는 최대 40배 우수함을 확인할 수 있습니다.

이 문서의 나머지 부분은 다음과 같이 구성되어 있습니다. 2절에서, SmartSpice HPP와 그 장점에 대한 이해를 돕기 위해 Bordered Block–Diagonal 형태의 개념을 소개합니다. 3절은 HPP의 회로 분할 접근법을 소개하는 반면, 4절은 매트릭스 분할 전략에 초점을 맞추고 있습니다. 5절에서, 블록 동형과 블록 잠복이라는 HPP의 다른 두 가지 중요한 특징을 제시합니다. 6절에서, SmartSpice에서 시뮬레이션 시간을 단축하기 위한 RCL 감축 엔진, Jivaro를 간략히 소개합니다. 7절에서, SmartSpice HPP의 사용법과 본 문서에 제시된 기능을 모두 소개합니다. 8절에서 몇 가지 실험 결과를, 9절에서 결론을 제시합니다. 10절에서 간략한 요약을 제시합니다.

2. Bordered Block-Diagonal (BBD) Form

선형 시스템의 처리를 위한 병렬화 방법을 사용하려면, 시스템 행렬을 경계 블록–대각선 (BBD) 형태로 재구성할 수 있습니다. 그림 1은 글로벌 매트릭스를 BBD 형태로 재정렬하는 방법을 나타냅니다.

바람직한 병렬화를 달성하기 위해 BBD 형태는 분할–후–정복 방식을 따릅니다. 이를 위해 '분할' 관점에서, 큰 행렬을 독립적으로 처리할 수 있는 더 작은 하위 행렬로 나눕니다. 그 다음 "정복"의 관점에서, 원래의 큰 행렬을 처리하기 위해, 소위 "경계 행렬"을 사용하여 하위 행렬을 함께 연결하는 방식을 활용합니다.

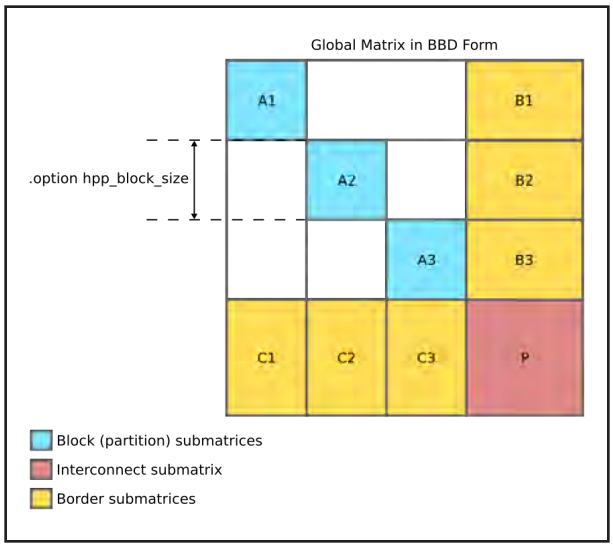

그림 1. BBD 형태로 표현한 전체 행렬

SPICE 시뮬레이션 관점에서, 전체 행렬은 회로 분할 (그림 1의 A1–An), 상호 연결(그림 1의 P) 및 경계 하위 행렬 (그림 1의 B1–Bn 및 C1–Cn)을 나타내는 행렬로 재구성됩니다.

다음 절에서 SmartSpice HPP가 BBD 형태를 활용하여 시뮬레이션 시간을 단축하는 방법에 대해 설명합니다.

3. SmartSpice HPP에서의 회로 분할

SmartSpice HPP의 특징은 분할에 기초한 시뮬레이션입니다. 회로 분할은 현재 두 가지 방법을 통해 이용할 수 있습니다:

1. **그래프 회로 분할:** 회로 토폴로지에 기초하여 자동으로 계산하는 토폴로지 지향 분할
2. **계층적 회로 분할:** 회로의 사용자 정의 계층에 의한 토폴로지 지향 분할

행렬 분할 (수학적 해석과 연계)이 아니라 회로 분할 (물리적 해석과 연계)을 참조한다는 점에 주목하십시오. 그럼에도 불구하고, SmartSpice HPP는 BBD 같은 구조로 회로 분할 (계층 기반 또는 그래프 기반)을 수행합니다. 각 분할은 HPP 엔진에 의해 별도로 처리되어 매트릭스 로딩 단계, 선형 솔버 동작 및 일반적인 과도 시뮬레이션 단계에 소요되는 시간을 단축할 수 있습니다.

4. SmartSpice HPP에서의 행렬 분할

SmartSpice HPP에서 사용자는 어떤 행렬 분할 접근법을 사용할지 제어할 수 있습니다. SmartSpice HPP에서는 두 가지 분할 옵션이 지원됩니다:

1. **Metis 기반 행렬 분할**: 솔버는 사전 단계에서 Metis 중첩 해체 알고리즘을 사용하고, Metis의 정보를 사용하여 BBD 형태로 적절한 행렬을 만듭니다.
2. **회로 기반 행렬 분할:** 솔버는 가능하다면, 회로 분할로부터 BBD 구조에 관한 정보를 재사용하여 BBD 형태로 행렬을 만듭니다.

5. 블록 동형 및 블록 잠복

SmartSpice HPP는 또한 블록 동형과 블록 잠복을 활용하여 시뮬레이션 속도를 높일 수 있습니다. 이 두 가지 기술은 다음과 같습니다.

주어진 허용오차를 고려하여, 여러 블록이 같은 크기, 내부 상태 및 외부 자극을 갖는 경우 동형이라고 합니다. 이 기능이 사용하는 경우, 과도 시뮬레이션에서 HPP 엔진은 잠재적으로 동형인 블록을 동적으로 검사합니다. 그 다음, 각 동형 블록 그룹에 대해 하나의 블록만 시뮬레이션하여 그 결과를 다른 동형 블록에 재사용합니다.

과도 분석에서 일부 연속적 반복 동안, 내부 상태가 주어진 공차 내에 있는 경우 블록은 잠복되어 있다고 합니다. 계산 시점 동안 블럭이 잠복되어 있는 것으로 확인되면, 이 블록은 다시 계산하지 않습니다. 대신 이전 시점의 내부 데이터를 재사용합니다.

6. Jivaro와 함께 SmartSpice HPP

포스트 레이아웃 시뮬레이션을 실행할 때, SmartSpice HPP를 활용하여 정확도를 유지하면서 시뮬레이션 시간을 단축하는 방법은 기생 감소 툴 Jivaro를 이용하는 것입니다. 사용자가 하나의 툴로서 Jivaro에 액세스할 수 없더라도, Jivaro 엔진의 특허받은 수학적 접근 방식이 SmartSpice 내부에 상주하며 MOR (Model Order Reduction)을 수행하여, 높은 정확도를 유지하면서 기생의 복잡도를 줄입니다.

7. SmartSpice HPP 사용법

"smartSpice –hpp" 를 실행하여 SmartSpice HPP를 사용할 수 있습니다. 기본적으로 다른 범용 SPICE 옵션도 다음과 같이 이 플래그에서 설정합니다:

- hsimspeed=3
- bypass=2
- expbypass=1e-3
- cflflag=1

SmartSpice HPP는 윈도우와 리눅스 모두에서 사용할 수 있습니다. 그러나 일반적인 제한으로 과도 분석만 지원합니다. 또한 제한된 수의 SPICE 모델만 지원합니다. (HPP 모드에서 지원하는 모델의 목록은 SmartSpice 사용자 설명서에서 확인할 수 있습니다.)

아래에서 SmartSpice HPP의 솔버 호환에 대한 자세한 사항과 회로 분할, 매트릭스 분할, 동형, 잠복 및 Jivaro 감축의 사용법을 소개합니다.

SmartSpice HPP의 솔버 호환

SmartSpice HPP는 현재 SmartSpice에 구현된 모든 솔루션 (예: XMS, SPEED, SPS)을 지원합니다. 그러나 HPP 기능을 최대한 활용하려면, SPS 솔버를 사용하는 것이 좋습니다. 이는 넷리스트 옵션을 사용하여 수행할 수 있습니다:

```
.option solver = sps
```

또는 명령줄의 인수 (SPS 솔버를 강제로 사용하여 넷리스트에 설정된 ".option solver" 구문을 오버라이드):

```
-forcesolver sps
```

SPS 솔버는 행렬 분할, 동형, 잠복에 필요합니다. 다른 솔버는 현재 이러한 기능을 지원하지 않습니다.

회로 분할 활용

SmartSpice HPP에서 기본적으로 그래프 분할이 설정됩니다. 사용자가 계층적 분할을 사용하려면 ".option user_hier_level"을 설정해야 합니다. 각 분할 모드와 관련된 옵션 목록은 표 1에서 확인할 수 있습니다. 참조한 회로 분할 모드는 다음 요구 사항 목록을 충족하는 경우에만 동작합니다.

그래프 분할 요구 사항:

- 없음

계층 분할 요구 사항:

- 회로의 사용자 정의 계층

행렬 분할 활용

SmartSpice HPP에서 기본적으로 Metis 기반 행렬 분할이 설정됩니다. 행렬 분할을 제어하려면 다음과 같이 ".option matrix_partitioning_mode"를 사용해야 합니다:

- **.option matrix_partitioning_mode=0**

  회로 기반 행렬 분할을 사용합니다. (이 모드만 동형 및 잠복을 지원합니다.)

- **.option matrix_partitioning_mode=2 (default)**

  Metis 기반 행렬 분할을 명시적으로 사용합니다.

- **.option matrix_partitioning_mode=3**

  행렬 분할을 사용하지 않고 단일 전역 행렬을 작성합니다 (멀티 스레드를 계속 사용할 수 있습니다.)

Metis 기반 행렬 분할의 경우, 솔버에서 생성한 상호 연결 행렬이 회로 기반 행렬 분할에서 제공하는 행렬보다 훨씬 작을 수 있으므로, 성능이 향상될 수 있습니다. 그러나 Metis 기반 분할은 동형 및 잠복을 지원하지 않습니다.

참조한 행렬 분할 모드는 다음 요구 사항 목록을 충족하는 경우에만 동작합니다.

행렬 분할 요구 사항:

- SPS Solver

동형 및 잠복을 활용

블록 동형은 계층적 회로 분할 및 솔버 SPS를 사용하는 경우에만 HPP에서 사용할 수 있습니다. 그래프 분할 방법에서 동형은 지원되지 않습니다. 표 2에 동형 및 잠복에 적용할 수 있는 옵션 목록을 요약하였습니다. 요구 사항 목록은 다음과 같습니다.

Option Name	*Option Description*
Graph Partitioning Options	
hpp_partition_count=<N>	Sets desired partitions count to <N> for graph partitioning. By default, the number of partitions is determined automatically for better performance.
Hierarchical Partitioning Options	
user_hier_level=<L>	Specifies the hierarchy level to split the partitioning into interconnect and block partitions. All the cells from top level to level <L-1>, will compose the interconnect partition (P in Fig. 1). All the cells from level <L> on, will compose the block partitions (A1-An in Fig. 1). If not defined, an auto detection algorithm will calculate <L> to minimize interconnect network and maximize number of separate blocks.
hpp_block_size=<S>	Specifies the desired size in terms of nodes of block partitions (A1-An in Fig. 1). The default value is 100.
hpp_hiersim	Enables automatic detection of array blocks (useful for memory designs, flat-panel displays, etc.).

표 1. 그래프 및 계층 기반 회로 분할에 적용할 수 있는 옵션 목록

Option Name	*Alias Name*	*Option Description*
Isomorphism Options		
hpp_block_isomorphism	iso	Activates isomorphic blocks detection algorithm.
hpp_block_isomorphism_reltol	iso_reltol	Isomorphism relative voltage error tolerance for blocks. Default is 1e-3.
hpp_block_isomorphism_abstol	iso_abstol	Isomorphism absolute current error tolerance for blocks. Default is 1e-9.
hpp_block_isomorphism_vntol	iso_vntol	Isomorphism absolute voltage error tolerance for blocks. Default is 5.e-5.
Latency Options		
hpp_block_latency	lat	Activates latent blocks detection algorithm.
hpp_block_latency_reltol	lat_reltol	Latency relative voltage error tolerance for blocks. Default is 1e-3.
hpp_block_latency_abstol	lat_abstol	Latency absolute current error tolerance for blocks. Default is 1e-9.
hpp_block_latency_vntol	lat_ vntol	Latency absolute voltage error tolerance for blocks. Default is 5.e-5.

표 2. 동형 및 잠복에 적용할 수 있는 옵션 목록

동형 요구 사항:

- 솔버 SPS
- 계층적 회로 분할
- 회로 기반 행렬 분할

지연 시간 요구사항:

- 솔버 SPS
- 그래프 또는 계층적 회로 분할
- 회로 기반 행렬 분할

Jivaro 감축 기능을 활용

SmartSpice HPP는 기본적으로 기생 감소 기법이 설정되지 않습니다. 포스트 레이아웃 시뮬레이션을 하는 경우, SmartSpice 내부에서 기생 감소를 위한 Jivaro를 이용하는 것이 좋습니다. 표 3에 Jivaro 감축에 적용할 수 있는 옵션의 목록을 요약하였습니다. 요구 사항 목록은 다음과 같습니다.

Jivaro 감축 요구사항:

- 없음

8. 실험 결과

SmartSpice HPP의 장점을 소개하기 위해, 이 절에서 설계에 대한 실험 결과를 소개합니다. 이를 위해, BSIM4 모델을 기반으로 45nm 기술로 설계한 포스트 레이아웃 동기식 2MB (256 64비트 워드) SRAM을 사용했습니다. 전체 메모리 설계를 사용하여 일반적인 타이밍 및 전력 측정을 시뮬레이션합니다. 즉, 회로를 임계 경로로 줄이기 위해 설계 최적화를 적용하지 않았습니다. 실험은 Intel(R) Xeon(R) CPU E5-2699 v3 @2.30GHz 서버에서 수행하였습니다.

실험 설정

이러한 실험을 위해, 우리는 전체 디자인에서 단일 메모리 셀의 특성을 연습했습니다. 메모리 설계에서 잠재적으로 중요한 경로의 타이밍과 파워 동작을 모두 조사하려고 할 때 SmartSpice HPP를 어떻게 활용할 수 있는지 설명하고자 하였습니다.

이를 위해, 다음과 같이 과도 분석을 수행했습니다 (Pi는 i 번째 관심 기간을 나타냅니다):

P0: 대상 셀에 로직 1을 기록하여 알려진 상태로 전환

P1: 대상 셀에 로직 0을 기록

P2: 다른 메모리 셀에 로직 1을 기록하여 버스를 클린

P3: 대상 셀에서 읽기를 시도하고, 로직 0이 성공적으로 기록되었는지 확인

P4: 유휴 기간 동안 버스를 클린

P5: 대상 셀에 로직 1 기록

P6: 다른 메모리 셀에 로직 0을 기록하여 버스를 클린

P7: 대상 셀에서 읽기를 시도하고, 로직 1이 성공적으로 기록되었는지 확인

시뮬레이션이 완료되면, 다음과 같은 처리 후 측정을 수행하였습니다:

delay_hl: 대상 신호가 로직 1에서 로직 0으로 전환되는 경우, P3에서 클럭 대 출력 지연 (50%~50%) 측정

delay_lh: 대상 신호가 로직 0에서 로직 1로 전환되는 경우, P7에서 클럭 대 출력 지연 (50%~50%) 측정

Option Name	*Option Description*
rcl_reduction_type=val	Selects RCL netlist reduction engine. 0 - Reduction disabled (default) 2 - “Jivaro”
rcl_reduction_reltol=val	Specifies maximal relative error for transient responses of reduced and original circuits. (Default: 0.05.)
rcl_reduction_max_freq=val	Defines frequency range for which RCL reduction guarantees consistency of transient responses of reduced and original circuits. The default is 5.0e9 Hz.
rcl_reduction_flow_mode_hpp=val	Controls RCL-reduction in Graph-based circuit partitioning. 0 - Reduction before graph partitioning 1 - Reduction after graph partitioning (default)
rcl_reduction_minc=val	Defines a threshold for filtering capacitor values. All capacitor values less than <val> are replaced by an open circuit. Default is -1 (inactive).
rcl_reduction_minr=val	Defines a threshold for filtering resistor values. All resistor values less than <val> are replaced by short circuit. Default is 1E-15.
rcl_reduction_maxr=val	Defines a threshold for filtering Resistor values. All resistor values greater than <val> are replaced by open circuit. Default is 1E15.

표 3. Jivaro 축소에 적용할 수 있는 옵션 목록

slew_hl: 대상 신호가 로직 1에서 로직 0으로 전환되는 경우, P3에서 출력의 슬루율 (90%~10%) 측정

slew_lh: 대상 신호가 로직 0에서 로직 1로 전환되는 경우, P7에서 출력의 슬루율 (10%~90%) 측정

write0_pw: 대상 셀에 로직 0을 기록하는 P1에서 P2까지의 평균 전력 측정

write1_pw: 대상 셀에 로직 1을 기록하는 P5에서 P6까지의 평균 전력 측정

read0_pw: 대상 셀에서 로직 0을 읽는 P3에서 P4까지의 평균 전력 측정

read1_pw: 대상 셀에서 로직 1을 읽는 P3에서 P4까지의 평균 전력 측정

마지막으로, SPS 솔버, 그래프 회로 분할, Metis 기반 매트릭스 분할을 사용한 SmartSpice HPP 기본 설정과 기본 설정을 사용한 일반 SmartSpice 사이의 비교를 수행했습니다. 블록 동형 및 블록 잠복은 사용하지 않았습니다.

확장성 분석

여기서, SmartSpice HPP가 일반 SmartSpice와 비교했을 때 얼마나 더 확장성이 있는지 조사하였습니다. 이를 위해, 일반 SmartSpice를 멀티 스레드 ("-P <n>" 명령어 인자, <n>은 스레드 수)로 실행하여 단일 스레드와 그 성능을 비교하였습니다. 그 다음, SmartSpice HPP로 동일한 작업을 수행하고 두 시뮬레이터의 속도 상승 곡선을 도출하였습니다. 그림 3은 그 결과를 나타냅니다.

그림에서 볼 수 있는 것처럼, SmartSpice HPP 16 스레드는 SmartSpice HPP 단일 스레드보다 13.3배 빠릅니다. 이는 동일 조건에서 5.3배 빠른 일반 SmartSpice보다 최대 8배 더 확장성이 있다는 것을 보여줍니다.

이러한 결과는 HPP 알고리즘과 시뮬레이션 전략이 최신 멀티 코어 하드웨어 플랫폼을 활용하여, 아날로그 회로의 과도 시뮬레이션에서 속도를 높이는 방법을 나타냅니다.

성능 및 정확도 분석

여기서, SmartSpice HPP를 일반 SmartSpice와 비교했을 때의 속도 향상 및 정확도 측면에 대한 영향을 조사하였습니다. 이를 위해, SmartSpice HPP를 다음 두 가지 호환 기능과 함께 시험하였습니다: (1) 실행 시간과 정확도를 절충하는 터보 모드, (2) RCL 감소를 위한 Jivaro. 모든 시뮬레이션은 16 스레드 (-P 16)로 실행하였습니다. 그림 4는 그 결과를 나타냅니다.

그림에서 볼 수 있는 것처럼, HPP를 사용하면 슬루 측정에 최대 오차 약 2%, 전력 측정에 0.1% 미만으로 속도가 대략 5배 향상됩니다. Jivaro를 도입하면, 거의 동일한 정확도를 유지하면서 속도가 8배 이상 빨라집니다. 마지막으로, Jivaro와 Turbo를 함께 사용하면 HPP가 일반 SmartSpice보다 약 40배 더 빠를 수 있지만, 측정값에 대한 최대 오차는 슬루 수치에서 9% 미만, 지연 측정에서 약 2% 그리고 전력 수치에서 1% 미만으로 유지됩니다.

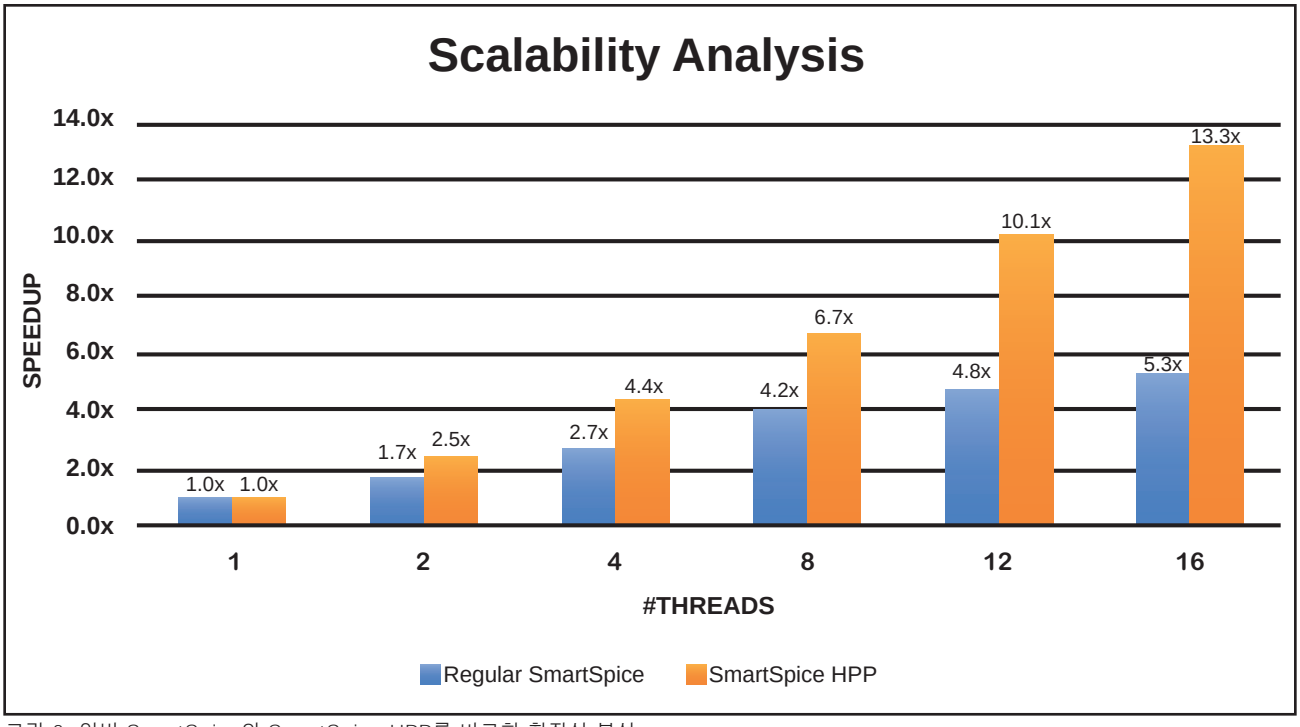

그림 3. 일반 SmartSpice와 SmartSpice HPP를 비교한 확장성 분석

이러한 결과는 SmartSpice HPP가 다른 SmartSpice 기능 (예: 터보 모드 및 Jivaro RCL 감소)을 얼마나 효과적으로 통합할 수 있는지를 나타냅니다. 또한 타이밍 측정에 허용 가능한 정확도와 전력 측정에 매우 높은 정확도를 유지하면서 HPP가 얼마나 더 빠를 수 있는지를 나타냅니다. SmartSpice HPP를 사용하여 시뮬레이션 시간이 약 90시간에서 2.2시간으로 단축되었습니다. 이는 설계자의 관점에서 생산성이 크게 향상되었음을 나타냅니다.

전력과 지연 측정이 회로 설계자에게 가장 중요한 특성 중의 하나가 되고 있으므로, 매우 정확한 전력과 지연 수치를 통해 회로 시뮬레이터의 속도가 40배 향상될 수 있다면 설계자의 생산성을 크게 높일 수 있습니다. 이는 FastSPICE 엔진이 이룰 수 없는 것입니다.

9. 최종 의견

이 절에서 SmartSpice HPP를 더 잘 활용하기 위한 몇 가지 권장 사항을 제시합니다. 자세한 내용은 다음을 확인하십시오.

회로 분할 권장 사항:

A. 회로의 계층 구조를 알 수 없는 경우

1) 그래프 회로 분할을 사용하는지 확인합니다. 이 경우, ".option user_hier_level"을 사용하지 마십시오.
2) 분할 수를 제어하려면 ".option hpp_partition_count"를 사용합니다.
3) 모델 평가의 속도를 높이려면 ".option hpp_block_latency"를 시도합니다.

B. 회로 계층 구조가 알려져 있으며, 동일한 크기의 셀 (예: TFT, SRAM)을 다수 포함하는 경우

1) 계층적 회로 분할을 사용하는지 확인합니다. 이 경우, ".option user_hier_level"을 사용합니다.
2) 계층 수준을 설정하는 ".option user_hier_level=<n>"을 시도할 수 있지만, SmartSpice HPP도 자동으로 이를 처리할 수 있습니다.
3) ".option hpp_block_size"는 매우 작은 블록으로부터 시뮬레이션을 보호하는 데 도움이 될 수 있습니다.
4) ".option hpp_hiersim"은 자동 메모리 어레이 감지 기능을 제공하므로, 시뮬레이션 시간을 단축하는 데 도움이 될 수 있습니다.
5) ".option hpp_block_isomorphism" 및 ".option hpp_block_latency"를 사용하면 모델 평가의 속도를 높이는 데 도움이 될 수 있습니다.

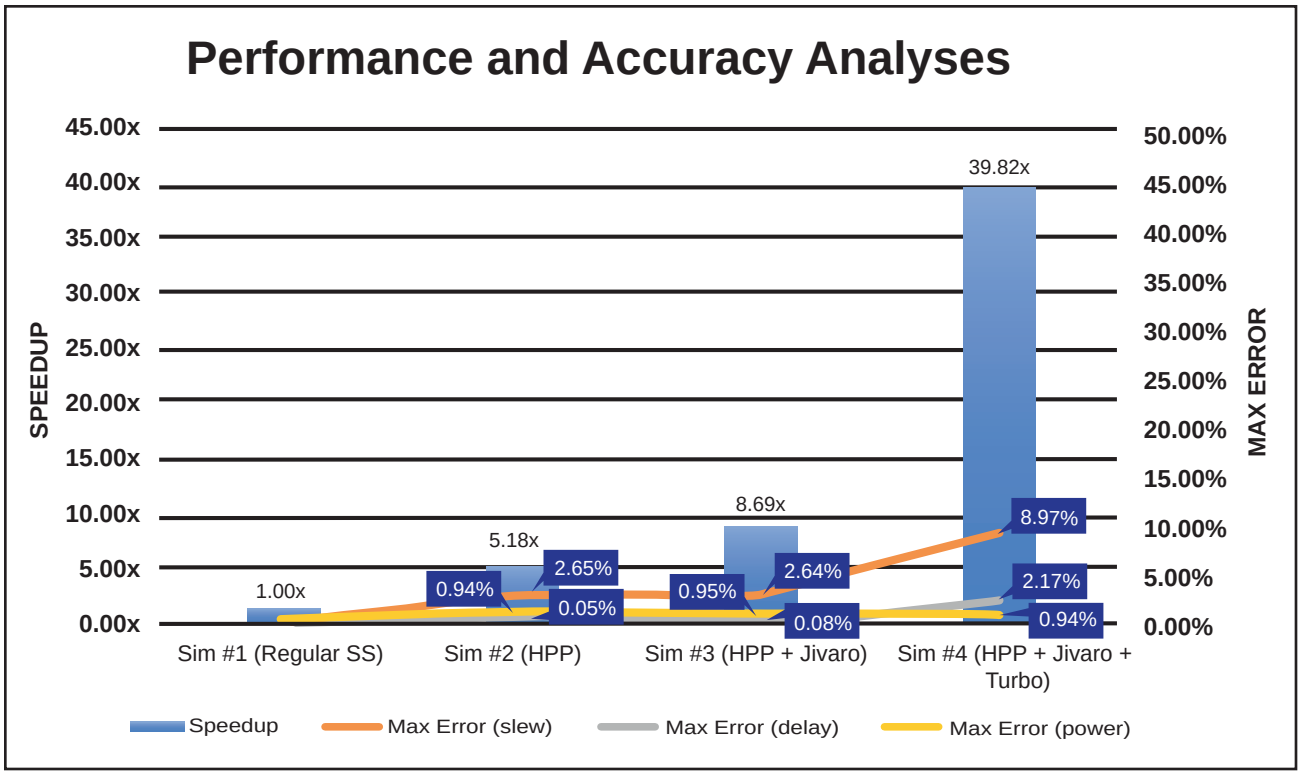

그림 4. 일반 SmartSpice와 SmartSpice HPP를 비교한 성능 및 정확도 분석

기타 일반 권장 사항:

SmartSpice HPP는 레이아웃 전후의 회로 과도 시뮬레이션 속도를 높이기 위해 멀티 스레드 모드 (-P N, N>1)에서 동작해야 합니다. HPP 모드는 최신 멀티 코어 하드웨어 플랫폼에서 멀티 스레드 알고리즘을 통해 확장성과 최적화된 모델 및 솔버를 구현합니다.

대규모 회로는 비회전 재정렬 알고리즘으로 SPS 솔버를 사용하는 것이 좋습니다. 이 알고리즘은 다음 옵션을 설정하여 사용할 수 있습니다:

```
.option pivot=1 pivtol=1e-13
```

8절에서 제시했듯이, SmartSpice HPP는 RCL 감소를 위해 고성능 "-turbo" 모드 및 Jivaro와 호환됩니다. 이 두 가지 기능은 대규모 포스트 레이아웃 넷리스트 시뮬레이션에 적극 권장됩니다.

10. 요약

본 애플리케이션 노트에서 SmartSpice HPP와 관련된 주요 개념을 소개하고 사용 방법을 설명했습니다. 상이한 정확도/성능 절충 결과를 나타내는 각각의 시뮬레이션 시나리오를 제시하였습니다. 시뮬레이션 결과는 SmartSpice HPP가 일반 SmartSpice보다 최대 8배 더 확장성이 있다는 것을 보여줍니다. 또한 다른 SmartSpice 기능과 통합될 경우, HPP가 최대 40배 빨라지는 동시에 사용자가 시뮬레이션 정확도를 제어할 수 있음을 보여줍니다. SmartSpice HPP는 레이아웃 전후의 중규모 회로에서 대규모 회로에 이르기까지 다양한 설계에서 빠르고 정확한 과도 시뮬레이션에 폭넓게 사용할 수 있습니다.