An Adaptive Parallel Flow for Power Distribution Network Simulation Using Discrete Fourier Transform

Xiang Hu[1], Wenbo Zhao[2], Peng Du[2],

Amirali Shayan[2], Chung-Kuan Cheng[2]

[1]ECE Dept., [2]CSE Dept.

University of California, San Diego

01/19/2010

Outline

- Motivation
- Adaptive simulation flow using discrete Fourier transform (DFT)
  - Basic DFT flow
  - Problems with the basic DFT flow
  - Adaptive DFT flow
- Experimental results
  - Error analysis
  - Time efficiency of the adaptive flow
  - Time efficiency of parallel processing
- Conclusions

Outline

- Motivation
- Adaptive simulation flow using discrete Fourier transform (DFT)
  - Basic DFT flow
  - Problems with the basic DFT flow
  - Adaptive DFT flow
- Experimental results
  - Error analysis
  - Time efficiency of the adaptive flow
  - Time efficiency of parallel processing
- Conclusions

PDN Simulation: Why Frequency Domain?

- Huge PDN netlists
  - Time-domain simulation: serial - slow
  - Frequency-domain simulation: parallel – fast
- Frequency dependent parasitics
- Simulation results
  - Time-domain: voltage drops, simultaneous switching noise (SSN) – input dependent
  - Frequency-domain: impedance, anti-resonance peaks – input independent

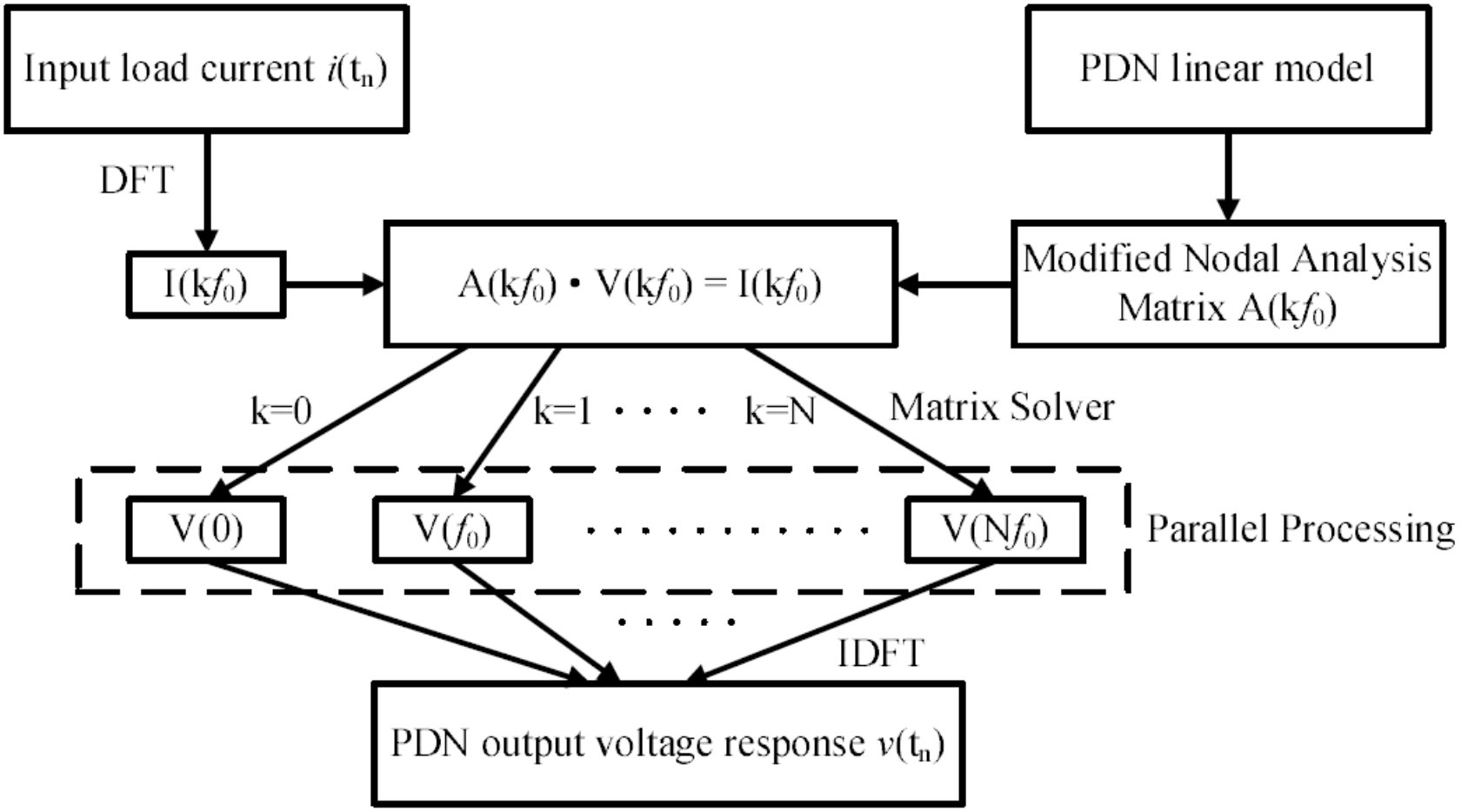
Transform Operations: Why Discrete Fourier Transform

- Laplace Transform [Wanping ’07]
  - Input: Series of ramp functions
  - Output: Rational expressing via vector fitting
    - Vector fitting may introduce large errors
  - Choice of frequency samples is case dependent
- Discrete Fourier Transform (DFT)
  - Input: periodic signal
  - Inverse DFT is straightforward: vector fitting is not needed
  - Frequency sample points with uniform steps

Outline

- Motivation
- Adaptive simulation flow using discrete Fourier transform (DFT)
  - Basic DFT flow
  - Problems with the basic DFT flow
  - Adaptive DFT flow
- Experimental results
  - Error analysis
  - Time efficiency of the adaptive flow
  - Time efficiency of parallel processing
- Conclusions

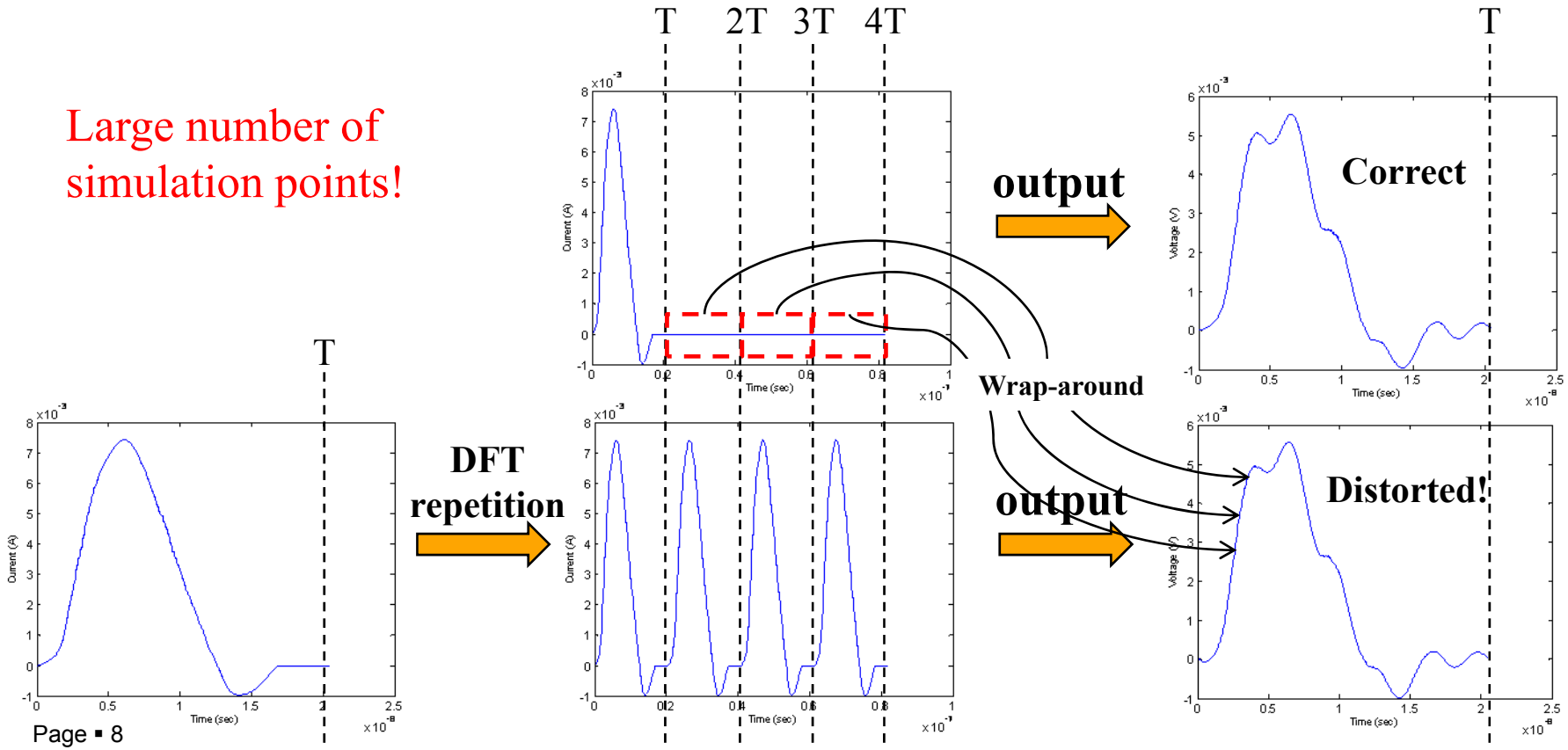
Basic DFT Simulation Flow



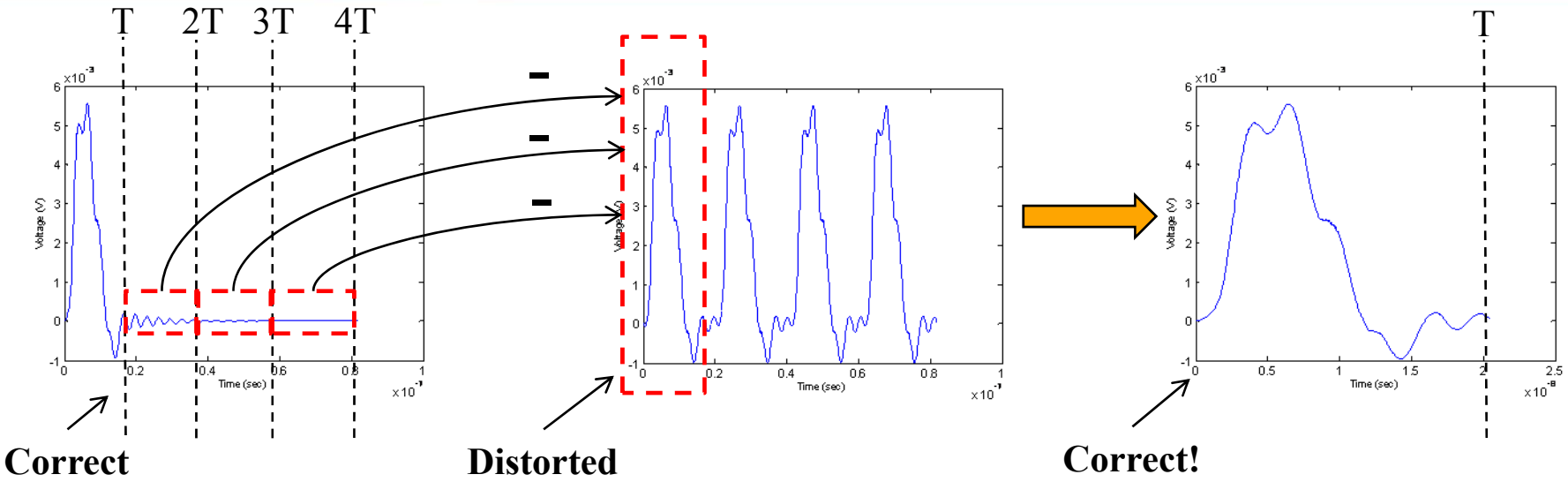
Problem with Basic DFT Flow

- "Wrap-around effect" requires long padding zeros at the end of the input
  - Periodicity nature of DFT
- Small uniform time steps are needed to cover the input frequency range

Large number of simulation points!



Adaptive DFT Simulation

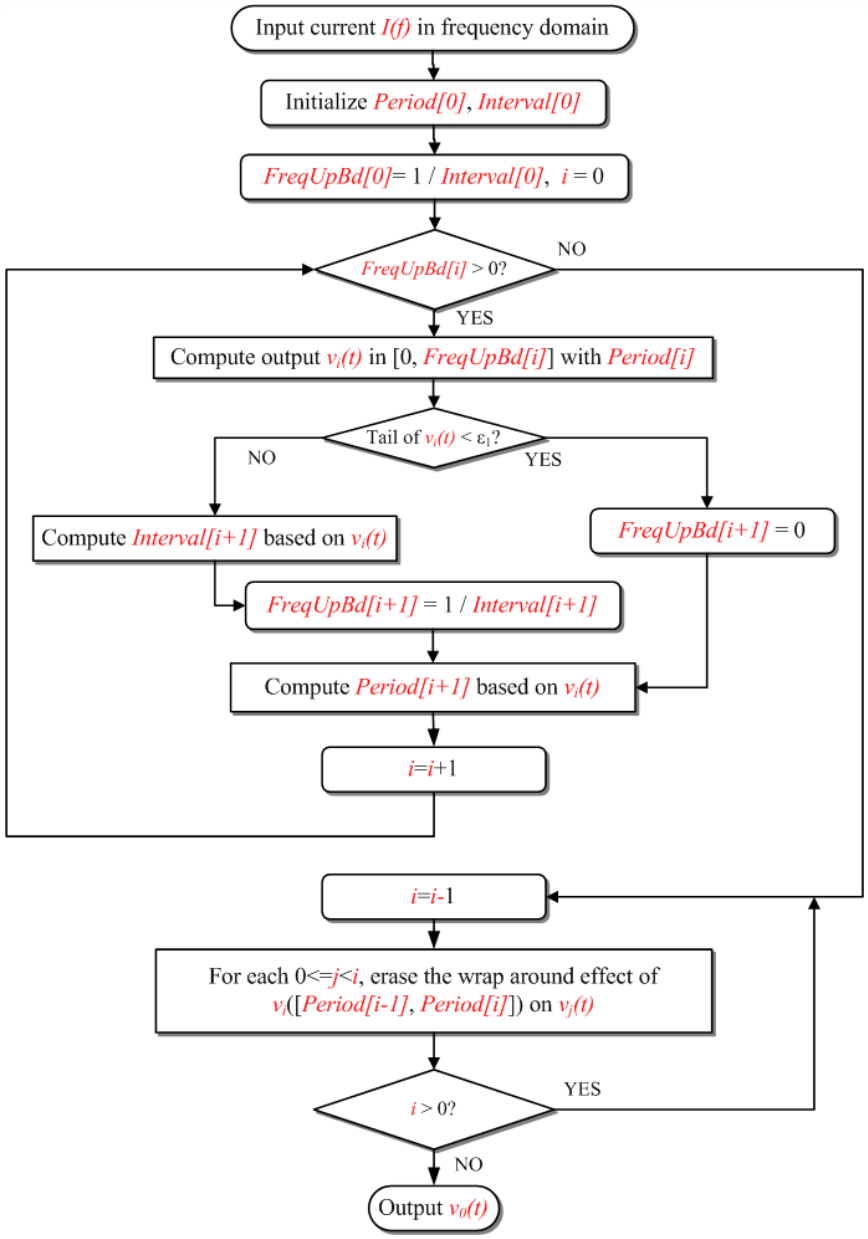


- Basic ideas of the adaptive DFT flow: cancel out the wrap-around effect by subtracting the tail from the main part of the output
  - Main part of the output: obtained with small time step and small period; distorted by the wrap-around effect
  - Tail of the output: low frequency oscillation; can be captured with large time steps

Total number of simulation points is reduced significantly!

Adaptive DFT Flow

- *Period[i]*: the input period at each iteration
- *Interval[i]*: the simulation time step at each iteration
- *FreqUpBd[i]*: the upper bound of the input frequency range at each iteration
- $v_i(t)$: tentative time-domain output within the frequency range [0, FreqUpBd] at each iteration
- Iteration #1: obtain the main part of the output
- Iteration #2~k: capture the oscillations in the tail of the output (high, middle, and low resonant frequencies)
- For each iteration #i, i=k, k-1, …, 2, subtract the captured tail from the outputs at iteration #j, j<i to eliminate the wrap-around effect

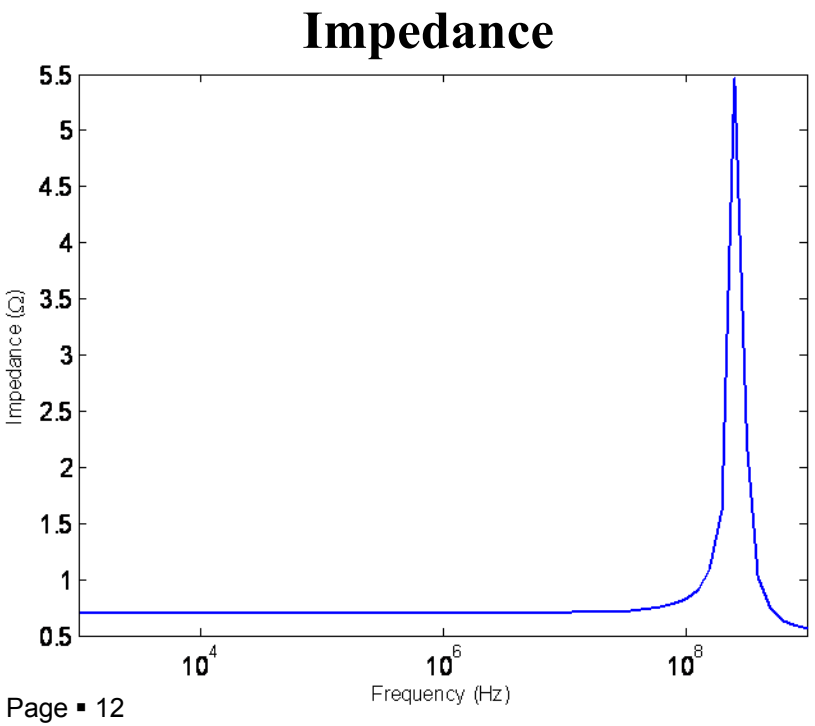


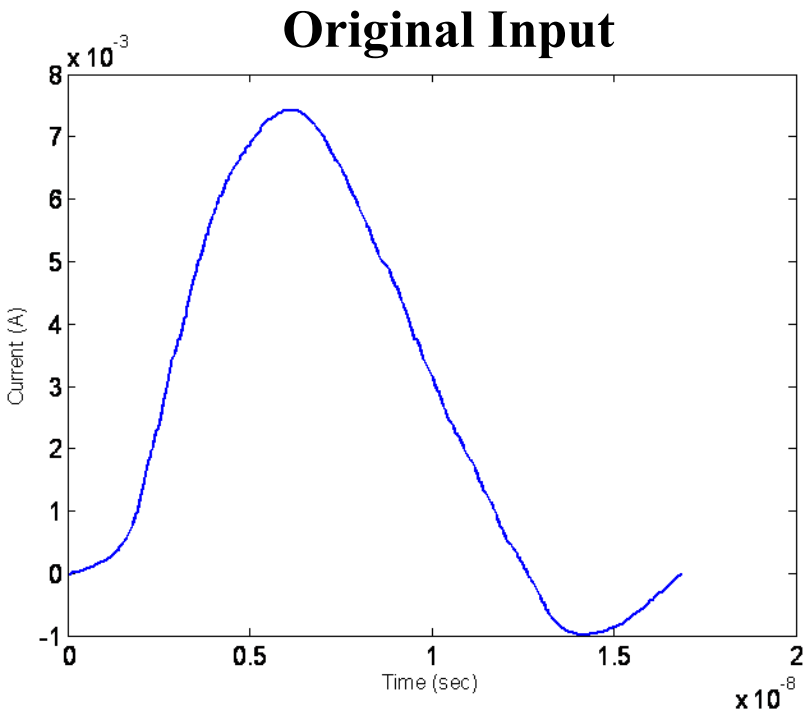
Outline

- Motivation
- Adaptive simulation flow using discrete Fourier transform (DFT)
  - Basic DFT flow
  - Problems with the basic DFT flow
  - Adaptive DFT flow
- Experimental results
  - Error analysis
  - Time efficiency of the adaptive flow
  - Time efficiency of parallel processing
- Conclusions

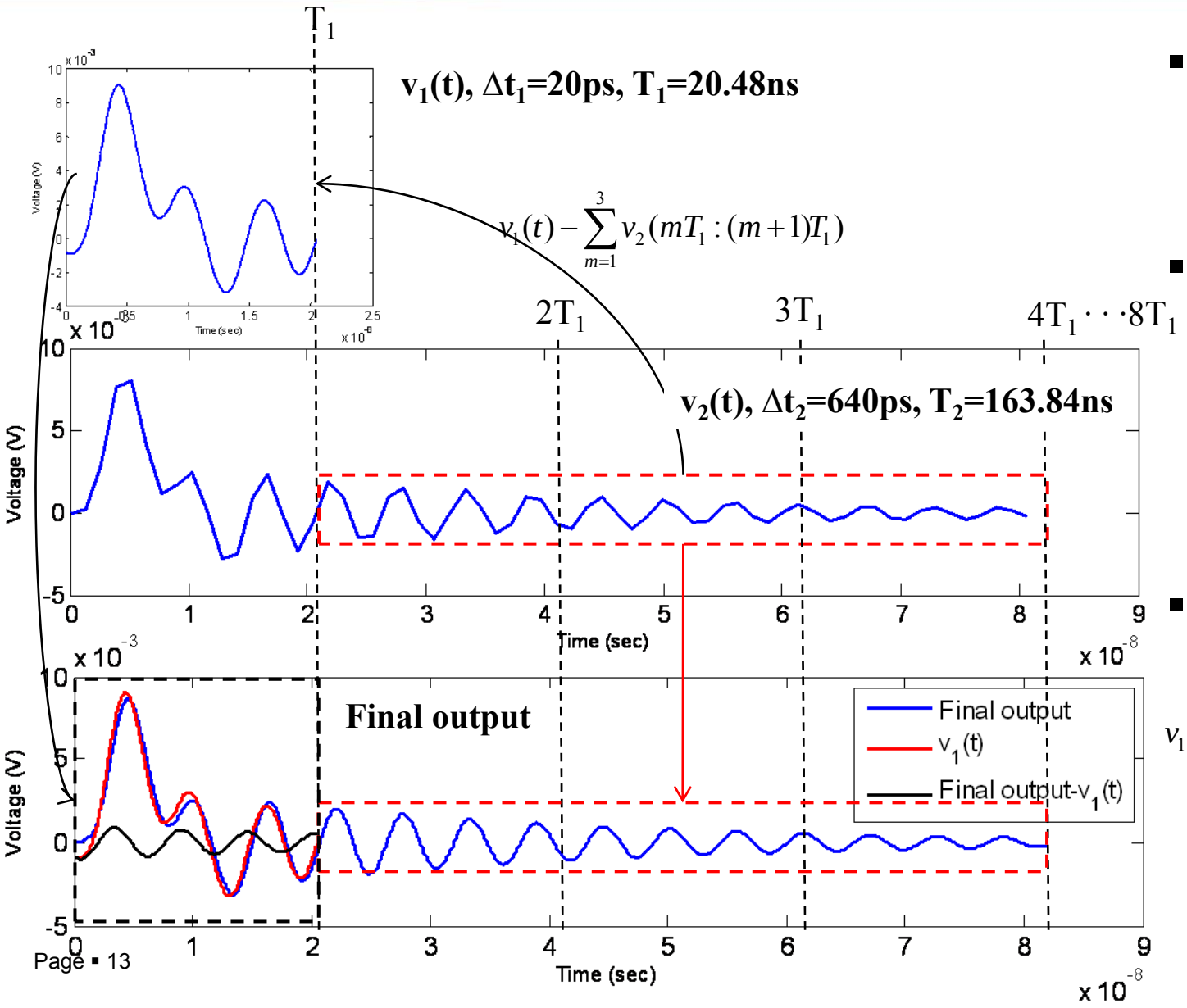
Experimental Results: Test Case & Input

- Test case: 3D PDN
  - One resonant peak in the impedance profile
- Input current
  - Time step: $\Delta t = 20ps$
  - Duration: $T_0 = 16.88ns$



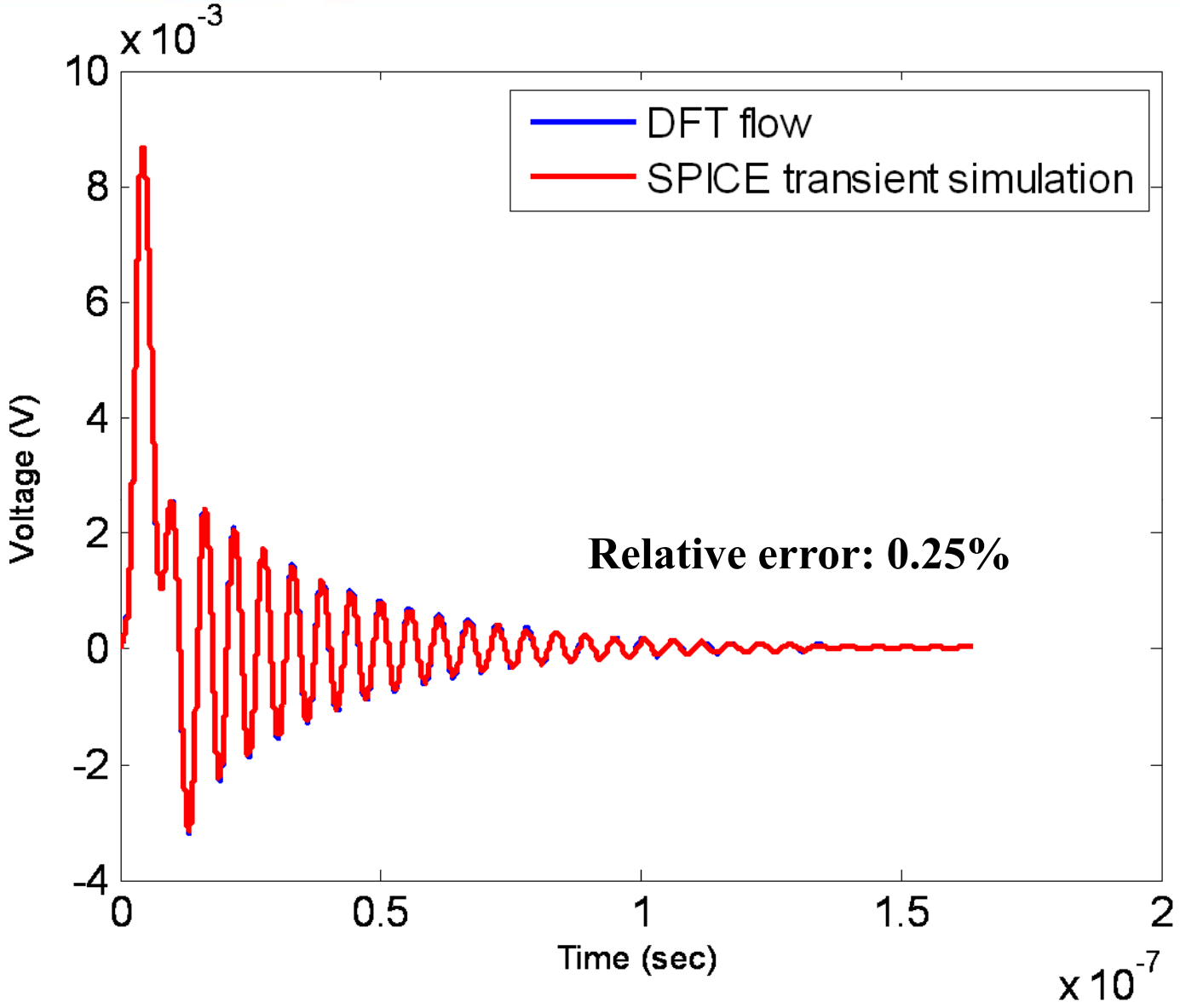


Experimental Results: Adaptive Flow Process



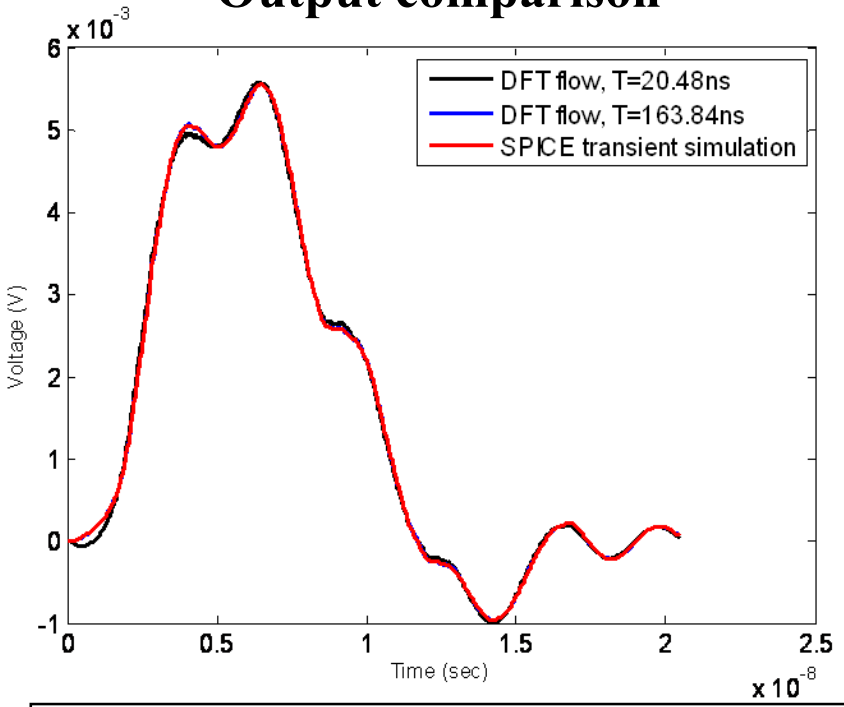
- Iteration #1: $v_1(t)$
  - Δt_1=20ps
  - T_1=20.48ns
- Iteration #2: $v_2(t)$
  - $\Delta t_2 = 64\Delta t_1$ = 640ps
  - $T_2 = 8T_1$ = 163.84ns
- Final output:
  - Main part: $v_1(t) - \sum_{m=1}^{3} v_2(mT_1 : (m+1)T_1)$
  - Tail: $v_2(T_1 : 8T_1)$

Experimental Results: DFT Flow vs. SPICE

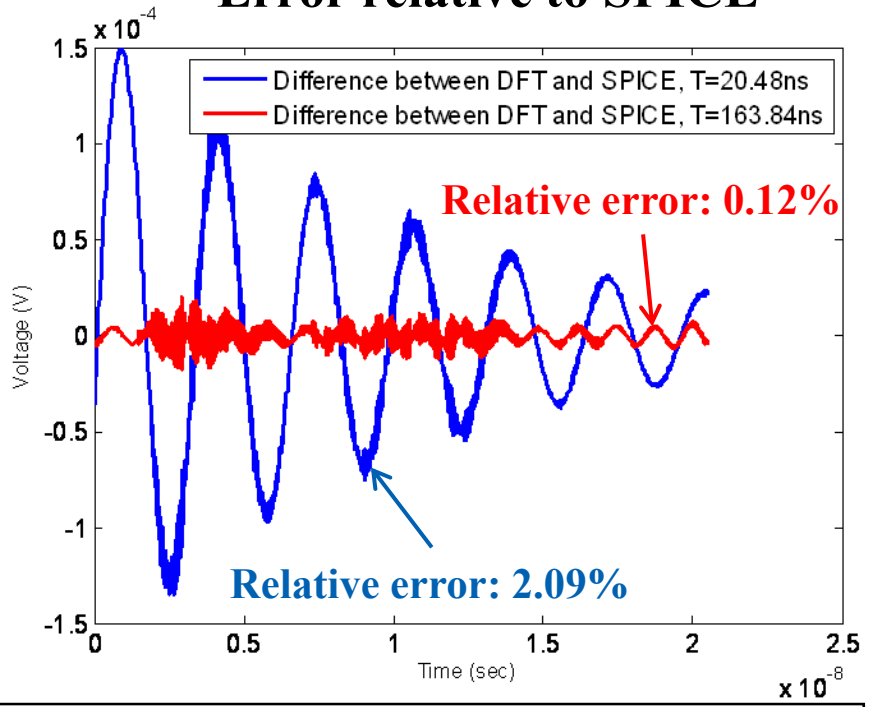


Error Analysis: Error Caused by Wrap-around Effect

Output comparison



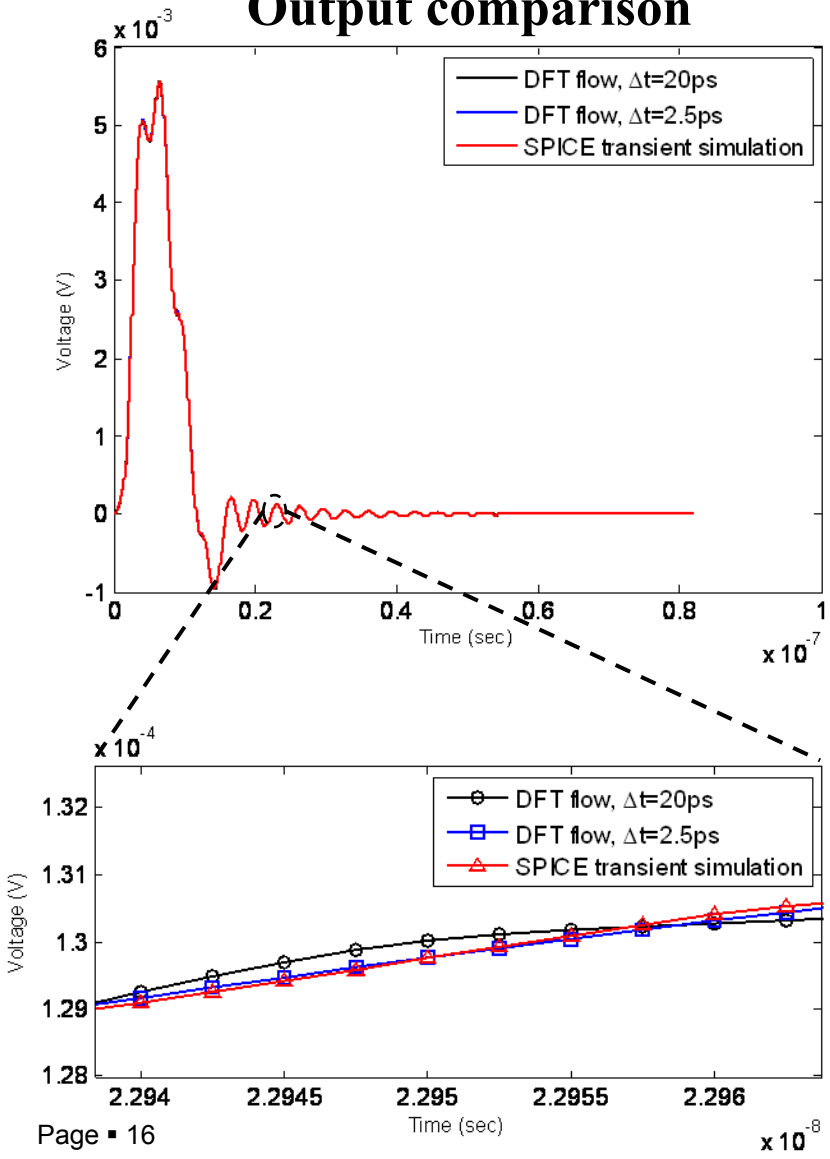
Error relative to SPICE



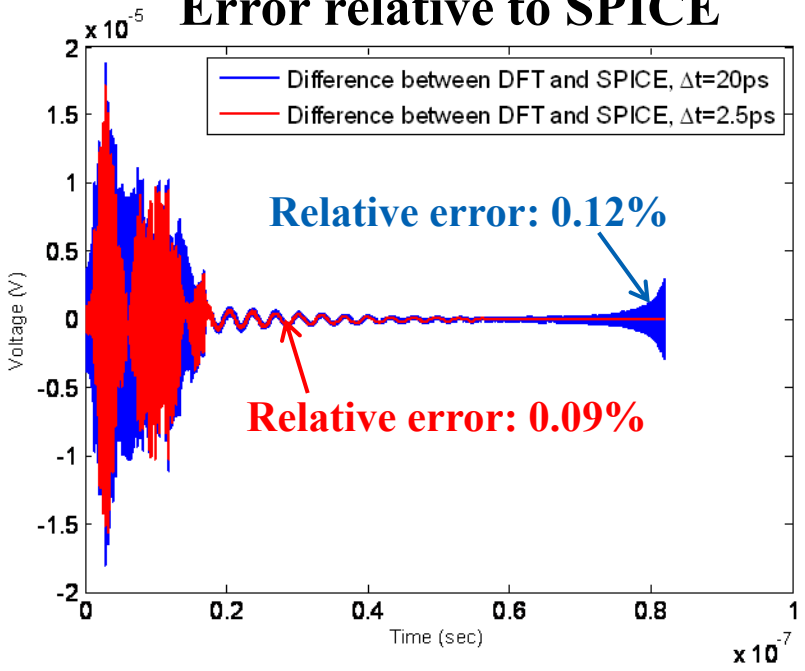
Theorem 1: Let v_0 be the initial value of the output voltage. Suppose $\{\int_{T-T'}^{T}(v(t)-v_0)^2\,dt\}^{1/2} \le \epsilon$ for some $T' > 0$, then the mean square error, i.e., $\|\Delta v(t)\|$ is bounded by $\epsilon\sqrt{T/T'}$.

Error Analysis: Error Caused by Different Interpolation Methods

Output comparison



Error relative to SPICE

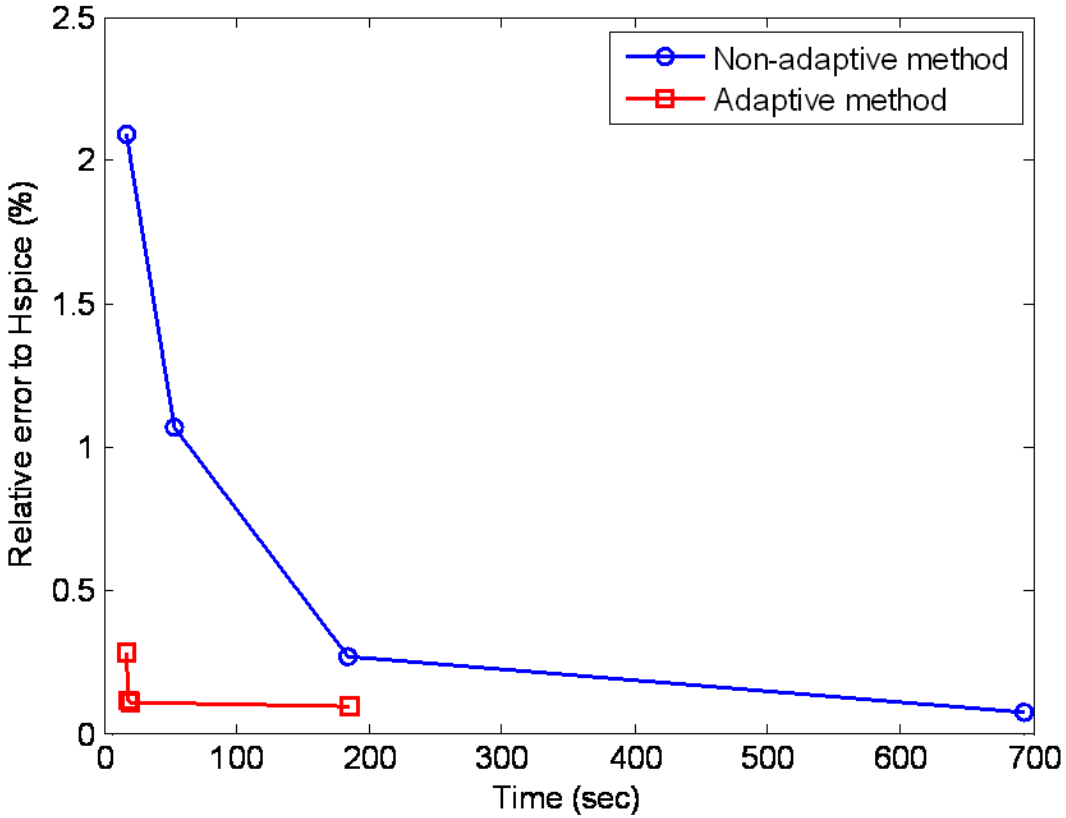


- SPICE: PWL interpolation
- DFT: sinusoidal interpolation

$$v(t) = \sum_{k=-N/2+1}^{N/2} V(f_k) e^{j2\pi tk/T}$$

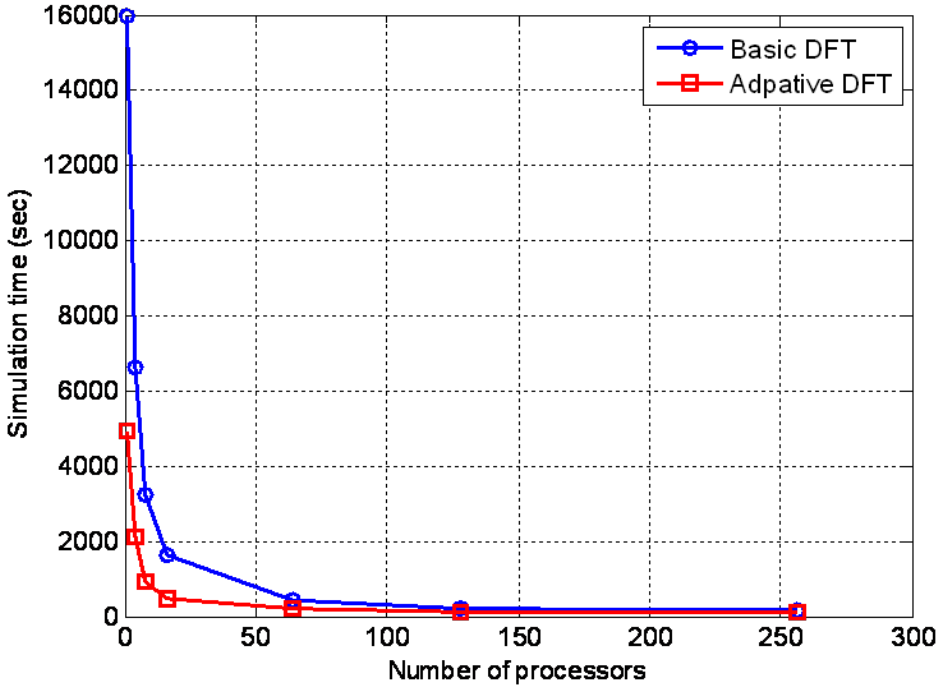
Time Complexity Analysis: Adaptive vs. Non-adaptive

- Adaptive flow time complexity: $O(\sum_{i=1}^{k} T_i / \Delta t_i)$
  - T_i: simulation period at iteration #i, $T_1 < T_2 < \cdots < T_k$
  - Δt_i: simulation time step at iteration #i, $t_1 < t_2 < \cdots < t_k$
- Non-adaptive flow time complexity: $O(T_k / \Delta t_1)$



Parallel Processing

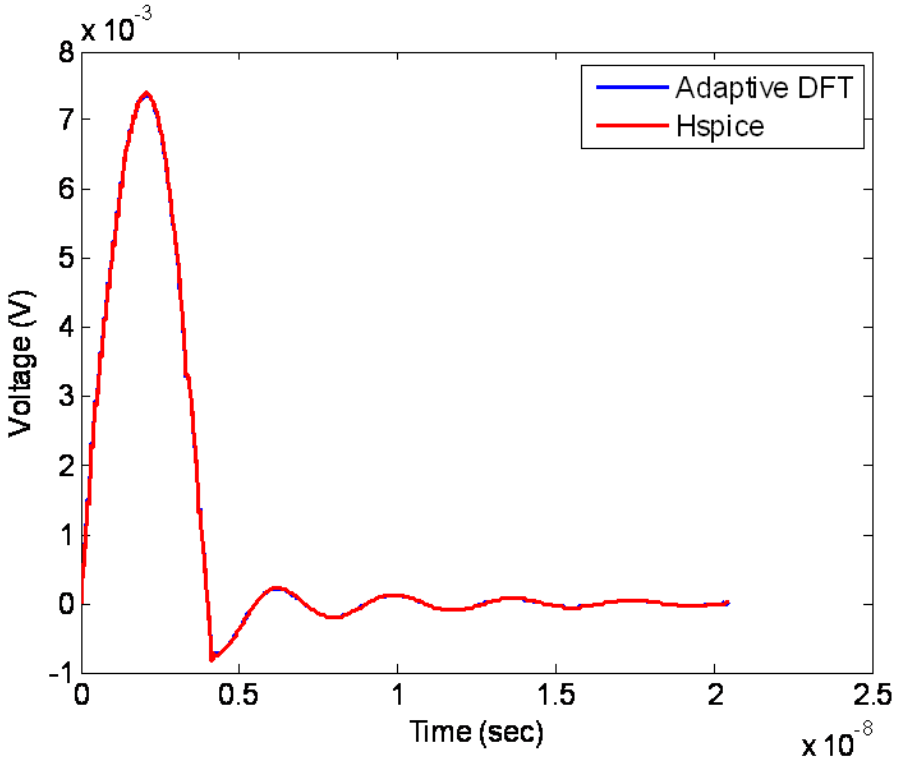
- Test case: 3D PDN
  - ~ 0.17 million nodes
- The adaptive DFT flow has more advantage when the number of available processors is limited.
- Simulations between each iteration of the adaptive flow need to be processed serially



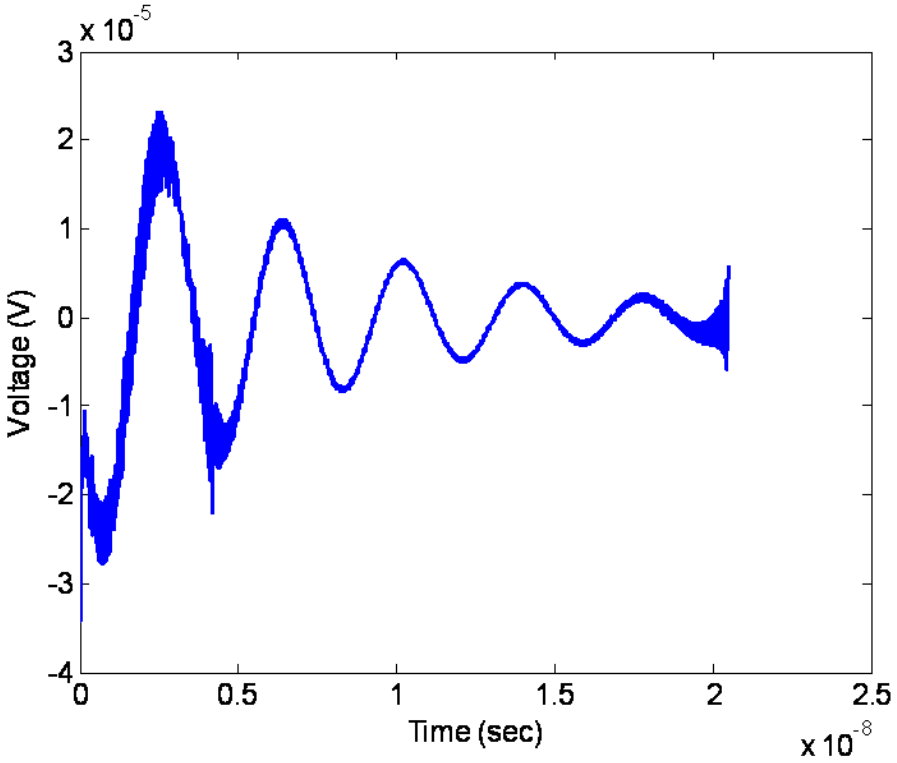
	Simulation time with different number of processors (sec)						
	1 prc	4 prcs	8 prcs	16 prcs	64 prcs	128 prcs	256 prcs
Hspice	21374	NA	NA	NA	NA	NA	NA
Basic DFT	15976	6635	3225	1647	425	218	143
Adaptive DFT	4947	2096	904	490	180	120	115

Parallel Processing: DFT Flow vs. SPICE

Simulation result



DFT error compared to Hspice



Outline

- Motivation
- Adaptive simulation flow using discrete Fourier transform (DFT)
  - Basic DFT flow
  - Problems with the basic DFT flow
  - Adaptive DFT flow
- Experimental results
  - Error analysis
  - Time efficiency of the adaptive flow
  - Time efficiency of parallel processing
- Conclusions

Conclusions

- Implemented an adaptive flow for large PDN simulation using DFT
- Total number of simulation points is reduced significantly compared to the basic DFT flow
- Achieved a relative error of the order of 0.1% compared to SPICE
- 10x speed up with a single processor compared to SPICE.
- Parallel processing is incorporated to reduce the simulation time even more significantly

Thank You !