TECHNISCHE UNIVERSITÄT DRESDEN

Entwurf und Implementierung eines hochparallelen Black-Scholes Monte-Carlo-Simulators

Patrick Russell
s0970860@mail.zih.tu-dresden.de

Dresden, 24. September 2015

Aufgabenstellung

Motivation

Entwurf

Implementierung & Test

Auswertung & Ausblick

01 Aufgabenstellung

- Implementierung eines Monte-Carlo-Simulators auf RT-Ebene (VHDL)
- Simulation des Preises einer Aktie mittels des Black-Scholes-Modells
- FIFOs als Benutzerschnittstelle
- Hardware Parallelisierung der Simulationsiterationen
- Generisch einstellbare Anzahl an Monte-Carlo-Kernen
- Auswertung des möglichen Speedups gegenüber einer Softwarelösung

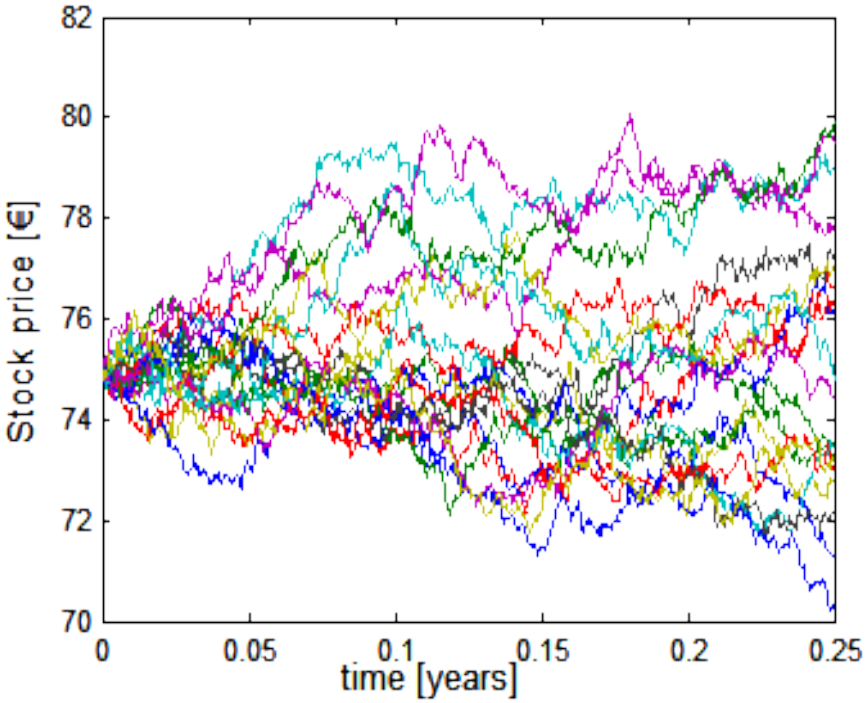
02 Motivation - Black-Scholes-Modell

- 1973 von Fischer Black und Myron Samuel entwickelt
- Mathematisches Modell zur Bewertung von Finanzoptionen
- Berechnen möglicher Preise einer Aktie zu einem gegebenen Zeitpunkt

$$S_t = S_0\left(1 + \left(\mu - \frac{\sigma^2}{2}\right)t + \varepsilon\sigma\sqrt{t}\right) \quad [1]$$

S_0 … Ausgangspreis der Aktie
t … Zeitpunkt, an dem der Aktienpreis bestimmt wird
μ … zu erwartender Zinssatz pro Zeiteinheit
σ … Volatilität pro Zeiteinheit
ε … Normalverteile Zufallszahl

Monte-Carlo Simulation



[3]

- Ermittlung des zu erwartenden Aktienpreises zum Zeitpunkt t
- Vorberechnung der Koeffizienten q und w

$$S_t = S_0\left(1 + \left(\mu - \frac{\sigma^2}{2}\right)t + \varepsilon\sigma\sqrt{t}\right)$$

$$S_t = S_0\left(q + \varepsilon w\right)$$

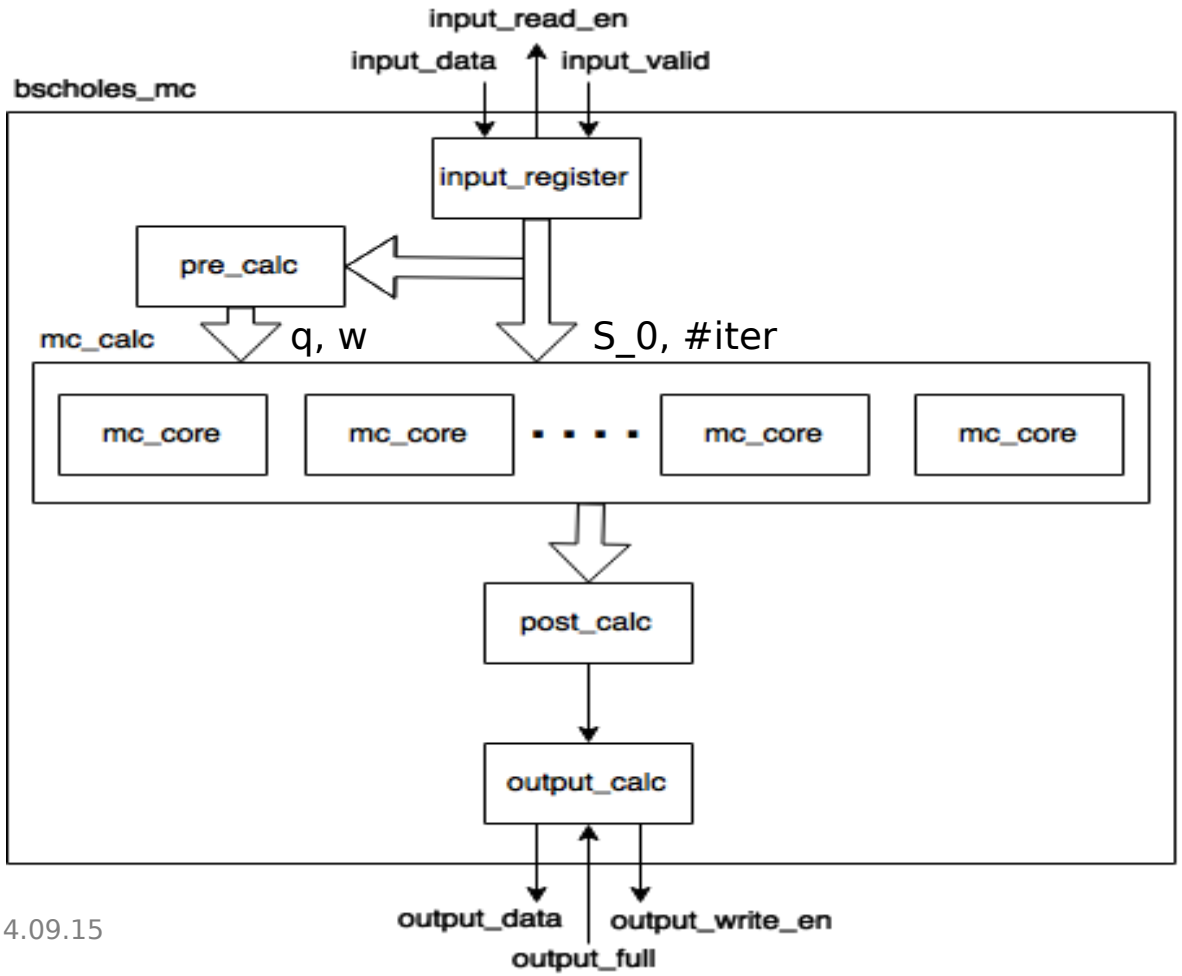
$$q = 1 + \left(\mu - \frac{\sigma^2}{2}\right)t$$

$$w = \sigma\sqrt{t}$$

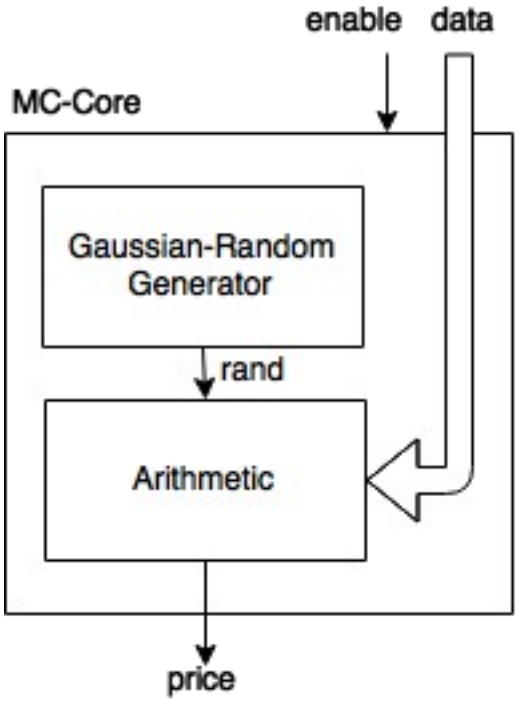
03 Entwurf - Simulationsphasen

1. Entgegennahme der Eingaben aus dem Eingangs-FIFO
   - #iter, S_0, μ, δ, t
2. Vorverarbeitung – Berechnung der Koeffizienten q und w
3. Durchführung der Simulation
   - Simulationsiterationen durchlaufen
   - Aufsummierung der Teilergebnisse
   - Berechnung des arithmetischen Mittelwertes (Division)
4. Ausgabe des Ergebnisses in das Ausgabe-FIFO

Blockschaltbild



Aufbau – Monte-Carlo-Kern

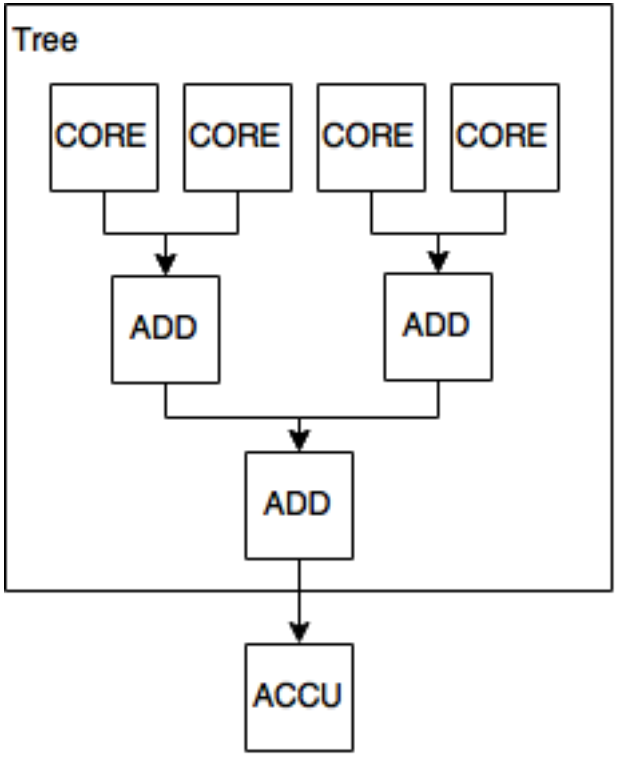


- Anzahl der Kerne bestimmt die Parallelität
- Jeder Kern benötigt einen eigenen Zufallszahlengenerator
- Jeder Kern durchläuft die vom Benutzer übergebene Anzahl an Iterationen
- Zufallszahlengenerator nicht Teil dieser Arbeit

04 Implementierung – Datenformat / -verarbeitung

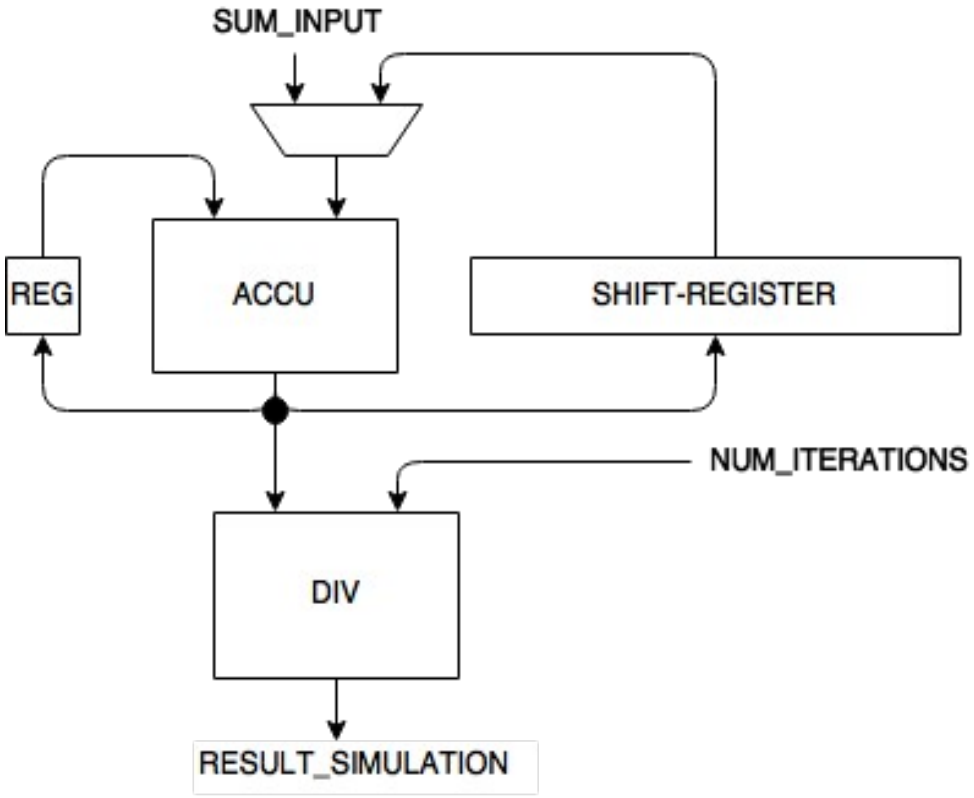
- Datenformat in Gleitkommadarstellung doppelter Genauigkeit
  => Ermöglicht großen Darstellungsbereich und einfache Weiterverarbeitung
- Datenverarbeitung mittels Floating-Point-IP-Cores von Xilinx optimiert auf Durchsatz und Taktfrequenz
- Händische Gleichverteilung des Ressourcenverbauchs
- Jeder Monte-Carlo-Kern erreicht einen Durchsatz von einer Simulationsiteration pro Takt

Nachbereitung und Ergebnisberechnung (1)



- Entgegennahme der Teilergebnisse aller Monte-Carlo-Kerne
- Realisierung durch Baumstruktur aus Addierern
- Baumtiefe steigt logarithmisch mit der Anzahl an Kernen
- Aufaddierte Teilergebnisse werden an einen Akkumulator weitergegeben

Nachbereitung und Ergebnisberechnung (2)



- Akkumulator nicht kombinatorisch
- Hält mehrere Teilsummen
- Aufsummierung und Ergebnis-berechnung nach Simulationsende

05 Auswertung - Funktionsumfang

- Hochparalleler generischer Black-Scholes Monte-Carlo-Simulator
- Berechnung des zu erwartenden Aktienpreis
- Simulationsdurchsatz von einer Simulationsiteration pro Takt
- Sinnvolle Zufallszahlengeneratoren noch nicht implementiert

Simulatorphase	#Takte
Vorverarbeitung	75
MC-Simulation	$47 + N$
Warten auf Nachbearbeitung	$12 * \lceil log_2 C \rceil$
Berechnung des Ergebnisses	283

C ... Anzahl Monte-Carlo-Kerne

M ... Gesamtanzahl Simulationsiterationen

N ... Simulationsiterationen pro Kern (M/C)

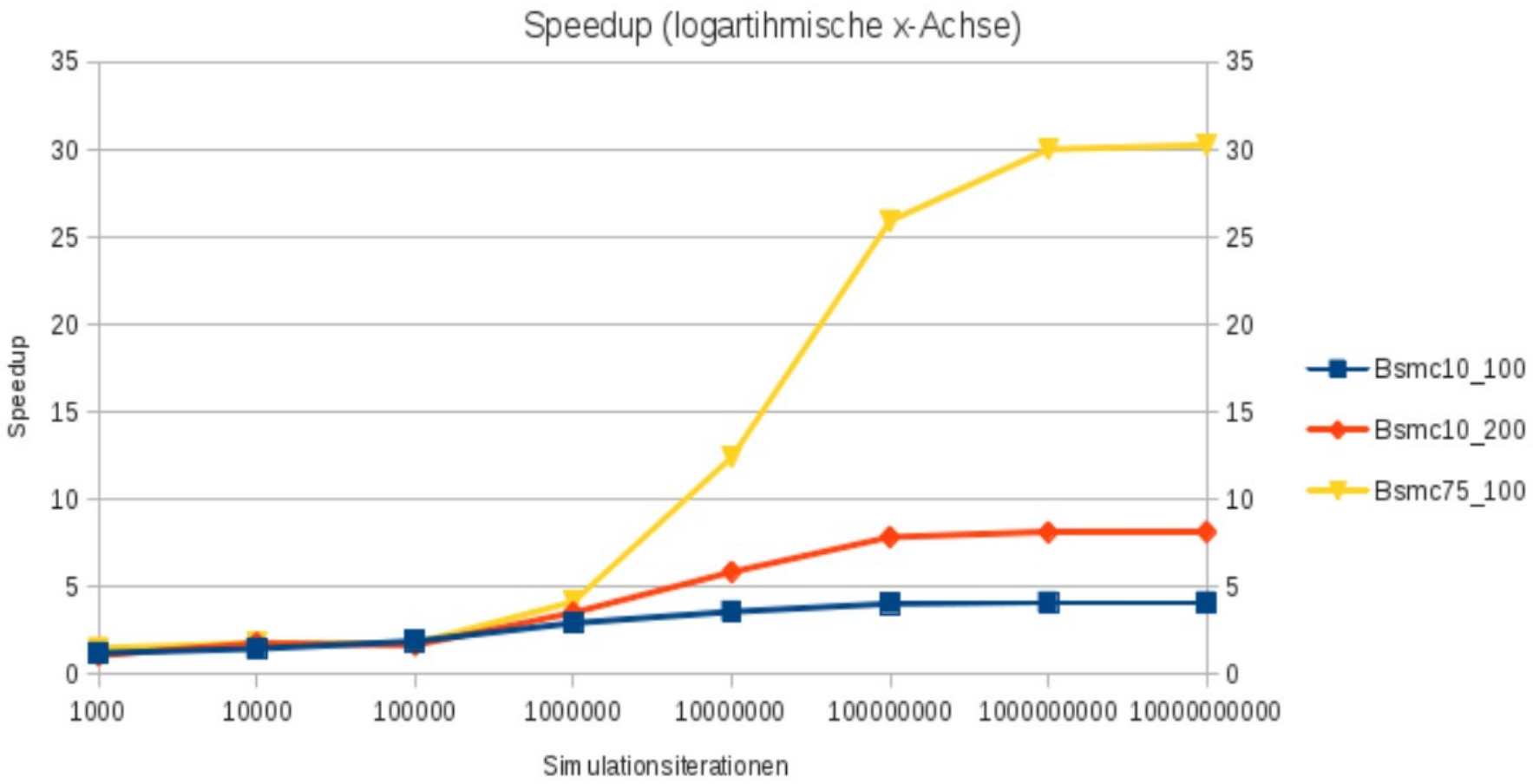
Ressourcenverbrauch und Taktfrequenz

	Virtex7 485t	bsmc_10	bsmc_75	MC-Core
Slice Register	607.200	58.308 (9%)	317.824 (52%)	~4000
Slice LUTs	303.600	44.057 (14%)	240.972 (79%)	~3000
DSP48E1s	2.800	295 (10%)	1.920 (68%)	25
Slices	75.900	17.082 (22%)	67625 (89%)	~777
Maximum frequency	-	261.363Mhz	263Mhz	-

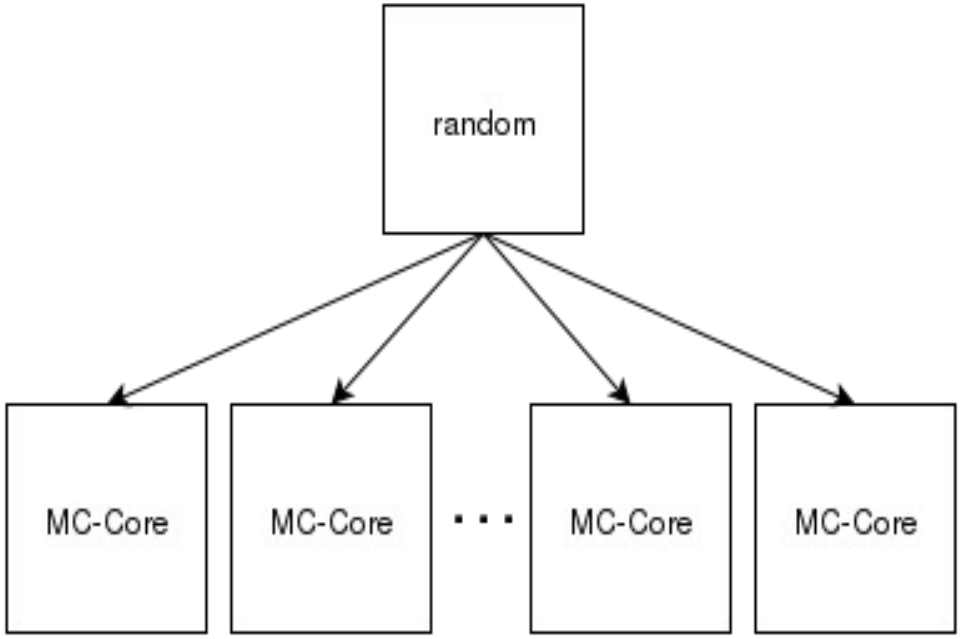
Geschwindigkeitsvergleich – Software I

- Vergleich der Ausführungszeit des Simulators mit einer Softwarelösung
- Simulator mit 10 bzw. 75 Kernen und 100 bzw. 200 Mhz Taktfrequenz implementiert
- Zufallszahlengenerator lieferte alle drei Takte eine neue Zahl
- Software lief auf einem Intel(R) Core(TM) i7-2600K CPU @3.40Ghz Prozessor mit 4 Kernen (GCC 5.1, OpenMP, Hyperthreading, -03 Optimierungsoption)

Geschwindigkeitsvergleich – Software II

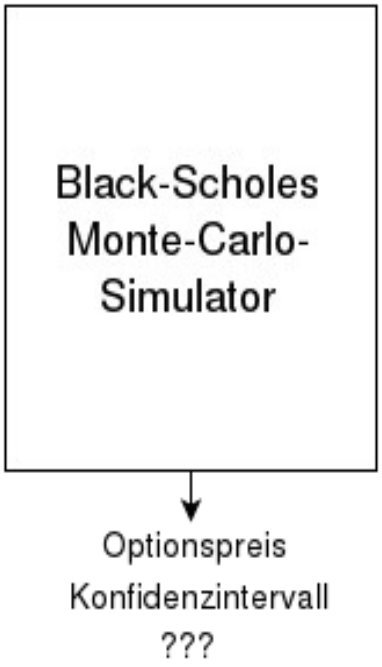


06 Ausblick - Zufallszahlen



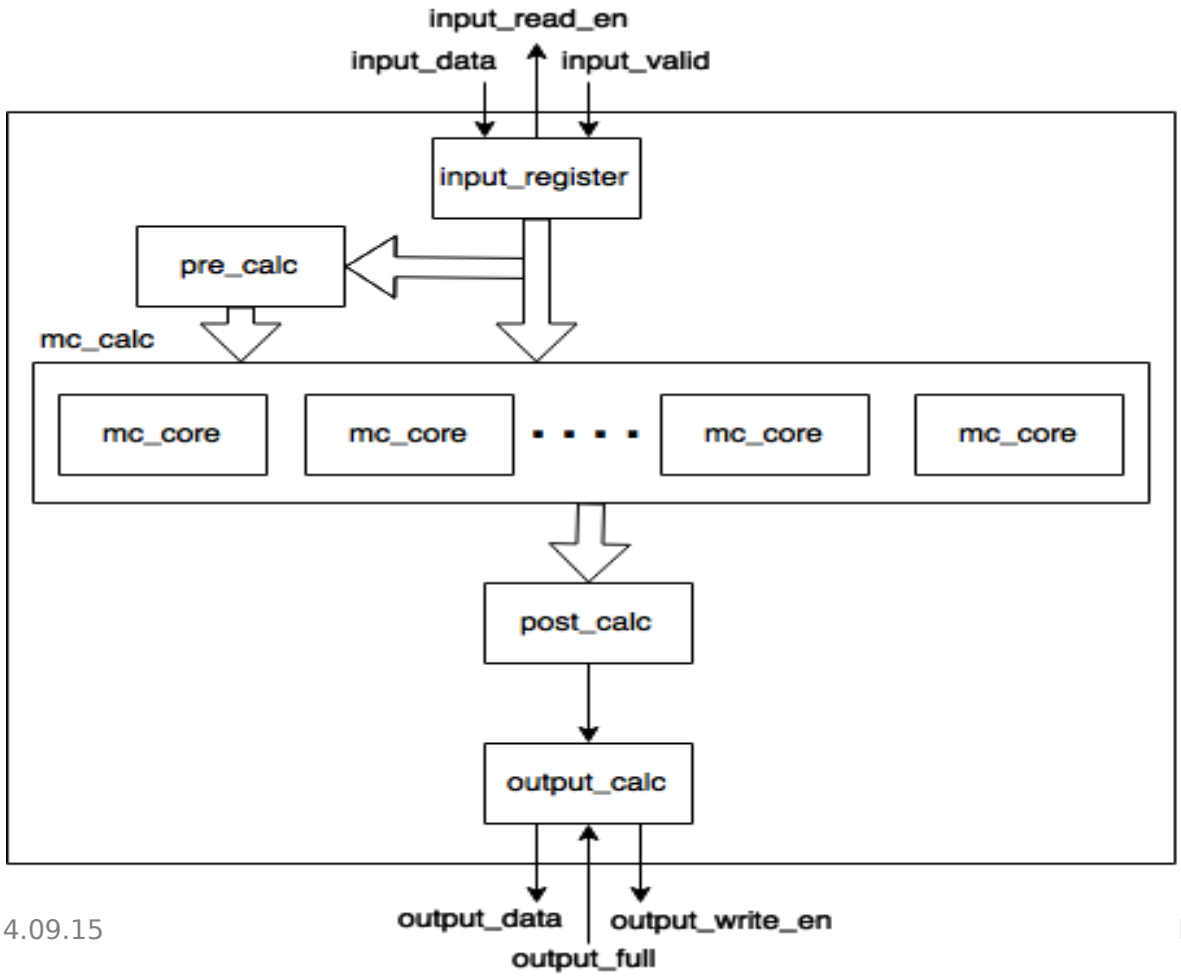
- Unterschiedliche Seed-Werte für jeden Monte-Carlo-Kern
- Übergeordneter Generator generiert nötige Seed-Werte
- Benutzer übergibt Anfangswert

Simulation weiterer Größen



- Unkritische Implementierung für unabhängige Größen
- Zwischenspeicherung oder reproduzierbare Simulation für abhängige Größen

Verallgemeinerung des Simulators



Literatur

[1] Desmond J. Higham: „Black Scholes for Scientific Computing Students“, International Conference on Computational Science, Emory University, Atlanta, Georgia, USA, May 2005.

[2] Xiang Tian, Khaled Benkrid and Xiaochen Gu: „High Performance Monte-Carlo Based Option Pricing on FPGAs“, The University of Edinburgh, School of Electronics and Engineering, Scotland, UK, August 2008.

[3] https://www.cwi.nl/system/files/u480/monte-carlo.gif