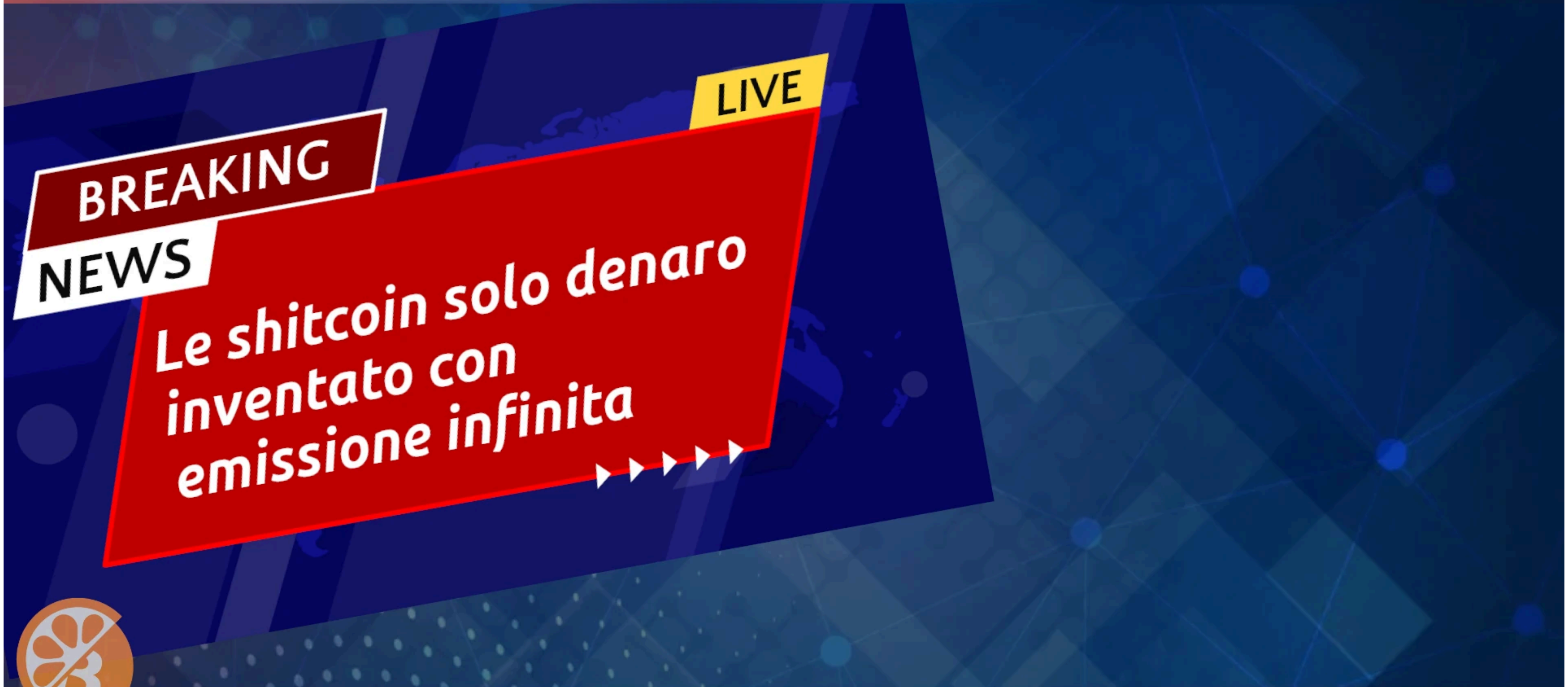
BITCOIN SUPPLY

FORMULA

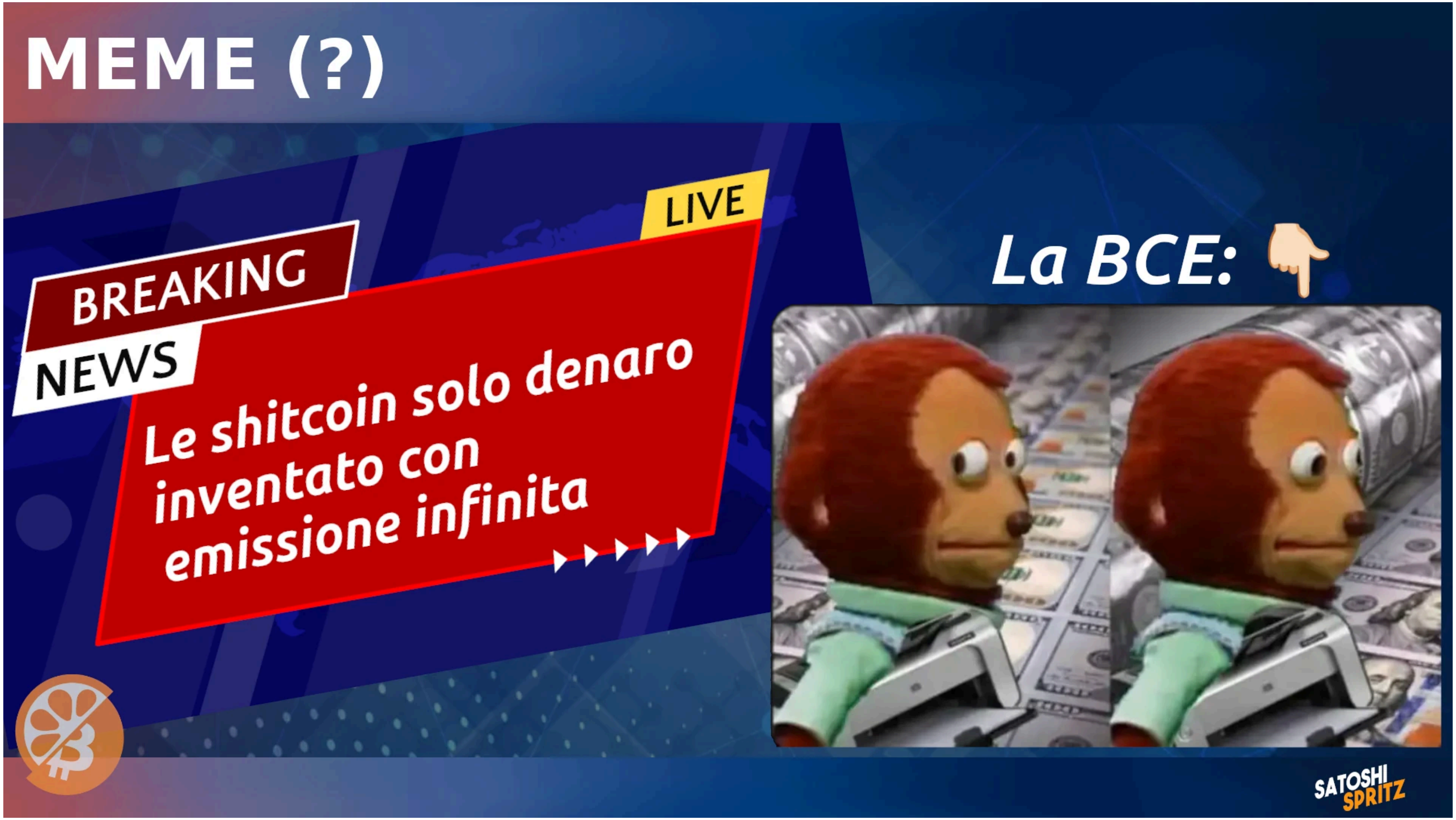
$$\sum_{i=0}^{32} 210.000 * \left[\frac{50}{2^i} \right]$$

SATOSHI SPRITZ

MEME (?)



SATOSHI SPRITZ


MEME (?)
BREAKING
NEWS
LIVE
Le shitcoin solo denaro inventato con emissione infinita
La BCE: 👇
SATOSHI SPRITZ

Bitcoin Supply Formula

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

ed one.

6. Incentive

By convention, the first transaction in a block is a special transaction that starts a new coin owned by the creator of the block. This adds an incentive for nodes to support the network, and provides a way to initially distribute coins into circulation, since there is no central authority to issue them. The steady addition of a constant of amount of new coins is analogous to gold miners expending resources to add gold to circulation. In our case, it is CPU time and electricity that is expended.

The incentive can also be funded with transaction fees. If the output value of a transaction is less than its input value, the difference is a transaction fee that is added to the incentive value of the block containing the transaction. Once a predetermined number of coins have entered circulation, the incentive can transition entirely to transaction fees and be completely inflation free.

The incentive may help encourage nodes to stay honest. If a greedy attacker is able to assemble more CPU power than all the honest nodes, he would have to choose between using it to defraud people by stealing back his payments, or using it to generate new coins. He ought to find it more profitable to play by the rules, such rules that favour him with more new coins than everyone else combined, than to undermine the system and the validity of his own wealth.

financial institutions cannot
increases transaction costs, limiting the
off the possibility for small casual transactions,
of ability to make non-reversible payments for non-

eclaiming Disk Space

Bitcoin Supply Formula

Una volta che un **numero prestabilito di monete** è entrato in circolazione, l'incentivo può passare interamente alle fee di transazione ed essere completamente privo di inflazione

SATOSHI SPRITZ

Bitcoin Supply Formula

$$\sum_{i=0}^{32} 210.000 * \left[\frac{50}{2^{i}} \right]$$

Bitcoin Supply Formula

$$\sum_{i=0}^{32} 210.000 * \left[\frac{50}{2^{i}} \right]$$

modo matematico ed elegante di definire

Bitcoin Supply Formula

$$\sum_{i=0}^{32} 210.000 * \left[\frac{50}{2^{i}} \right]$$

modo matematico ed elegante di definire

- l'emissione monetaria

SATOSHI SPRITZ

Bitcoin Supply Formula

$$\sum_{i=0}^{32} 210.000 * \left[\frac{50}{2^{i}} \right]$$

modo matematico ed elegante di definire

- l'emissione monetaria
- l'hard cap di 21 milioni

SATOSHI SPRITZ

Bitcoin Supply Formula

La formula è una

SOMMATORIA

Bitcoin Supply Formula

La formula è una
SOMMATORIA

SATOSHI SPRITZ

Bitcoin Supply Formula

La formula è una
SOMMATORIA

che vuol dire "somma di addizioni (+)"

SATOSHI SPRITZ

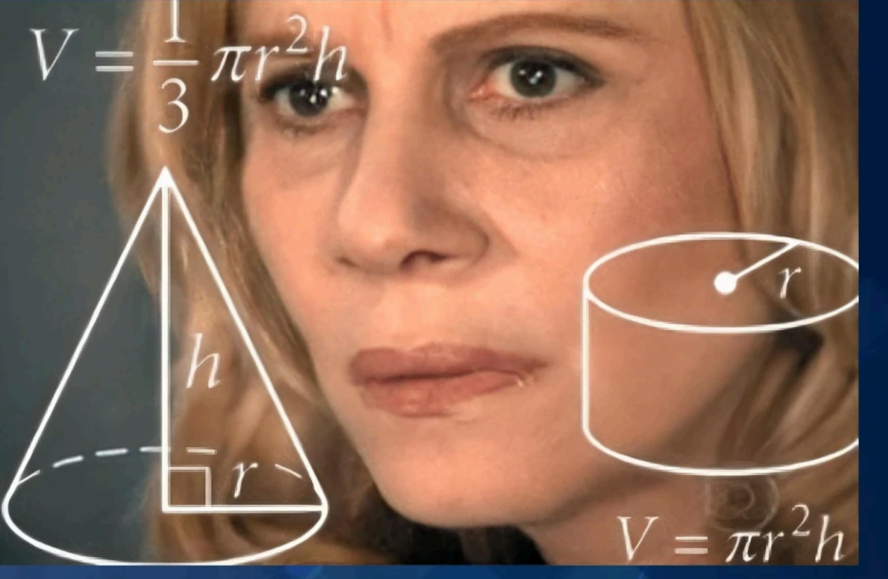
Bitcoin Supply Formula

Più precisamente è una
SERIE GEOMETRICA

SATOSHI SPRITZ

Bitcoin Supply Formula

Più precisamente è una
SERIE GEOMETRICA

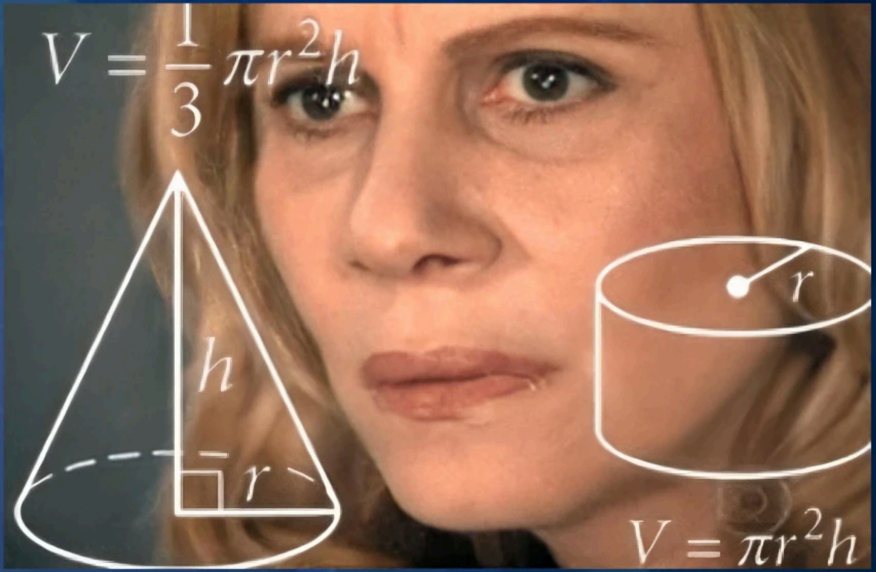


SATOSHI SPRITZ

Bitcoin Supply Formula

Più precisamente è una
SERIE GEOMETRICA

una **PROGRESSIONE**



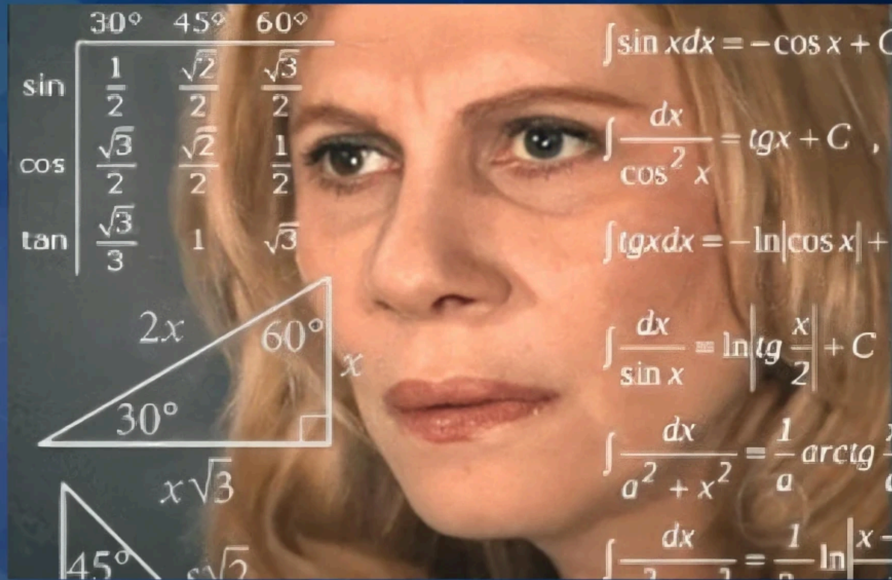
Bitcoin Supply Formula

...una funzione di
DECADIMENTO ESPONENZIALE

SATOSHI SPRITZ

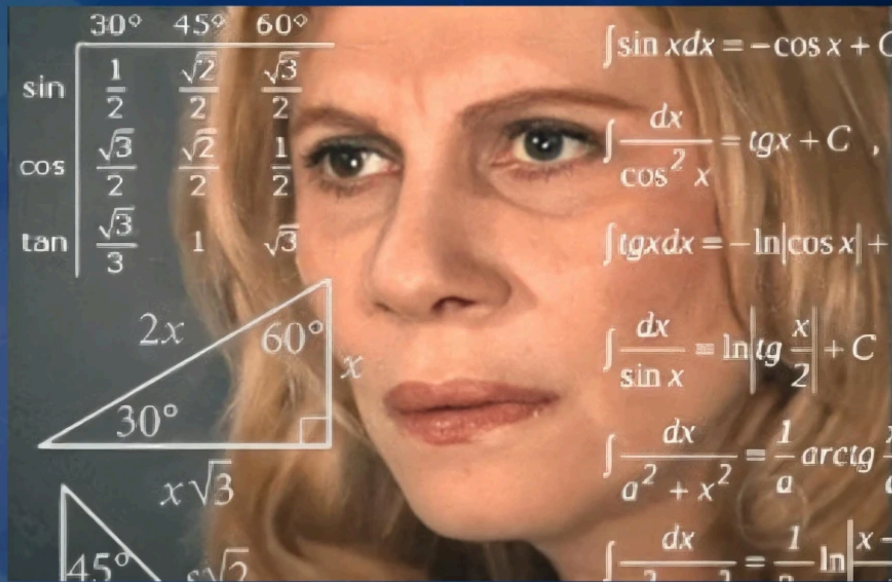
Bitcoin Supply Formula

...una funzione di
DECADIMENTO ESPONENZIALE



Bitcoin Supply Formula

...una funzione di
DECADIMENTO ESPONENZIALE



somma di ricompense decrescenti

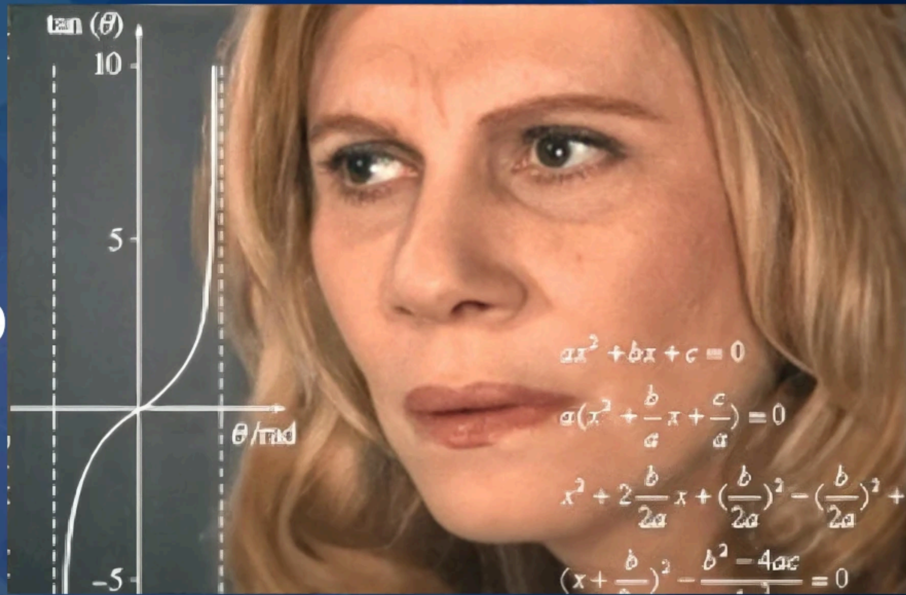
SATOSHI SPRITZ

Bitcoin Supply Formula

Essenzialmente un modello matematico che
- approssima la scarsità di materie prime come l'oro
- la ricompensa per blocco si dimezza ogni 210.000 blocchi
- il limite massimo è il risultato della sommatoria

Bitcoin Supply Formula

Essenzialmente un modello matematico che
- approssima la scarsità di materie prime come l'oro
- la ricompensa per blocco si dimezza ogni 210.000 blocchi
- il limite massimo è il risultato della sommatoria



SATOSHI SPRITZ

Bitcoin Supply Formula

Si tratta di una funzione matematica programmata che controlla la velocità con cui nuovi bitcoin entrano in circolazione, garantendo scarsità e prevedibilità, rendendolo un modello di **valuta a offerta controllata**.

SATOSHI SPRITZ

Bitcoin Supply Formula

EPOCA

SATOSHI SPRITZ

Bitcoin Supply Formula

EPOCA

Una grandezza matematica, una **variabile**

SATOSHI SPRITZ

Bitcoin Supply Formula

EPOCA

Una grandezza matematica, una **variabile**

Equivale a un **periodo di tempo arbitrario**

SATOSHI SPRITZ


Bitcoin Supply Formula
EPOCA
SATOSHI
SPRITZ


Bitcoin Supply Formula
EPOCA
1^ Epoca
SATOSHI SPRITZ

Bitcoin Supply Formula

EPOCA

2^ Epoca

Bitcoin Supply Formula

In Bitcoin le Epoche sono i periodi tra un halving ed un altro

SATOSHI SPRITZ

Bitcoin Supply Formula

$$\sum_{i=0}^{32} 210.000 * \left[\frac{50}{2^{i}} \right]$$

SATOSHI SPRITZ


Bitcoin Supply Formula
$\sum_{i=0}^{32} 210.000 \left[\frac{50}{2^i} \right]$
∑ Sigma
il simbolo della
sommatoria
SATOSHI
SPRITZ


Bitcoin Supply Formula
$\sum_{i=0}^{32} 210.000 * \left[\frac{50}{2^{i}} \right]$
i è la variabile
che indica le
Epoche.
i=0 Epoca iniziale
SATOSHI SPRITZ

Bitcoin Supply Formula

$$\sum_{i=0}^{32} 210.000 * \left[\frac{50}{2^{i}}\right]$$

Le epoche hanno dei *Limiti*. 0 è il limite inferiore ->

i=32 Epoca finale

Bitcoin Supply Formula

$$\sum_{i=0}^{32} 210.000 * \left[\frac{50}{2^{i}} \right]$$


Bitcoin Supply Formula
$$\sum_{i=0}^{32} 210.000 * \left[\frac{50}{2^i} \right]$$
Il # di blocchi che compone l'Epoca (tra gli halving)
SATOSHI SPRITZ


Bitcoin Supply Formula
32
i=0
210.000 *
50
2^i
Block reward iniziale
Che equivale anche alla massima B.R.
SATOSHI SPRITZ

Bitcoin Supply Formula

$$\sum_{i=0}^{32} 210.000 * \left[\frac{50}{2^{i}} \right]$$

SATOSHI SPRITZ

Bitcoin Supply Formula

$$\sum_{i=0}^{32} 210.000 * \left[\frac{50}{2^{i}} \right]$$

Bitcoin Supply Formula

$$\sum_{i=0}^{32} 210.000 * \left[\frac{50}{2^{i}} \right]$$

Il dimezzamento

SATOSHI SPRITZ

Bitcoin Supply Formula

$$210.000 * \left[\frac{50}{2^i} \right]$$

Bitcoin Supply Formula

$$210.000 * \left[\frac{50}{2^{i}} \right]$$

$$i$$

SATOSHI SPRITZ

Bitcoin Supply Formula

$$i = 0$$

$$210.000 * \left[\frac{50}{2^{i}} \right]$$

Bitcoin Supply Formula

$$210.000 * \left[\frac{50}{2^0} \right]$$

SATOSHI SPRITZ

Bitcoin Supply Formula

$$210.000 * \left[\frac{50}{2^{0}} \right]$$

$$210.000 * \left[\frac{50}{1} \right]$$

SATOSHI SPRITZ

Bitcoin Supply Formula

210.000 * 50

10.500.000

SATOSHI SPRITZ

Bitcoin Supply Formula

Bitcoin totali minati nella prima Epoca

03/01/2009 → 28/11/2012

10.500.000

SATOSHI SPRITZ

Bitcoin Supply Formula

$$i = 1$$

$$210.000 * \left[\frac{50}{2^{i}} \right]$$

SATOSHI SPRITZ

Bitcoin Supply Formula

$$210.000 * \left[\frac{50}{2^{1}} \right]$$

Bitcoin Supply Formula

$$210.000 * \left[\frac{50}{2^{1}} \right]$$

$$210.000 * \left[\frac{50}{2} \right]$$

SATOSHI SPRITZ


Bitcoin Supply Formula
210.000 * 25
5.250.000
SATOSHI SPRITZ

Bitcoin Supply Formula

Bitcoin totali minati nella seconda Epoca

28/11/2012 → 09/07/2016

5.250.000

SATOSHI SPRITZ

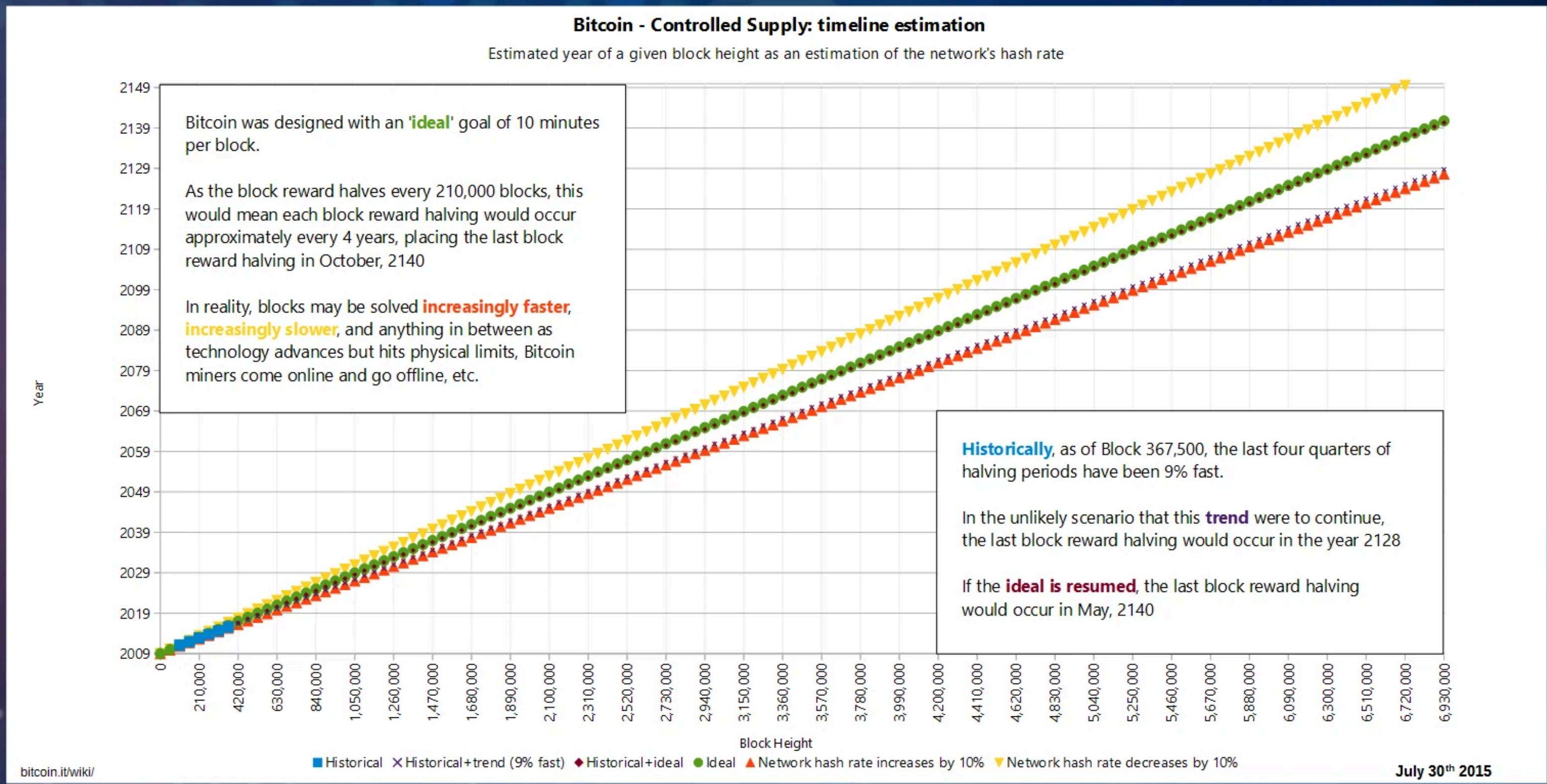
Bitcoin Supply Formula

Block	Epoca	₿/blocco	₿ (inizio)	₿ Aggiunti	Tot ₿ (parziale)	₿ aggiunti su 21M (relativo)	₿ aggiunti su 21M (assoluto)
0	1	50	0	10500000	10500000	50.00%	50.0000006%
210000	2	25	10500000	5250000	15750000	25%	75.0000008%
420000	3	12.5	15750000	2625000	18375000	12.5%	87.50000010%
630000	4	6.25	18375000	1312500	19687500	6.25%	93.75000010%
840000	5	3.125	19687500	656250	20343750	3.125%	96.87500011%
1050000	6	1.5625	20343750	328125	20671875	1.5625%	98.43750011%
1260000	7	0.78125	20671875	164062.5	20835937.5	0.78125%	99.21875011%
1470000	8	0.390625	20835937.5	82031.25	20917968.75	0.00390625%	99.60937511%
1680000	9	0.1953125	20917969	41015.625	20958984.375	0.001953125%	99.80468761%
1890000	10	0.09765625	20958984	20507.8125	20979492.1875	0.0009765625%	99.90234386%
2100000	11	0.04882812	20979492	10253.9052	20989746.0927	0.0004882812%	99.95117198%
2310000	12	0.02441406	20989746	5126.9526	20994873.0453	0.0002441406%	99.97558604%
2520000	13	0.01220703	20994873	2563.4763	20997436.5216	0.0001220703%	99.98779307%
2730000	14	0.00610351	20997437	1281.7371	20998718.2587	0.0000610351%	99.99389658%
2940000	15	0.00305175	20998718	640.8675	20999359.1262	0.0000305175%	99.99694833%

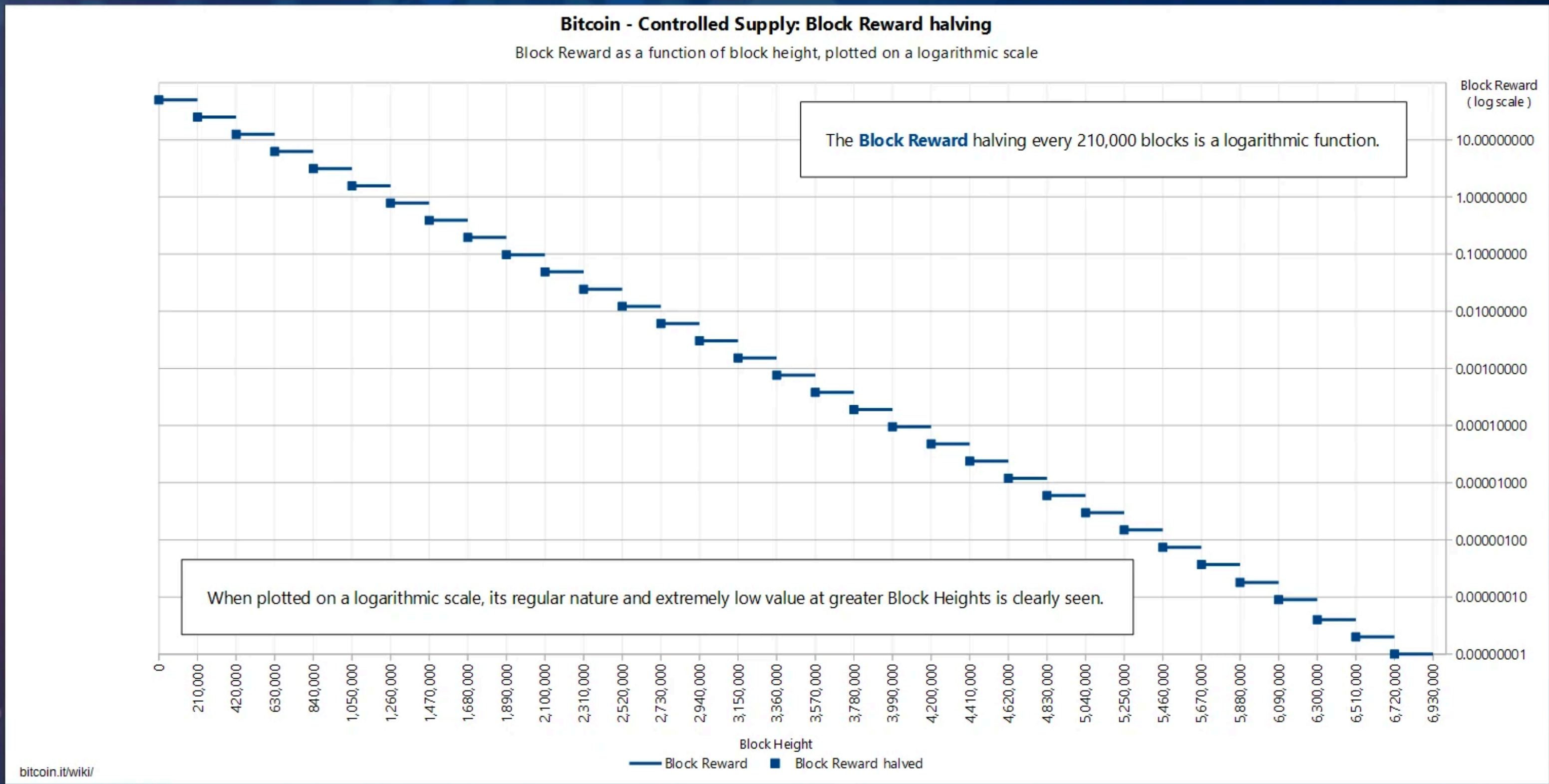
(la progressione)

SATOSHI SPRITZ

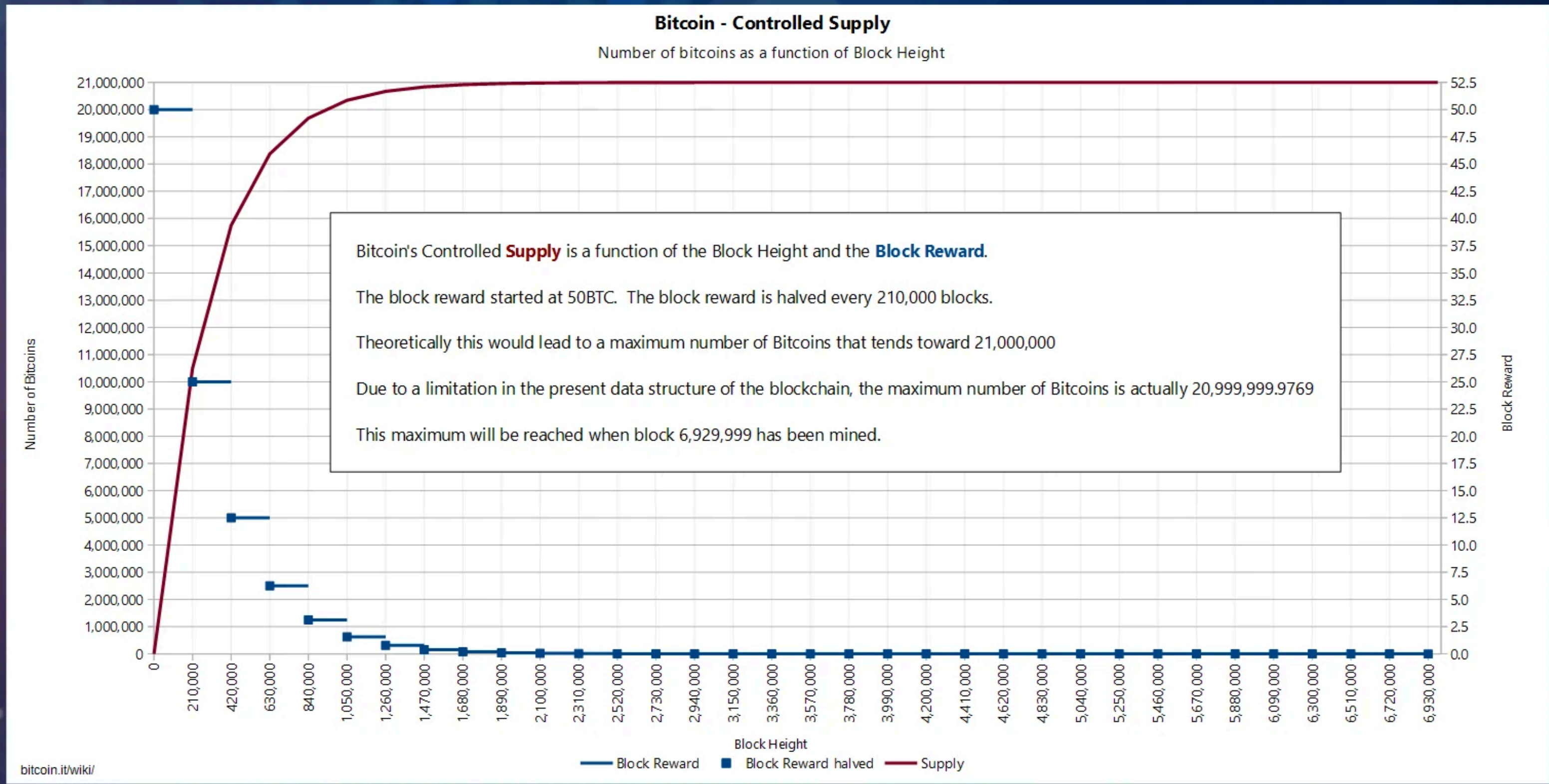
Bitcoin Supply Formula



Bitcoin Supply Formula



Bitcoin Supply Formula



Bitcoin Supply Formula

SATOSHI SPRITZ

Bitcoin Supply Formula

Bitcoin Supply Formula

Bitcoin Supply Formula

Mix regole di **CONSENSO**

SATOSHI SPRITZ

Bitcoin Supply Formula

Mix regole di **CONSENSO**

Pura **MATEMATICA**

Bitcoin Supply Formula

```
CAmount GetBlockSubsidy(int nHeight, const Consensus::Params& consensusParams)
{
    int halvings = nHeight / consensusParams.nSubsidyHalvingInterval;
    // Force block reward to zero when right shift is undefined.
    if (halvings >= 64)
        return 0;

    CAmount nSubsidy = 50 * COIN;
    // Subsidy is cut in half every 210,000 blocks which will occur
    approximately every 4 years.
    nSubsidy >>= halvings;
    return nSubsidy;
}
```

Bitcoin Supply Formula

`GetBlockSubsidy():`

fondamentale perché **tutti i nuovi Bitcoin vengono messi in circolazione con le ricompense che i miner si aggiudicano ogni volta che aggiungono un nuovo blocco alla blockchain**

SATOSHI SPRITZ

Bitcoin Supply Formula

```
CAmount GetBlockSubsidy(int nHeight, const Consensus::Params&
consensusParams)
{
    int halvings = nHeight /
    consensusParams.nSubsidyHalvingInterval;
    // Force block reward to zero when right shift is undefined.
    if (halvings >= 64)
        return 0;

    CAmount nSubsidy = 50 * COIN;
    // Subsidy is cut in half every 210,000 blocks which will
    occur approximately every 4 years.
    nSubsidy >>= halvings;
    return nSubsidy;
}
```

Bitcoin Supply Formula

```
CAmount GetBlockSubsidy(int nHeight, const Consensus::Params&
consensusParams)
{
    int halvings = nHeight /
    consensusParams.nSubsidyHalvingInterval;
    // Force block reward to zero when right shift is undefined.
    if (halvings >= 64)
        return 0;

    CAmount nSubsidy = 50 * COIN;
    // Subsidy is cut in half every 210,000 blocks which will
    occur approximately every 4 years.
    nSubsidy >>= halvings;
    return nSubsidy;
}
```

Apre la funzione e specifica che ciò che segue sono i parametri per la *block subsidy*

Bitcoin Supply Formula

```
CAmount GetBlockSubsidy(int nHeight, const Consensus::Params&
consensusParams)
{
    int halvings = nHeight /
    consensusParams.nSubsidyHalvingInterval;
    // Force block reward to zero when right shift is undefined.
    if (halvings >= 64)
        return 0;

    CAmount nSubsidy = 50 * COIN;
    // Subsidy is cut in half every 210,000 blocks which will
    occur approximately every 4 years.
    nSubsidy >>= halvings;
    return nSubsidy;
}
```

Bitcoin Supply Formula

```
CAmount GetBlockSubsidy(int nHeight, const Consensus::Params&
consensusParams)
{
    int halvings = nHeight /
    consensusParams.nSubsidyHalvingInterval;
    // Force block reward to zero when right shift is undefined.
    if (halvings >= 64)
        return 0;

    CAmount nSubsidy = 50 * COIN;
    // Subsidy is cut in half every 210,000 blocks which will
    occur approximately every 4 years.
    nSubsidy >>= halvings;
    return nSubsidy;
}
```

È l'equazione che calcola il numero degli **halving**

Bitcoin Supply Formula

```
CAmount GetBlockSubsidy(int nHeight, const Consensus::Params&
consensusParams)
{
    int halvings = nHeight /
    consensusParams.nSubsidyHalvingInterval;
    // Force block reward to zero when right shift is undefined.



    CAmount nSubsidy = 50 * COIN;
    // Subsidy is cut in half every 210,000 blocks which will
    occur approximately every 4 years.
    nSubsidy >>= halvings;
    return nSubsidy;
}
```

Bitcoin Supply Formula

```
CAmount GetBlockSubsidy(int nHeight, const Consensus::Params&
consensusParams)
{
    int halvings = nHeight /
    consensusParams.nSubsidyHalvingInterval;
    // Force block reward to zero when right shift is undefined.


    CAmount nSubsidy = 50 * COIN;
    // Subsidy is cut in half every 210,000 blocks which will
    occur approximately every 4 years.
    nSubsidy >>= halvings;
    return nSubsidy;
}
```

Numero intero che identifica quanti halving sono avvenuti

Bitcoin Supply Formula

```
CAmount GetBlockSubsidy(int nHeight, const Consensus::Params&
consensusParams)
{
    int halvings = nHeight /
    consensusParams.nSubsidyHalvingInterval;
    // Force block reward to zero when right shift is undefined.
    if (halvings >= 64)
        return 0;

    CAmount nSubsidy = 50 * COIN;
    // Subsidy is cut in half every 210,000 blocks which will
    occur approximately every 4 years.
    nSubsidy >>= halvings;
    return nSubsidy;
}
```

Bitcoin Supply Formula

```
CAmount GetBlockSubsidy(int nHeight, const Consensus::Params&
consensusParams)
{
    int halvings = nHeight /
    consensusParams.nSubsidyHalvingInterval;
    // Force block reward to zero when right shift is undefined.
    if (halvings >= 64)
        return 0;

    CAmount nSubsidy = 50 * COIN;
    // Subsidy is cut in half every 210,000 blocks which will
    occur approximately every 4 years.
    nSubsidy >>= halvings;
    return nSubsidy;
}
```

Il numero attuale di blocchi sulla blockchain

SATOSHI SPRITZ

Bitcoin Supply Formula

```
CAmount GetBlockSubsidy(int nHeight, const Consensus::Params&
consensusParams)
{
    int halvings = nHeight /
    consensusParams.nSubsidyHalvingInterval;
    // Force block reward to zero when right shift is undefined.
    if (halvings >= 64)
        return 0;

    CAmount nSubsidy = 50 * COIN;
    // Subsidy is cut in half every 210,000 blocks which will
    occur approximately every 4 years.
    nSubsidy >>= halvings;
    return nSubsidy;
}
```

Bitcoin Supply Formula

```
CAmount GetBlockSubsidy(int nHeight, const Consensus::Params&
consensusParams)
{
    int halvings = nHeight /
    consensusParams.nSubsidyHalvingInterval;
    // Force block reward to zero when right shift is undefined.
    if (halvings >= 64)
        return 0;

    CAmount nSubsidy = 50 * COIN;
    // Subsidy is cut in half every 210,000 blocks which will
    occur approximately every 4 years.
    nSubsidy >>= halvings;
    return nSubsidy;
}
```

Il numero di blocchi che devono passare prima che si verifichi un altro halving

Bitcoin Supply Formula

```
CAmount GetBlockSubsidy(int nHeight, const Consensus::Params&
consensusParams)
{
    int halvings = nHeight /
    consensusParams.nSubsidyHalvingInterval;
    // Force block reward to zero when right shift is undefined.
    if (halvings >= 64)
        return 0;

    CAmount nSubsidy = 50 * COIN;
    // Subsidy is cut in half every 210,000 blocks which will
    occur approximately every 4 years.
    nSubsidy >>= halvings;
    return nSubsidy;
}
```

Bitcoin Supply Formula

Un commento che aiuta sviluppatori e osservatori a comprendere meglio cosa sta succedendo con il codice

```
{
    consensusParams.nSubsidyHalvingInterval;
    // Force block reward to zero when right shift is undefined.
    if (halvings >= 64)
        return 0;

    CAmount nSubsidy = 50 * COIN;
    // Subsidy is cut in half every 210,000 blocks which will
    occur approximately every 4 years.
    nSubsidy >>= halvings;
    return nSubsidy;
}
```

Bitcoin Supply Formula

Un commento che aiuta sviluppatori e osservatori a comprendere meglio cosa sta succedendo con il codice

```
// Force block reward to zero when right shift is undefined.
if (halvings >= 64)
    return 0;
```

Correzione di un bug per forzare la ricompensa del blocco a zero in una circostanza:

if (halvings >=64) return 0;

SATOSHI SPRITZ

Bitcoin Supply Formula

```
CAmount GetBlockSubsidy(int nHeight, const Consensus::Params&
consensusParams)
{
    int halvings = nHeight /
    consensusParams.nSubsidyHalvingInterval;
    // Force block reward to zero when right shift is undefined.
    if (halvings >= 64)
        return 0;

    CAmount nSubsidy = 50 * COIN;
    // Subsidy is cut in half every 210,000 blocks which will
    occur approximately every 4 years.
    nSubsidy >>= halvings;
    return nSubsidy;
}
```

Bitcoin Supply Formula

calcolo vero e proprio della *block subsidy*.
Il valore di `COIN` è espresso in satoshi, 10^8 x 50 (valore della prima ricompensa)

```
CAmount GetBlockSubsidy(int nHeight, const Consensus::Params& consensusParams)
{
    int halvings = nHeight / consensusParams.nSubsidyHalvingInterval;
    // Force block reward to zero when right shift is undefined.
    if (halvings >= 64)
        return 0;

    CAmount nSubsidy = 50 * COIN;
    // Subsidy is cut in half every 210,000 blocks which will occur approximately every 4 years.
    nSubsidy >>= halvings;
    return nSubsidy;
}
```

SATOSHI SPRITZ

Bitcoin Supply Formula

```
CAmount GetBlockSubsidy(int nHeight, const Consensus::Params&
consensusParams)
{
    int halvings = nHeight /
    consensusParams.nSubsidyHalvingInterval;
    // Force block reward to zero when right shift is undefined.



    CAmount nSubsidy = 50 * COIN;
    // Subsidy is cut in half every 210,000 blocks which will
    occur approximately every 4 years.
    nSubsidy >>= halvings;
    return nSubsidy;
}
```

Bitcoin Supply Formula

```
CAmount GetBlockSubsidy(int nHeight, const Consensus::Params&
consensusParams)
{
    int halvings = nHeight /
    consensusParams.nSubsidyHalvingInterval;
    // Force block reward to zero when right shift is undefined.



    CAmount nSubsidy = 50 * COIN;
    // Subsidy is cut in half every 210,000 blocks which will
    occur approximately every 4 years.
    nSubsidy >>= halvings;
    return nSubsidy;
}
```

Bitcoin Supply Formula

```
CAmount GetBlockSubsidy(int nHeight, const Consensus::Params&
consensusParams)
{
    int halvings = nHeight /
    consensusParams.nSubsidyHalvingInterval;
    // Force block reward to zero when right shift is undefined.

    CAmount nSubsidy = 50 * COIN;
    // Subsidy is cut in half every 210,000 blocks which will
    occur approximately every 4 years.
    nSubsidy >>= halvings;
    return nSubsidy;
}
```

Funzione di calcolo
della **block subsidy**

Bitcoin Supply Formula

>> è l'operatore vero e proprio

```
CAmount GetBlockSubsidy(int nHeight, const Consensus::Params&
consensusParams)
{
    int halvings = nHeight /
    consensusParams.nSubsidyHalvingInterval;
    // Force block reward to zero when right shift is undefined.



    CAmount nSubsidy = 50 * COIN;
    // Subsidy is cut in half every 210,000 blocks which will
    occur approximately every 4 years.
    nSubsidy >>= halvings;
    return nSubsidy;
}
```

Bitcoin Supply Formula

>> è l'operatore vero e proprio

Il software sa quando effettuare e quando interrompere la divisione

```
CAmount GetBlockSubsidy(int nHeight, const Consensus::Params& consensusParams)
{
    int halvings = nHeight / consensusParams.nSubsidyHalvingInterval;
    // Force block reward to zero when right shift is undefined.

    CAmount nSubsidy = 50 * COIN;
    // Subsidy is cut in half every 210,000 blocks which will occur approximately every 4 years.
    nSubsidy >>= halvings;
    return nSubsidy;
}
```