MQAM transmitter

November 11, 2016

This block generates one or two optical signals. It can also generate a binary signal. It allows the user to controll the whole system. This block is a big block of code that combines all of the other in order to generate a optical signal. A schematic representation of this block is shown in figure 1.

Input parameters

This block has a special set of functions that allow the user to change the basic configuration of the transmitter. This functions are listed bellow.

- setMode(PseudoRandom);
- setBitPeriod(1.0/50e9);
  (double)
- setPatternLength(3);
  (int)
- setNumberOfBits(10000);
  (long)
- setNumberOfSamplesPerSymbol(32);
  (int)
- setRollOffFactor(0.9);
  (double $\in$ [0,1])
- setIqAmplitudes({ { 1, 1 }, { -1, 1 }, { -1, -1 }, { 1, -1 } });
- setOutputOpticalPower_dBm(0);
- setSaveInternalSignals(true);

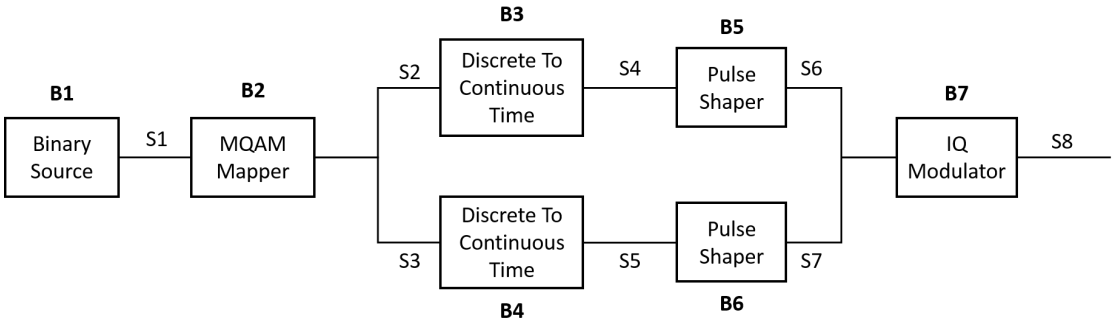

Figure 1: Schematic representation of the block MQAM transmitter.

Methods

MQamTransmitter(vector<Signal *> &inputSignal, vector<Signal *> &outputSignal);

void set(int opt);

void setMode(BinarySourceMode m) { B1.setMode(m); };

BinarySourceMode const getMode(void) { return B1.getMode(); };

void setProbabilityOfZero(double pZero) { B1.setProbabilityOfZero(pZero); };

double const getProbabilityOfZero(void) { return B1.getProbabilityOfZero(); };

void setBitStream(string bStream) { B1.setBitStream(bStream); };

string const getBitStream(void) { return B1.getBitStream(); };

void setNumberOfBits(long int nOfBits) { B1.setNumberOfBits(nOfBits); }

long int const getNumberOfBits(void) { return B1.getNumberOfBits(); }

void setPatternLength(int pLength) { B1.setPatternLength(pLength); }

int const getPatternLength(void) { return B1.getPatternLength(); }

void setBitPeriod(double bPeriod) {B1.setBitPeriod(bPeriod);}; double const getBitPeriod(void) { return B1.getBitPeriod(); }

void setM(int mValue){B2.m = mValue;}; int const getM(void) { return B2.m; };

void setNumberOfSamplesPerSymbol(int n){ B3.setNumberOfSamplesPerSymbol(n); B4.setNumberOfSamplesPe };

int const getNumberOfSamplesPerSymbol(void){ return B3.getNumberOfSamplesPerSymbol(); };

void setRollOffFactor(double rOffFactor){ B5.setRollOffFactor(rOffFactor); B6.setRollOffFactor(rOffFactor); };

double const getRollOffFactor(void){ return B5.getRollOffFactor(); };

void setSeeBeginningOfImpulseResponse(bool sBeginningOfImpulseResponse){ B5.setSeeBeginningOfImpulseRes B6.setSeeBeginningOfImpulseResponse(sBeginningOfImpulseResponse); };

double const getSeeBeginningOfImpulseResponse(void){ return B5.getSeeBeginningOfImpulseResponse(); };

Functional description

This block can generate one or two optical signals. If it generates two they have polarizations along perpendicular axis so that it is possible to combine them to produce a signal with any type of polarization. It also generates a binary signal that should be used to measure the Bit Error Rate (BER).

Output Signals

Number: 1 or 2 optical and 1 binary

Type: Optical signal

Example

Sugestions for future improvement